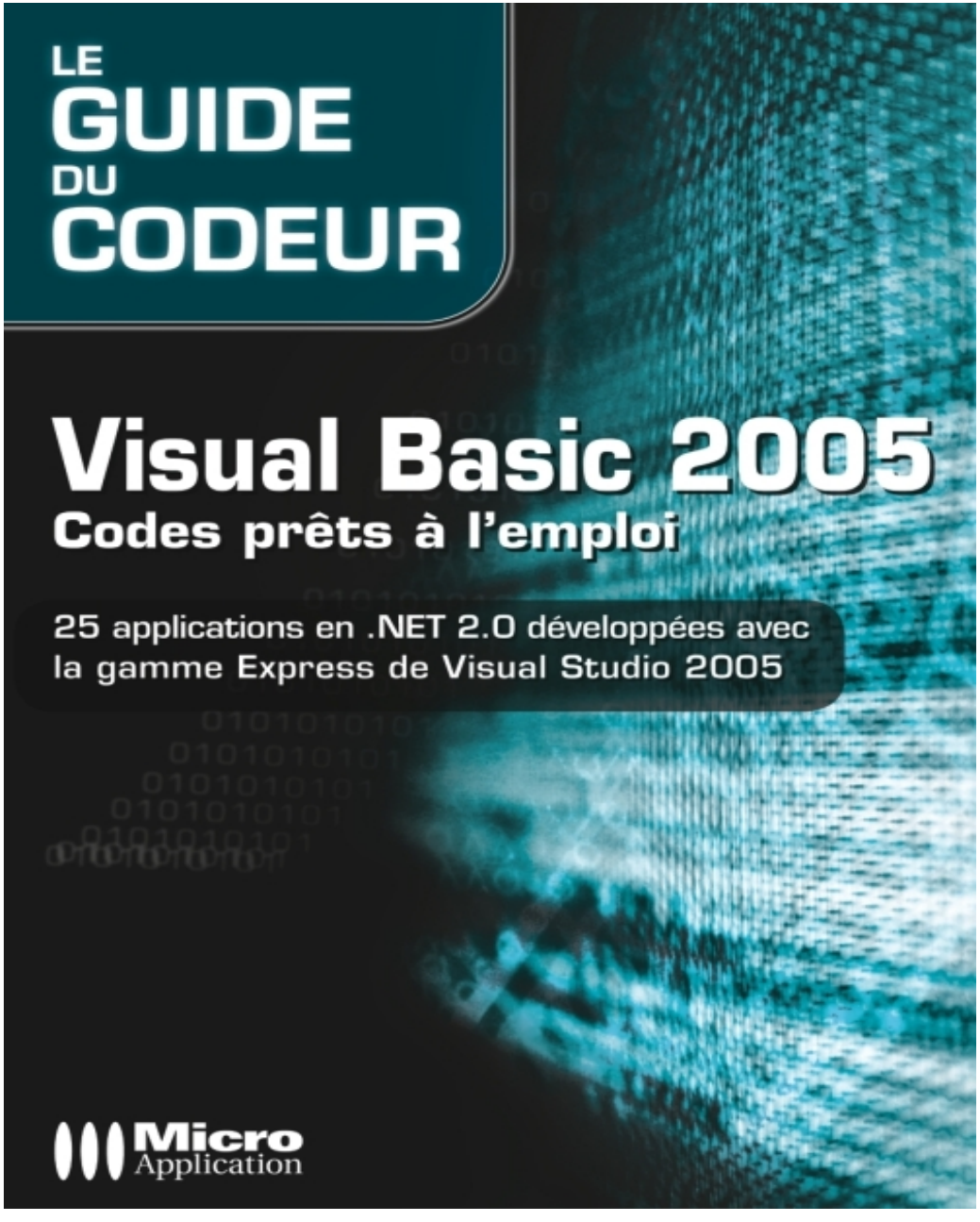

LE
GUIDE
DU
CODEUR
Visual Basic 2005
Codes prêts à l'emploi
25 applications en .NET 2.0 développées avec
la gamme Express de Visual Studio 2005
Micro
Application

1ère Édition - Septembre 2006

Auteurs

Patrice LAMARCHE, Antoine GRIFFARD, Mauricio DIAZ ORLICH

ISBN : 2-7429-6826-1

Couverture réalisée par Room22.

MICRO APPLICATION
20-22, rue des Petits-Hôtels
75010 PARIS
Tél. : 01 53 34 20 20
Fax : 01 53 34 20 00
http://www.microapp.com

Support technique
Également disponible sur
www.microapp.com

Retrouvez des informations sur cet ouvrage !

Rendez-vous sur le site Internet de Micro Application **www.microapp.com**. Dans le module de recherche, sur la page d'accueil du site, entrez la référence à 4 chiffres indiquée sur le présent livre. Vous accédez directement à sa fiche produit.

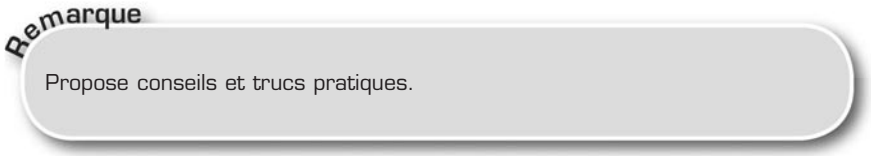
Avant-propos

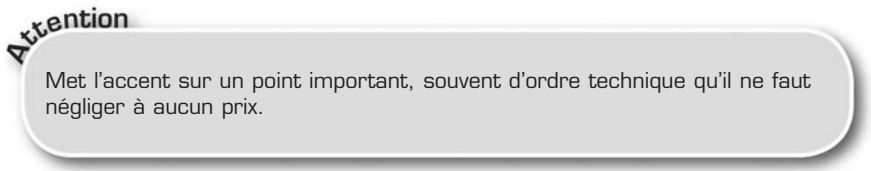
Le collection *Guide du codeur* s'adresse aux personnes initiées à la programmation qui souhaitent découvrir une technologie particulière. Sans négliger les aspects théoriques, nous donnons toujours priorité à la pratique afin que vous puissiez rapidement être autonome. Avant d'entrer dans le vif du sujet, notez ces quelques informations générales à propos de la collection.

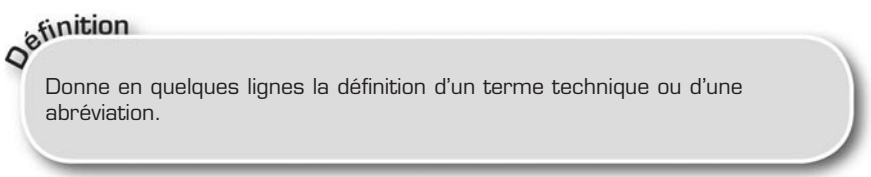
Conventions typographiques

Afin de faciliter la compréhension de techniques décrites, nous avons adopté les conventions typographiques suivantes :

- **gras** : menu, commande, boîte de dialogue, bouton, onglet.
- *italique* : zone de texte, liste déroulante, case à cocher, bouton radio.
- `Police bâton` : instruction, listing, texte à saisir.
- ➥ : dans les programmes, indique un retour à la ligne dû aux contraintes de la mise en page.







Astuce

Il s'agit d'informations supplémentaires relatives au sujet traité.

Sommaire

Présentation

Cet ouvrage est le fruit de l'expérience de trois auteurs : Antoine Griffard et Patrice Lamarche (MVP) sont membres de l'équipe Wygwam, composée d'experts .NET reconnus par leurs pairs et par Microsoft, qui les désigne comme Most Valuable Professionals et Regional Director. Mauricio Díaz est formateur à SUPINFO, où il donne des cours pour le Laboratoire .NET.

Cet ouvrage vise à vous apprendre à programmer grâce à la gamme d'outils gratuits Visual Studio Express de Microsoft, qui comprend entre autres :

- Visual Basic 2005 Express Edition, pour le développement d'applications Windows en Visual Basic ;
- Visual Web Developer Express Edition, pour le développement d'applications web ;
- SQL Server 2005 Express Edition, pour la gestion de bases de données.

Ces outils permettent aux étudiants, aux utilisateurs avancés, aux programmeurs débutants ou passionnés, de découvrir la programmation et d'approfondir leurs connaissances grâce à des environnements de développement proches de ceux utilisés par les développeurs professionnels.

Il ne s'agit pas de vous noyer sous de nombreuses informations théoriques à propos du développement d'applications, de la programmation orientée objet. Vous serez guidé pas à pas pour réaliser quelques exemples d'applications, prétextes à la découverte d'un domaine que l'on pense souvent réserver à des génies de l'informatique. Cela pouvait être vrai auparavant, mais c'est loin d'être le cas à présent, notamment grâce à Visual Basic. Ce langage simple et puissant permet de réaliser rapidement des applications aussi variées que des sites web personnels, des gestionnaires de performances ou encore des traitements de texte.

Alors, n'attendez plus, apprenez par la pratique à construire des applications avec Visual Basic 2005 Express Edition.

Pour télécharger les bootstrappers seulement (ceux-ci vont ensuite, une fois exécutés, lancer le téléchargement complet pour une installation dans la foulée) :

- Visual Basic 2005 Express : **http://download.microsoft.com/download/8/7/9/87938b02-80fa-430a-9e69-9a56a41d2096/vbsetup.exe**.
- Visual Web Developer 2005 Express : **http://download.microsoft.com/download/5/a/7/5a7c77cb-3ebe-4cc3-900a-e958488e686e/vwdsetup.exe**.
- SQL Server 2005 Express : **http://download.microsoft.com/download/5/6/1/561c80b2-e77f-4b0c-8c40-0a6512e136f5/SQLEXPR_FRN.EXE**.

Pour télécharger les images ISO (et graver un CD par exemple) :

- Visual Basic 2005 Express : **http://download.microsoft.com/download/7/5/b/75b12965-b2de-4a42-b5e1-bda6b2cf7c01/vb.iso**.
- Visual Web Developer 2005 Express : **http://download.microsoft.com/download/4/9/f/49f9dd2a-d537-4d40-a127-add7f7327a47/vwd.iso**.

Gestion simple d'une vidéothèque

Si vous avez une collection de CD, de DVD, de timbres postaux, etc., vous savez combien il est difficile de trouver un bon logiciel pour la gérer. Vous allez donc, au cours de ce chapitre, développer votre propre application qui vous permettra, d'une manière simple, de gérer votre collection de DVD.

Réaliser une application, basée sur une base de données, avec la plateforme .NET, est une tâche facile. Vous vous en rendrez compte à travers cet exemple, que vous pourrez adapter pour qu'il réponde mieux à vos besoins.

1.1 Configuration

En plus de Visual Basic 2005 Express, que vous utiliserez pour écrire votre programme, vous aurez besoin de SQL Server 2005 Express, qui sera votre système de gestion de bases de données (SGBD).

Ces deux produits sont mis à disposition gratuitement sur Internet par Microsoft.

Référez-vous aux annexes de ce livre pour les adresses.

1.2 Classes et espaces de noms utilisés

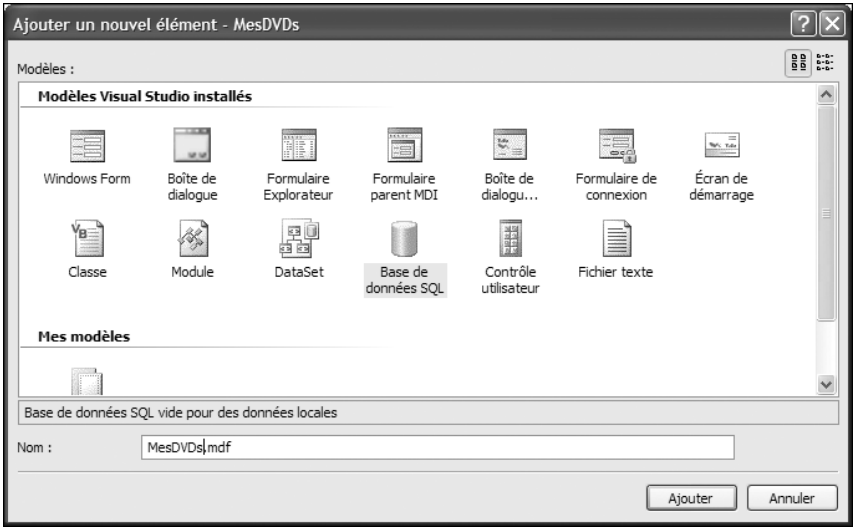
Vous ferez appel principalement à des classes dans les espaces de noms `System.Windows.Forms` et `System.Data`. Cependant, vous ne vous en rendrez peut-être pas compte car la plupart des manipulations se feront via le Concepteur de vues de Visual Basic 2005 Express.

1.3 Accès aux données

Une fois que vous aurez créé un projet de type *Application Windows*, vous pourrez commencer à travailler sur votre application en commençant par la création de la base de données sur laquelle elle se basera.

Création de la base de données

Pour créer votre base de données, cliquez du bouton droit sur le nom de votre projet et sélectionnez la commande **Nouvel élément** du sous-menu **Ajouter**. Saisissez `MesDVDs` comme nom de votre base.



▲ **Figure 1-1 :** Création d'une nouvelle base de données

L'Assistant Configuration de source de données se lance lorsque vous cliquez sur OK. Vous pouvez le fermer en cliquant sur **Terminer**, car, votre nouvelle base ne contenant pas d'éléments, il n'y a rien à configurer pour le moment. Remarquez les fichiers *MesDVDs.mdf* et *app.config*, qui sont apparus dans l'Explorateur de solutions.

Remarque

Travailler avec des bases de données dans Visual Basic 2005 Express

Le fichier visible dans l'Explorateur de solutions, et qui se trouve dans le répertoire de votre projet, correspond à la version de votre base de données qui sera modifiée par l'environnement de développement. Il fait office de base vierge et il est copié dans le répertoire de déploiement de votre application chaque fois que la solution est générée.

Le fichier original ne contient souvent que la structure de la base. C'est dans la copie sur laquelle vous travaillerez lorsque vous déboguerez votre application ou lorsque vous la déploierez que vous allez stocker vos données.

Le premier fichier correspond à votre base de données SQL Server 2005. Le deuxième est un fichier XML qui a été créé pour stocker, entre autres, la chaîne de connexion à la base.

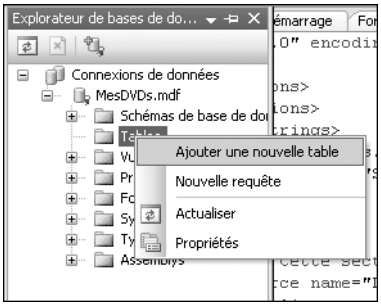
```
<connectionStrings>
  <add name="MesDVDs.My.MySettings.MesDVDsConnectionString"
connectionString="Data Source=.\SQLEXPRESS;
AttachDbFilename=|DataDirectory|\MesDVDs.mdf;
Integrated Security=True;User Instance=True"
providerName="System.Data.SqlClient" />
</connectionStrings>
```

▲ Chaîne de connexion à la base de données

Dans la chaîne de connexion sont spécifiés des paramètres tels que les identifiants et l'emplacement de la base. Par défaut, celle-ci se situe dans le même dossier que l'application.

Configuration des tables

Vous allez maintenant ajouter une table à votre base de données pour y stocker des informations. Double-cliquez sur le fichier *.mdf* que vous venez de créer : l'Explorateur de bases de données apparaît.



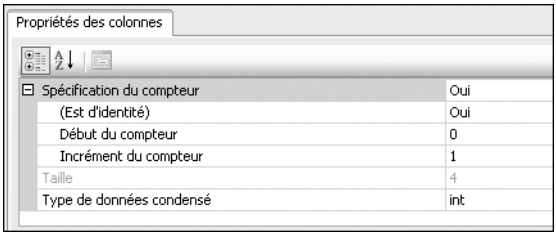
◄ **Figure 1-2 :** Ajouter des tables à votre base de données

Cliquez du bouton droit sur le dossier *Tables* pour en ajouter une nouvelle. Créez les champs de votre base conformément à l'image suivante :

Nom de la colonne	Type de données	Null autorisé
ID	int	☐
Titre	nvarchar(255)	☐
Description	nvarchar(MAX)	☑
Acteurs	nvarchar(255)	☑
Realisateur	nvarchar(255)	☑
Annee	int	☑
FichierImage	nvarchar(255)	☑

◄ **Figure 1-3 :** Champs de la base MesDVDs

Pour définir la clé primaire de votre table, cliquez du bouton droit sur le champ `ID` et sélectionnez la commande **Définir la clé primaire**. Vous allez aussi modifier les propriétés de ce champ afin que sa valeur soit auto-incrémentée à chaque insertion.



◄ **Figure 1-4 :** Auto-incrémenter la clé primaire

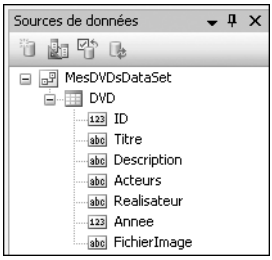
Pour finir, appelez votre table `DVD` en modifiant la propriété `(Nom)` dans le volet des propriétés.

Configuration de la source de données

Maintenant que votre table est configurée, elle peut être utilisée comme source de données par l'environnement de développement.

Affichez le volet *Sources de données* à l'aide de la commande **Afficher les sources de données** du menu **Données**. Vous y trouverez une source appelée `MesDVDsDataSet` qui a été créée automatiquement en même temps que votre base de données. Cependant, elle n'est pas encore liée à la table qui se trouve maintenant dans celle-ci. Pour la lier, cliquez sur le bouton **Configurer le DataSet à l'aide de l'Assistant** et sélectionnez la table `DVD` dans l'arborescence.

◄ **Figure 1-5 :** Création d'un DataSet typé



◄ **Figure 1-6 :** Les éléments de la table DVD

Maintenant que votre source de données est configurée, vous pouvez concevoir l'interface graphique de votre application.

1.4 Interface utilisateur

Votre application sera constituée d'une fenêtre qui affichera une liste de DVD dans sa partie supérieure et les détails du film sélectionné dans la partie inférieure.

▲ **Figure 1-7 :** Interface graphique de l'application terminée

Mise en place des composants

Votre fenêtre propose, dans sa partie supérieure, une liste de films qui se trouvent actuellement dans la base de données. Cette liste permettra de sélectionner un film pour afficher ses détails dans la partie inférieure. De plus, les éléments de la liste pourront être filtrés par des mots-clés saisis dans le champ de texte qui se trouve au-dessus de celle-ci.

Commencez par ajouter un contrôle `Label` auquel vous changerez la propriété `Text` afin qu'il affiche `Filtrer :` et ajoutez un contrôle `TextBox` à côté de celui-ci. Modifiez la propriété (`Name`) du champ de texte en lui attribuant la valeur `Filtre`. Placez-les vers le haut du formulaire, mais laissez de la place pour insérer une barre d'outils par la suite.

En dessous des contrôles que vous venez d'ajouter, déposez un contrôle de type DataGridView. Ignorez les options de configuration de la balise active pour le moment. Vous allez y revenir une fois que tous les contrôles seront en place.

Glissez maintenant un contrôle GroupBox en dessous du DataGridView et placez une PictureBox et deux contrôles de type Button à l'intérieur. Saisissez Détails dans la propriété Text du contrôle GroupBox pour modifier son titre.

L'un des boutons servira à ouvrir une image et l'autre à la supprimer. Changez les propriétés (Name) et Text de ceux-ci en leur donnant les valeurs Ouvrir et Supprimer. Vous pouvez aussi attribuer une image aux boutons, si vous le désirez, en modifiant leur propriété Image.

Enfin, saisissez la valeur Zoom dans la propriété SizeMode du contrôle PictureBox afin que les images soient redimensionnées sans être déformées lorsqu'elles sont trop grandes pour leur conteneur.

Liaison des composants aux données

Avant de lier à votre base de données certains composants que vous venez d'insérer, vous allez utiliser Visual Basic 2005 Express pour insérer des contrôles. En ce sens, recourez à la liste qui se trouve dans le volet *Sources de données*.

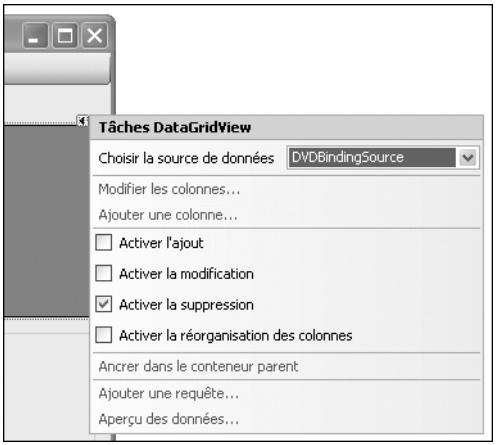
Glissez le contrôle Titre qui se trouve dans le volet *Sources de données* dans votre formulaire : deux contrôles, Label et TextBox, sont automatiquement créés et, plus important encore, ce dernier est automatiquement lié à votre base de données. Cela veut dire que, pour un enregistrement donné, votre application sera capable d'afficher dans ce champ la valeur de la colonne Titre de votre table DVD. Lorsque vous lui demanderez de faire une mise à jour, elle saura aussi quelle est la colonne à modifier.

Des contrôles sont apparus dans la section de contrôles non visibles de votre application :

- MesDVDsDataSet stockera temporairement les données que vous récupérerez à partir de la base de données pendant que vous travaillez dessus.
- DVDBindingSource servira de source de données aux contrôles liés à la base, comme le contrôle DataGridView que vous avez inséré précédemment et TextBox qui a été inséré lorsque vous avez glissé le champ Titre dans le formulaire.
- DVDTableAdapter est l'objet que vous allez utiliser pour communiquer avec la base de données.
- DVDBindingNavigator correspond à la barre d'outils qui est apparue en haut de votre formulaire.

Vous allez maintenant ajouter les autres composants correspondant aux colonnes de votre table, à commencer par l'`Année`. Toutefois, pour vous assurer que la valeur saisie pour l'année sera numérique, vous allez cliquer sur le bouton fléché qui apparaît à côté du contrôle pour sélectionner le type `NumericUpDown`. Lorsque vous glissez le contrôle sur le formulaire, un `Label` est ajouté, comme pour le titre, mais à la place du contrôle `TextBox`, vous aurez un contrôle `NumericUpDown`, qui ne peut contenir que des nombres. Ajoutez aussi les champs `Description`, `Acteurs` et `Réalisateur`.

Pour finir, liez le contrôle `DataGridView` à votre source de données à l'aide de sa balise active.



◄ **Figure 1-8 :** Lier le contrôle DataGridView à la source de données DVDBindingSource

Cliquez aussi sur le lien *Modifier les colonnes* pour supprimer toutes les colonnes, sauf *Titre* et *Description*.

Il ne reste plus qu'un contrôle à lier à votre base de données : `PictureBox`. Toutefois, vous devrez faire cette liaison en écrivant le code nécessaire à la main dans la section suivante.

1.5 Réalisation

La plupart de vos contrôles ont été liés à la source de données par l'environnement de développement de sorte que votre application est presque finie. Toutefois, il reste encore un contrôle à lier et quelques gestionnaires d'événements à écrire pour rendre l'utilisation du programme plus agréable.

Liaison de contrôles par le code

Afin de lier les contrôles restants à la source de données `DVDBindingSource`, double-cliquez sur le formulaire pour accéder au gestionnaire d'événements `Load`. En dessous de la ligne qui sert à charger le `MesDVDsDataSet` avec les données de la base, liez la propriété `ImageLocation` du contrôle `PictureBox` que vous avez inséré dans votre formulaire à la valeur de la colonne `FichierImage`.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles MyBase.Load
    Me.DVDTableAdapter.Fill(Me.MesDVDsDataSet.DVD)
    Me.PictureBox1.DataBindings.Add( _
      New Binding("ImageLocation", Me.DVDBindingSource, _
                  "FichierImage", True))
End Sub
```

▲ Liaison du contrôle PictureBox à la source de données

Maintenant que votre contrôle est lié à la source de données, vous pouvez sauvegarder le chemin de l'image affiché dans le contrôle `PictureBox` dans la base de données. Vous ne pouvez pas, pour l'instant, associer une image à ce contrôle.

Vous allez donc implémenter les gestionnaires d'événements `Click` des deux boutons que vous avez placés dans le formulaire pour associer, ou dissocier, une image et l'enregistrement courant.

```
Private Sub Ouvrir_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles Ouvrir.Click
   Dim OpenFileDialog As New OpenFileDialog
   If OpenFileDialog.ShowDialog = DialogResult.OK Then
       PictureBox1.ImageLocation = OpenFileDialog.FileName
   End If
End Sub

Private Sub Supprimer_Click(ByVal sender As System.Object, _
 ByVal e As System.EventArgs) Handles Supprimer.Click
   PictureBox1.ImageLocation = String.Empty
End Sub
```

▲ Associer et dissocier une image et l'enregistrement courant

La première méthode est appelée lorsque l'on clique sur le bouton **Ouvrir**, et elle utilise un contrôle `OpenFileDialog` pour chercher l'image à afficher dans le contrôle `PictureBox`. Le chemin de l'image est automatiquement associé à l'enregistrement courant, car la propriété `ImageLocation` est liée à la colonne `FichierImage`. La deuxième méthode sert simplement à effacer la valeur de la propriété `ImageLocation`, ce qui a pour effet d'effacer l'image du contrôle et de la base.

Attention

L'emplacement des images

Étant donné que dans la base de données ne sont stockés que les chemins vers les images, si vous supprimez ces dernières, elles ne pourront plus être affichées par l'application. Pensez, par exemple, à placer toutes vos images dans un dossier réservé à votre collection pour éviter ce problème.

Vous pouvez à présent ajouter des enregistrements dans la base de données à l'aide de la barre d'outils `DVDBindingNavigator` et de vos contrôles liés. Cependant, l'interface graphique vous autorise encore à faire des manipulations qui peuvent générer des erreurs. Pour éviter cela, vous aller créer quelques gestionnaires d'événements de plus dans la section suivante.

Améliorer l'ergonomie de l'application

Pour empêcher les utilisateurs de saisir des données quand aucun enregistrement n'est sélectionné, désactivez tous les contrôles liés à la base de données. Heureusement, tous ces contrôles se trouvent dans un contrôle `GroupBox`, ce qui veut dire qu'il suffit de modifier la propriété `Enabled` de ce dernier pour activer ou désactiver tous les contrôles qu'il contient.

Écrivez donc le gestionnaire d'événements `ListChanged` du contrôle `DVDBindingSource`, votre source de données. Ce type d'événement sera déclenché au démarrage de l'application et à chaque fois qu'un enregistrement de la base sera ajouté ou supprimé.

```
Private Sub DVDBindingSource_ListChanged( _
  ByVal sender As System.Object, _
  ByVal e As System.ComponentModel.ListChangedEventArgs) _
  Handles DVDBindingSource.ListChanged
    If DVDBindingSource.Count = 0 Then
        GroupBox1.Enabled = False
    Else
        GroupBox1.Enabled = True
        If TitreTextBox.Text.Length = 0 Then
            TitreTextBox.Focus()
        End If
    End If
End Sub
```

▲ Désactiver les contrôles quand aucun enregistrement n'est sélectionné

On vérifie le nombre d'enregistrements dans la source de données et on désactive le contrôle `GroupBox`, et donc tous les contrôles qui se trouvent à l'intérieur, si ce nombre est nul. Dans le cas contraire, les contrôles doivent être

activés. Une fois les contrôles activés, le curseur est placé dans le champ de texte correspondant au titre, s'il est vide, pour permettre à l'utilisateur de saisir une valeur.

Remarque

Tester l'application

Si vous testez votre application maintenant, vous pouvez utiliser la barre d'outils et les champs liés pour ajouter des enregistrements dans la base.

Toutefois, si vous lancez votre application à partir de Visual Basic 2005 Express, la base de données de l'application dans laquelle vous avez fait des modifications sera écrasée par la base de données vierge de la solution. Vous devez chercher l'exécutable de votre projet et le lancer directement si vous souhaitez tester la persistance des données.

Il y a un autre détail gênant dans le fonctionnement de votre application : si vous ne cliquez pas sur le bouton **Sauvegarder** de la barre d'outils avant de la quitter, vous perdez les données qui n'ont pas été enregistrées. Vous allez donc gérer l'événement `FormClosing` de votre formulaire pour enregistrer tous les changements avant de quitter l'application.

```
Private Sub Form1_FormClosing(ByVal sender As Object, _
  ByVal e As FormClosingEventArgs) _
  Handles MyBase.FormClosing
   Me.DVDBindingSource.EndEdit()
   If Me.MesDVDsDataSet.DVD.GetChanges() IsNot Nothing Then
       Me.DVDTableAdapter.Update(Me.MesDVDsDataSet.DVD)
   End If
End Sub
```

▲ Enregistrer les données avant de quitter l'application

Ce code termine l'édition de l'enregistrement courant et vérifie si la table a été modifiée grâce à la méthode `GetChanges`. Si c'est le cas, elle est mise à jour à l'aide de la méthode `Update` du `DVDTableAdapter`.

Filtrer la collection

Lorsque votre collection aura atteint une certaine taille, vous désirerez sûrement faire des recherches parmi vos DVD. Pour cette raison, vous avez prévu un champ de texte appelé *Filtrer* dans votre formulaire. Il s'agit, d'ailleurs, du seul contrôle de saisie qui n'a pas été lié à la source de données.

Vous allez créer un gestionnaire d'événements `TextChanged` pour ce champ de texte.

```
Private Sub Filtre_TextChanged(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Filtre.TextChanged
    If Filtre.Text.Length > 2 Then
        Me.DVDBindingSource.Filter = String.Format( _
         "{0} like '%{1}%' OR {2} like '%{1}%'", _
        Me.MesDVDsDataSet.DVD.TitreColumn.ColumnName, _
        Me.Filtre.Text, _
        Me.MesDVDsDataSet.DVD.DescriptionColumn.ColumnName)
    End If
End Sub
```

▲ Filtrer la source de données

Tout d'abord, on vérifie que le texte saisi comme filtre a, au moins, trois caractères. Une longueur inférieure nuirait aux performances de l'application puisque un trop grand nombre de résultats serait renvoyé.

Une fois la longueur du filtre vérifiée, il ne reste plus qu'à construire une clause `WHERE` pour l'appliquer au contrôle `DVDBindingSource`. La clause créée compare le texte saisi dans le champ *Filtre* avec le titre et la description de chaque enregistrement dans la base, dès que ce texte dépasse une longueur de deux caractères. Le fait d'utiliser l'événement `TextChanged` au lieu d'un bouton, par exemple, fait que les résultats sont affichés immédiatement au fur et à mesure que l'utilisateur saisit sa requête.

Votre application est maintenant prête à gérer votre collection de DVD.

1.6 Check-list

La réalisation de cette application vous a permis d'apprendre à :

- créer et gérer une base de données SQL Server 2005 Express à partir de Visual Basic 2005 Express ;
- utiliser une base de données SQL Server 2005 Express comme source de données de l'application ;
- lier des contrôles de l'application Windows aux données fournies par une source.

Gestion d'un album de photos

La plateforme .NET fournit des classes servant à charger, à manipuler et à afficher des images de différents formats. Vous allez utiliser ces classes pour écrire un programme permettant de visualiser et de réaliser des transformations simples sur des photographies et des images au format JPEG.

En développant cette application, vous apprendrez à vous servir des fonctionnalités de base de ces classes et vous découvrirez une astuce pour enregistrer des images qui sont chargées en mémoire dans leur fichier d'origine, une source d'erreurs souvent difficile à détecter.

2.1 Classes et espaces de noms utilisés

Il s'agit ici d'une application Windows. Vous aurez donc besoin de l'espace de noms `System.Windows.Forms`.

De plus, pour la manipulation d'images, vous aurez besoin de fonctionnalités de la librairie GDI+, qui sont encapsulées dans des classes se trouvant dans l'espace de noms `System.Drawing`. En particulier, les classes `Image` et `Bitmap` de l'espace de noms `System.Drawing` vous permettront d'effectuer toutes les opérations dont vous avez besoin.

Enfin, pour la manipulation des fichiers, vous aurez besoin des classes de l'espace de noms `System.IO`.

2.2 Interface utilisateur

Votre application sera constituée d'un seul formulaire qui affichera, à gauche, les aperçus des images contenues dans un répertoire choisi par l'utilisateur, et à droite, la photo sélectionnée en grand format ainsi que la barre d'outils servant à la manipuler.

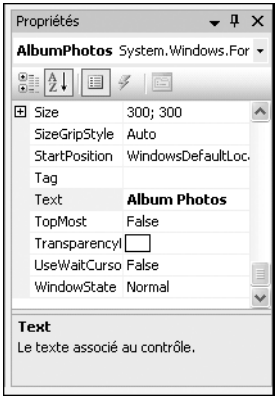
▲ **Figure 2-1 :** Interface graphique de l'album de photos

Préparer le formulaire principal

Commencez par créer un nouveau projet de type *Application Windows*. Il contiendra, par défaut, un formulaire de démarrage appelé `Form1`. Vous allez changer ce nom par un libellé plus parlant grâce à la propriété `(Name)` dans le volet des propriétés. Appelez votre formulaire `AlbumPhotos`.

◄ **Figure 2-2** : Personnaliser les propriétés d'un formulaire

Modifiez aussi la propriété `Text` en saisissant le texte `Album Photos`.



◄ **Figure 2-3** : Les propriétés

Ces deux modifications permettront non seulement de personnaliser l'interface graphique de votre application, mais aussi de rendre votre code auto-descriptif de manière à faciliter sa maintenance et sa compréhension par la suite.

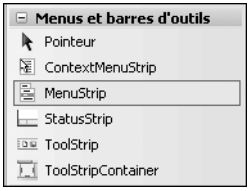
Enfin, avant de commencer à ajouter d'autres éléments de l'interface graphique, glissez un contrôle de type `FolderBrowserDialog` et un autre de type `SaveFileDialog` dans votre formulaire. Ces contrôles ne sont pas visibles dans le formulaire, mais ils vous permettront de sélectionner le répertoire dans lequel

se trouvent les fichiers que vous souhaitez parcourir ainsi que le nom d'une image lorsque vous voudrez l'enregistrer.

Barres de menus et d'état

Vous devez maintenant ajouter une barre de menus à votre application. Cette barre ne contiendra qu'un seul menu, en l'occurrence **Fichier**. Il permettra à l'utilisateur de charger dans l'application les images d'un répertoire sur le disque dur ou de quitter l'application.

Glissez, à partir de la boîte à outils, un contrôle de type `MenuStrip` dans votre formulaire. Ce contrôle se trouve dans la catégorie *Menus et barres d'outils*. Remarquez comme la barre vient se positionner automatiquement en haut de votre formulaire.



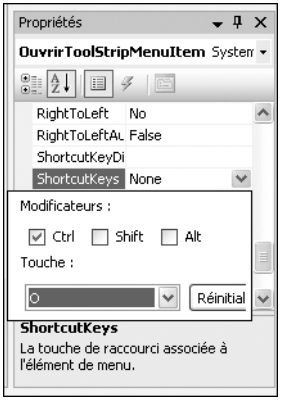
◄ **Figure 2-4 :** Insérer une barre de menus dans un formulaire

Appelez le premier, et le seul, menu de la barre `&Fichier`. L'esperluette devant le nom du menu permettra à l'utilisateur d'y accéder par un raccourci clavier.

◄ **Figure 2-5 :** Saisie du menu

Une fois que vous avez ajouté sous ce menu les commandes `&Ouvrir` et `&Quitter`, séparées par un `Separator`, cliquez sur la commande **Ouvrir** afin de configurer un raccourci clavier et une icône pour celle-ci.

Cherchez la propriété `ShortcutKeys` et cliquez sur son bouton fléché afin de spécifier le raccourci clavier Ctrl+O. Modifiez ensuite, si vous le souhaitez, la propriété `Image` afin de spécifier une icône à afficher à côté de la commande.



◄ **Figure 2-6 :** Attribuer un raccourci à une commande d'un menu

◄ **Figure 2-7 :** Menu Fichier terminé

Il ne reste plus qu'à insérer une barre d'état dans le formulaire en glissant, à partir de la boîte à outils, un contrôle de type `StatusStrip`. La barre viendra se positionner en bas du formulaire, permettant aux utilisateurs de l'application de le redimensionner plus facilement.

Diviser un formulaire en deux

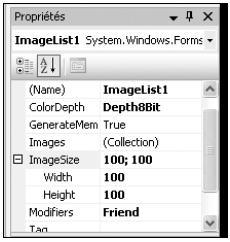
Le menu et la barre d'état en place, vous pouvez diviser l'espace restant de votre application en deux parties : celle de gauche affichera les miniatures des fichiers que l'utilisateur aura chargés grâce à la commande **Ouvrir**, et celle de droite affichera une version grand format de l'image que l'utilisateur aura sélectionnée.

Pour diviser votre formulaire, glissez dessus un contrôle de type `SplitContainer`. Ce contrôle se trouve dans la catégorie *Conteneurs* de la boîte à outils. Remarquez comme il adapte sa taille pour occuper automatiquement l'espace entre la barre de menus et celle d'état. Double-cliquez sur la propriété `IsSplitterFixed` de votre `SplitContainer` pour changer sa valeur à `True` afin d'éviter que l'utilisateur ne déplace le séparateur entre les deux parties de votre formulaire.

Panneau de gauche

À gauche, votre application affichera une liste de miniatures des images se trouvant dans un répertoire donné. Le meilleur moyen pour parvenir à ce résultat est d'utiliser un contrôle `ListView`. En effet, ce contrôle permet, comme son nom l'indique, d'afficher des listes et, en plus, d'associer une image à chaque élément de ladite liste.

Toutefois, les images utilisées ne se trouvent pas directement dans le contrôle `ListView`. Vous devez ajouter dans votre formulaire un contrôle `ImageList`, qui se trouve dans la section *Composants* de la boîte à outils. Configurez la propriété `ImageSize` du contrôle pour que les images stockées aient une largeur et une hauteur de 100 pixels.



◄ **Figure 2-8 :** Modifier la dimension des images dans une ImageList

Glissez maintenant, à partir de la section *Contrôles communs* de la boîte à outils, un contrôle `ListView` dans le panneau de gauche de votre `SplitContainer`.

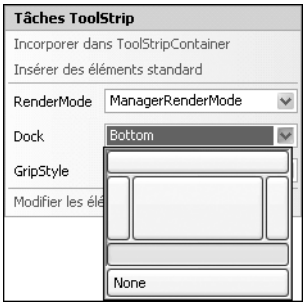
Pour configurer ce contrôle, cliquez sur la flèche de balise active et sur la commande **Ancrer dans le conteneur parent**. La liste occupera alors tout l'espace disponible dans le panneau. Vérifiez ensuite que le champ *Vue* contient bien la valeur `LargeIcon`. Modifiez la valeur si ce n'est pas le cas. Associez l'`ImageList` que vous venez de créer au contrôle en sélectionnant la valeur appropriée dans le champ *Grand ImageList*. Enfin, pensez à changer la valeur de la propriété `MultiSelect` afin qu'un seul élément du contrôle puisse être sélectionné à la fois.

◄ **Figure 2-9 :** Configuration du contrôle ListView

Panneau de droite

Ce panneau contiendra une barre d'outils qui permettra de naviguer dans la liste et de manipuler les images, ainsi qu'un contrôle `PictureBox` qui permettra d'afficher l'image sélectionnée.

Commencez par déposer un contrôle `ToolStrip` dans le `Panel2` de votre `SplitContainer`. En cliquant sur la flèche de balise active, configurez les propriétés `Dock` à `Bottom` et `GripStyle` à `Hidden`.



◄ **Figure 2-10 :** Positionner le contrôle ToolStrip

Ajoutez des boutons et des séparateurs à votre barre d'outils pour arriver au résultat suivant :

◄ **Figure 2-11 :** Barre d'outils de l'album de photos

Dans l'ordre, les boutons s'appellent : `PrecedentToolStripButton`, `SuivantToolStripButton`, `RetournerHToolStripButton`, `RetournerVToolStripButton`, `PivoterDToolStripButton`, `PivoterGToolStripButton`, `CopierToolStripButton`, `EnregistrerToolStripButton`, `EnregistrerSousToolStripButton`, `SupprimerToolStripButton`. Les noms et les images parlent d'eux-mêmes. Pensez à mettre la propriété `Enabled` de tous ces boutons à `False`, afin qu'ils ne soient pas accessibles avant le chargement des images.

Finalement, glissez un contrôle `PictureBox` dans l'espace inoccupé et, à l'aide de la flèche de balise active, ancrez-le au conteneur parent et sélectionnez le mode de redimensionnement `CenterImage`.

Vous êtes maintenant prêt à donner vie à votre interface.

2.3 Réalisation

Il y a trois régions dans l'application avec lesquelles l'utilisateur peut interagir : le menu **Fichier**, la `ListBox` qui affichera les images et, enfin, la barre d'outils. Vous allez écrire des gestionnaires d'événements pour les contrôles de chacune de ces régions.

Si vous lancez votre application telle qu'elle est et si la taille des panneaux du `SplitContainer` ne vous convient pas, fixez la largeur du panneau de gauche dans le gestionnaire d'événements `Load` de votre formulaire :

```
Private Sub Form1_Load(ByVal sender As System.Object, _
  ByVal e As System.EventArgs)
    SplitContainer1.SplitterDistance = 160
End Sub
```

▲ Définition de la taille des panneaux du SplitContainer

Ce détail réglé, vous pouvez commencer à programmer le menu **Fichier**.

Menu Fichier

Le code de la commande **Quitter** est trivial (un simple appel à la méthode `Close` du formulaire) et ne sera donc pas décrit en détail. En revanche, le gestionnaire d'événements `Click` de la commande `Ouvrir` est l'une des méthodes les plus importantes de l'application. Voici son implémentation :

```
Private Sub OuvrirToolStripMenuItem_Click(ByVal sender As _
  System.Object, ByVal e As System.EventArgs)
    FolderBrowserDialog1.SelectedPath = _
      My.Computer.FileSystem.SpecialDirectories.MyPictures
    If FolderBrowserDialog1.ShowDialog = DialogResult.OK _
      Then
        ImageList1.Images.Clear()
        ListView1.Items.Clear()

        For Each file As String In _
Directory.GetFiles(FolderBrowserDialog1.SelectedPath, _
                   "*.jp*g")
            Dim index As Integer = _
              ImageList1.Images.Add(ObtenirImage(file), _
                                    Nothing)
            Dim photo As ListViewItem = New _
ListViewItem(Path.GetFileNameWithoutExtension(file), index)
            photo.Tag = file
            ListView1.Items.Add(photo)
        Next

        For Each tsi As ToolStripItem In ToolStrip1.Items
```

```
            tsi.Enabled = (ListView1.Items.Count > 0)
        Next

        If (ListView1.Items.Count > 0) Then
            ListView1.Items(0).Selected = True
        Else
            PictureBox1.Image = Nothing
        End If
    End If
End Sub
```

▲ Peupler une ListBox avec les images contenues dans un répertoire

La plupart du code est auto-explicatif : on commence par configurer le `FolderBrowserDialog` pour que le chemin par défaut soit celui du répertoire *Mes Images* et ensuite on l'affiche. Si l'utilisateur valide un choix en utilisant le bouton OK, le travail commence.

Tout d'abord, on doit supprimer les éventuelles images susceptibles d'être stockées dans l'`ImageList` ainsi que les miniatures affichées dans la `ListView`. Ensuite on itère sur la liste de fichiers ayant les extensions *.jpg* ou *.jpeg* dans le répertoire sélectionné par l'utilisateur, que l'on récupère en utilisant la méthode `GetFiles` de la classe `Directory`.

Pour chaque image, on crée une miniature dans l'`ImageList`. L'image que l'on insère dans l'`ImageList` n'a pas été créée avec la méthode `FromFile` de la classe `Image` ou encore avec le constructeur de la classe `Bitmap`, mais avec une fonction appelée `ObtenirImage`. Retenez pour l'instant que cette fonction retourne un objet de type `Image`, les raisons de ce choix sont expliquées dans le prochain encadré.

Ensuite, un `ListViewItem` est créé auquel on associe le nom du fichier, sans extension, et le numéro d'image obtenu lorsque l'on a créé la miniature dans l'`ImageList`. La propriété `Tag` de ce `ListViewItem` contiendra le chemin complet vers l'image originale, qui sera utilisé à plusieurs reprises par la suite. Enfin, l'élément est ajouté dans le contrôle `ListView`.

Une fois les miniatures créées et affichées, il faut activer ou désactiver les boutons de la barre d'outils selon que images ont été trouvées ou non dans le répertoire fourni, ce que l'on peut aisément déterminer en vérifiant que le nombre d'éléments dans la `ListView` est supérieur à 0. Ensuite, il suffit d'itérer sur la collection `Items` de la barre et d'attribuer la valeur appropriée à la propriété `Enabled` de chaque bouton.

Finalement, s'il y a bien des éléments dans le contrôle `ListView`, le premier d'entre eux est sélectionné, ce qui affichera l'image correspondante dans le contrôle `PictureBox`. Dans le cas contraire, l'image de ce dernier est effacée.

Astuce

Travailler avec des images

Bien que la plateforme .NET prévoie maintes possibilités pour charger une image en mémoire, cela représente un danger potentiel : on ne peut pas garantir que les objets concernés auront libéré le fichier source lorsque l'on voudra récrire dessus, à moins de les détruire. La solution consiste à détruire l'objet qui aura chargé l'image le plus vite possible et à ne manipuler qu'une copie tout au long du programme. C'est justement ce que fait la fonction `ObtenirImage` :

```
Private Function ObtenirImage(ByRef chemin As String) _
  As Image
    Dim original As Bitmap = New Bitmap(chemin)
    Dim copie As Bitmap = New Bitmap(original.Width, _
                                     original.Height, _
                                     original.PixelFormat)
    Dim g As Graphics = Graphics.FromImage(copie)
    g.DrawImage(original, 0, 0)
    g.Dispose()
    original.Dispose()
    Return copie
End Function
```

▲ ObtenirImage charge une image en mémoire et en renvoie une copie

Afficher l'image sélectionnée

Vous affichez désormais les miniatures des images dans le contrôle `ListView` à gauche. Vous devez maintenant gérer l'événement `SelectedIndexChanged` de ce contrôle afin que l'image sélectionnée soit affichée dans le contrôle `PictureBox`.

```
Private Sub ListView_SelectedIndexChanged(ByVal sender _
  As System.Object, ByVal e As System.EventArgs)
    If ListView1.SelectedItems.Count > 0 Then
        PictureBox1.Image = _
ObtenirImage(ListView1.SelectedItems(0).Tag)
        If (PictureBox1.Image.Width > Width Or _
            PictureBox1.Image.Height > Height) Then
            PictureBox1.SizeMode = PictureBoxSizeMode.Zoom
        Else
            PictureBox1.SizeMode = _
            PictureBoxSizeMode.CenterImage
        End If
    End If
End Sub
```

▲ Afficher l'image sélectionnée dans le contrôle PictureBox

Après avoir vérifié qu'il y a bien une miniature sélectionnée, vous pouvez utiliser la fonction `ObtenirImage` pour charger la photo à partir du chemin stocké dans la propriété `Tag` du premier objet sélectionné du contrôle `ListView`. Si l'image est plus grande que le contrôle `PictureBox`, elle est affichée en mode Zoom, sinon elle est centrée dans le contrôle.

Barre d'outils

Étant donné la similitude de nombreux boutons de la barre d'outils, la gestion de leurs événements `Click` peut se faire avec seulement cinq gestionnaires d'événements simples.

Commencez par les boutons permettant la navigation :

```
Private Sub Navigation(ByVal sender As System.Object, _
  ByVal e As System.EventArgs)
    With ListView1
      If .Items.Count > 0 Then
        Dim SelectedIndex As Integer
        If .SelectedIndices.Count = 0 Then
          SelectedIndex = 0
        Else
          SelectedIndex = .SelectedIndices(0)
          If sender.Equals(PrecedentToolStripButton) _
                 And .SelectedIndices(0) > 0 Then
                  SelectedIndex = .SelectedIndices(0) - 1
          ElseIf sender.Equals(SuivantToolStripButton) _
          And .SelectedIndices(0) < .Items.Count - 1 Then
                  SelectedIndex = .SelectedIndices(0) + 1
          End If
        End If
          .Items(SelectedIndex).Selected = True
          .EnsureVisible(SelectedIndex)
      End If
    End With
End Sub
```

▲ Gestionnaire d'événements pour les boutons de navigation

Cette méthode récupère l'index de la miniature sélectionnée et, en fonction du bouton sélectionné et de la disponibilité des images, sélectionne l'index précédent ou suivant, ce qui provoquera l'affichage de la photo en grand format dans le contrôle `PictureBox`. La méthode `EnsureVisible` s'assure que l'élément sélectionné sera toujours visible dans le contrôle `ListView`.

Le gestionnaire des quatre boutons de transformation est un peu plus compliqué car il y a plus d'options à considérer :

```
Private Sub Transformations(ByVal sender As _
  System.Object, ByVal e As System.EventArgs)
    With PictureBox1
       If sender.Equals(RetournerHToolStripButton) Then
       .Image.RotateFlip(RotateFlipType.RotateNoneFlipX)
       ElseIf sender.Equals(RetournerVToolStripButton) Then
       .Image.RotateFlip(RotateFlipType.RotateNoneFlipY)
       ElseIf sender.Equals(PivoterDToolStripButton) Then
       .Image.RotateFlip(RotateFlipType.Rotate90FlipNone)
       ElseIf sender.Equals(PivoterGToolStripButton) Then
       .Image.RotateFlip(RotateFlipType.Rotate270FlipNone)
       End If

       If (.Image.Width > .Width Or _
           .Image.Height > .Height) Then
           .SizeMode = PictureBoxSizeMode.Zoom
       Else
           .SizeMode = PictureBoxSizeMode.CenterImage
       End If

       .Refresh()
   End With
End Sub
```

▲ Gestionnaire d'événements des boutons de transformation

On utilise la méthode `RotateFlip` avec une valeur de l'énumération `RotateFilpType` pour réaliser la transformation. Ensuite, on appelle la méthode `Refresh` sur le contrôle `PictureBox` afin de s'assurer que les transformations seront visibles dès la fin de l'exécution de la procédure.

Les gestionnaires des boutons **Copier** et **Supprimer** sont si simples qu'ils ne méritent pas que l'on s'attarde dessus. Souvenez-vous seulement que la propriété `Tag` de chaque `ListViewItem` contient le chemin complet du fichier image.

```
Private Sub Copier(ByVal sender As System.Object, _
  ByVal e As System.EventArgs)
    My.Computer.Clipboard.SetImage(PictureBox1.Image)
End Sub

Private Sub Supprimer(ByVal sender As System.Object, _
  ByVal e As System.EventArgs)
    File.Delete(ListView1.SelectedItems(0).Tag)
    ListView1.Items.Remove(ListView1.SelectedItems(0))
    PictureBox1.Image = Nothing
End Sub
```

▲ Gestionnaires d'événements des boutons Copier et Supprimer

Finalement, étant donné les précautions qui ont été prises lors du chargement des photos, le gestionnaire des boutons **Enregistrer** et **Enregistrer Sous** est lui aussi fort simple. Il suffit de déterminer, en fonction du bouton sur lequel l'utilisateur a cliqué, si l'image doit être enregistrée à son emplacement d'origine ou s'il est nécessaire d'afficher un `SaveFileDialog` afin de demander un nouvel emplacement. Ensuite, on utilise la méthode `Save` de l'`Image` du `PictureBox` pour enregistrer la photo et l'on met à jour le contrôle `ListView` avec une nouvelle miniature.

```
Private Sub Enregistrer(ByVal sender As Object, _
  ByVal e As System.EventArgs)
    Dim chemin As String = Nothing

    If sender.Equals(EnregistrerToolStripButton) Then
        chemin = ListView1.SelectedItems(0).Tag
    ElseIf (SaveFileDialog1.ShowDialog = _
            Windows.Forms.DialogResult.OK) Then
        chemin = SaveFileDialog1.FileName
    End If

    If Not String.IsNullOrEmpty(chemin) Then
        PictureBox1.Image.Save(chemin)
        ImageList1.Images.Item( _
        ListView1.SelectedItems(0).Index) = -
        New Bitmap(PictureBox1.Image, 100, 100)
        ListView1.Refresh()
    End If
End Sub
```

▲ Gestionnaire d'événements des boutons d'enregisterment

Les fonctionnalités de base étant implémentées, vous pouvez profiter de votre nouvel album de photos.

2.4 Check-list

Le développement de cette application vous a permis d'apprendre à :

- manipuler des fichiers au format JPEG avec la plateforme .NET ;
- éviter des erreurs lors de l'écriture de fichiers image chargés avec les classes `Bitmap` et `Image` ;
- réaliser des transformations simples sur des images, à l'aide de ces mêmes classes.

Envoi de messages chiffrés

Vous allez tirer parti, dans ce chapitre, des fonctionnalités de messagerie et de cryptographie que propose la plateforme .NET.

L'application que vous obtiendrez vous permettra de chiffrer vos messages importants avant de les envoyer par courrier électronique. Vous utiliserez pour cela un algorithme de chiffrage symétrique, dont le fonctionnement vous sera expliqué en détail.

3.1 Classes et espaces de noms utilisés

Comme pour toute application Windows, vous aurez besoin de l'espace de noms `System.Windows.Forms`. De plus, votre application fera appel à de nombreuses bibliothèques de classes de base, telles que `System.Net.Mail` pour l'envoi des messages et `System.Securty.Cryptography` pour le chiffrement.

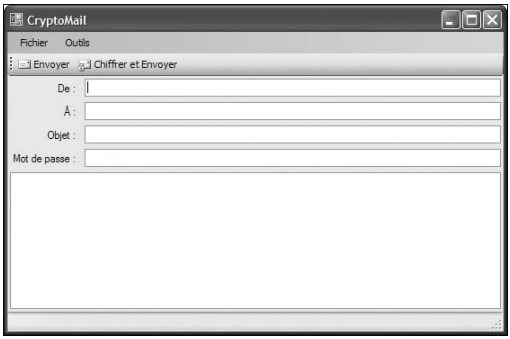
Enfin, vous aurez besoin de quelques classes auxiliaires qui se trouvent dans les espaces de noms `System.Text` et `System.IO`.

3.2 Interface utilisateur

Vous allez créer deux formulaires pour cette application : un premier, le principal, qui permettra aux utilisateurs de composer un message et de l'envoyer, chiffré, s'il le désire, à un destinataire, et un second, qui lui permettra de déchiffrer les messages qu'il reçoit.

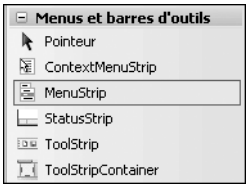
Formulaire principal

Lorsque vous créerez un nouveau projet de type *Application Windows*, Visual Basic 2005 Express préparera pour vous un formulaire qui sera affiché au démarrage de l'application. Vous allez ajouter des contrôles à ce formulaire pour permettre à un utilisateur d'envoyer des messages par courrier électronique.



▲ **Figure 3-1 :** Formulaire principal de l'application

Commencez par glisser un contrôle de type `MenuStrip`, un autre de type `ToolStrip` et un autre de type `StatusStrip`, que vous trouverez dans la section *Menus et barres d'outils* de la boîte à outils, dans le formulaire. Ces contrôles vont se placer automatiquement à leurs positions habituelles, c'est-à-dire, en haut du formulaire pour les deux premiers, et en bas pour le dernier.



◄ **Figure 3-2 :** Contrôles MenuStrip, StatusStrip et ToolStrip dans la boîte à outils

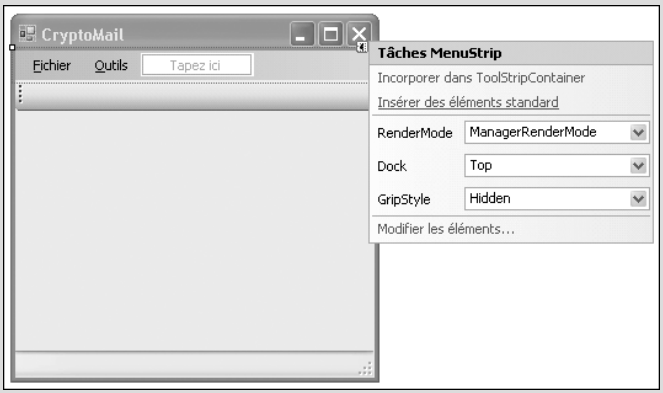
Dans cette application, le contrôle `StatusStrip` aura un rôle purement décoratif. En revanche, vous devrez ajouter des menus et des boutons au `MenuStrip` et au `ToolStrip` pour pouvoir les utiliser pour réaliser des actions.

Ajoutez maintenant les menus et les commandes du contrôle `MenuStrip`. Vous devez obtenir un menu **Fichier** avec les commandes **Nouveau** et **Quitter**, et un menu **Outils** avec la commande **Déchiffrer**.

Astuce

Création de menus

Vous pouvez utiliser la flèche de balise active qui apparaît à droite du contrôle `MenuStrip` lorsque celui-ci est sélectionné, pour insérer automatiquement les menus les plus utilisés.



▲ **Figure 3-3 :** Insertion d'éléments standard dans un MenuStrip à l'aide de la flèche de balise active

Astuce

Les menus ajoutés sont déjà configurés avec des images et des options d'accessibilité, ce qui vous fera gagner du temps. Vous êtes libre ensuite de modifier ou de supprimer les menus dont vous n'avez pas besoin.

Ajoutez deux boutons dans la barre d'outils grâce au bouton qui apparaît lorsque le contrôle est sélectionné. Renommez-les en modifiant leur propriété `(Name)` : le premier s'appellera `Envoyer` et le deuxième `ChiffrerEtEnvoyer`. Choisissez des images pour chacun des boutons en modifiant leur propriété `Image` et modifiez aussi leur propriété `Text` afin que le premier affiche `Envoyer` et l'autre `Chiffrer et Envoyer`. Pour que les boutons affichent l'image et le texte, vous devrez aussi modifier leur propriété `DisplayStyle` et lui attribuer la valeur `ImageAndText`.

◀ **Figure 3-4 :** Barre d'outils

Avant de vous occuper des autres composants de votre formulaire, déposez un contrôle de type `TableLayoutPanel` dans celui-ci. Il permet de placer les contrôles d'un formulaire de manière tabulaire, ce qui est approprié dans le cas de la présente application.

Pour configurer `TableLayoutPanel`, cliquez sur le lien *Modifiez les lignes et les colonnes* du menu de la balise active. La fenêtre **Styles de ligne et de colonne** s'ouvre. Dans la liste déroulante *Afficher*, sélectionnez *Lignes* et utilisez le bouton **Ajouter** pour insérer trois nouvelles lignes afin d'en avoir cinq au total. Enfin, dans le volet des propriétés, attribuez la valeur `Fill` à la propriété `Dock` du contrôle `TableLayoutPanel` afin que celui-ci occupe tout l'espace disponible entre les barres d'outils et d'état. Ne vous souciez pas de la taille des cellules pour l'instant, vous y reviendrez dès que les autres composants seront en place.

Glissez maintenant quatre contrôles `Label` dans la première cellule des quatre premières lignes, et quatre contrôles `TextBox` à côté de celles-ci, ainsi qu'un cinquième dans la première colonne de la dernière ligne.

Modifiez les propriétés des contrôles que vous venez d'ajouter comme suit :

Propriétés des contrôles du formulaire principal			
Contrôles Label		**Contrôles TextBox**	
`Text`	`Dock`	`(Name)`	`Dock`
`De :`	`Right`	`DeTextBox`	`Fill`
`À :`		`ATextBox`	
`Objet :`		`ObjetTextBox`	

Propriétés des contrôles du formulaire principal			
Contrôles Label		Contrôles TextBox	
Mot de passe :		MdpTextBox	
		MessageTextBox	

Sélectionnez le contrôle `MessageTextBox` et modifiez ses propriétés `Multiline` à `True` et `ColumnSpan` à 2, pour pouvoir écrire plusieurs lignes et pour que la zone de texte s'étende sur les deux colonnes de la ligne. Modifiez aussi la propriété `UseSystemPasswordChar` du contrôle `MdpTextBox` en lui attribuant la valeur `True`, pour que les caractères du mot de passe soient remplacés par un symbole spécial lors de l'affichage.

Pour terminer, ouvrez la fenêtre **Styles de ligne et de colonne** du contrôle `TableLayoutPanel` pour changer la taille des lignes et des colonnes. Sélectionnez l'option *Redimensionner automatiquement* dans la rubrique *Type de taille* pour toutes les lignes et toutes les colonnes. Si vous faites cela lorsque vous ajoutez les lignes supplémentaires à votre contrôle, vous obtenez des lignes et des colonnes trop petites pour y placer les différents éléments.

Maintenant que le formulaire principal est terminé, il faut créer un deuxième formulaire qui vous permettra de déchiffrer vos messages.

Formulaire de déchiffrement

Pour ajouter un formulaire à votre projet, cliquez du bouton droit sur le nom de celui-ci et sélectionnez **Formulaire Windows** dans le sous-menu **Ajouter**. Appelez votre nouveau formulaire `Dechiffrer`, en écrivant cette valeur dans le champ de texte *Nom* et en cliquant sur OK.

Vous allez utiliser un nouveau contrôle `TableLayoutPanel` dans ce formulaire, qui aura trois lignes et trois colonnes. En plus, vous affecterez à sa propriété `Dock` la valeur `Fill` pour que la grille comprenne tout le formulaire.

Ajoutez maintenant des contrôles `Label`, `TextBox` et `Button` dans la première ligne, et deux `TextBox` dans la deuxième et la troisième ligne. Configurez-les en vous basant sur le tableau suivant :

Propriétés des contrôles du formulaire de déchiffrement				
	(Name)	Text	Dock	Multiline
Label		Mot de passe :	Fill	
TextBox	MdpTextBox		Fill	
Button		Déchiffrer		

Propriétés des contrôles du formulaire de déchiffrement				
	(Name)	Text	Dock	Multiline
TextBox	MessageChiffre		Fill	True
TextBox	MessageClair		Fill	True

En plus, configurez le premier contrôle `TextBox` pour que les caractères du mot de passe soient remplacés lors de l'affichage, et les deux contrôles `TextBox` suivants pour qu'ils s'étendent sur trois colonnes.

Finalement, dans la fenêtre **Styles de ligne et de colonne** du contrôle `TableLayoutPanel`, spécifiez le redimensionnement automatique pour la première et la troisième colonne, et une largeur de 100 % pour la deuxième. De la même manière, spécifiez le redimensionnement automatique pour la première ligne, et des valeurs de 50 % pour les deux lignes suivantes.

◀ **Figure 3-5 :** Fenêtre de déchiffrement terminée

Les deux fenêtres de votre interface graphique sont désormais terminées. Vous pourrez passer à la réalisation de votre application.

3.3 Réalisation

Votre application a, principalement, deux parties fonctionnelles : l'envoi de messages par courrier électronique et le chiffrement et déchiffrement de ces messages.

Vous allez commencer par gérer le chiffrement, qui requiert la création d'une classe auxiliaire.

Classe auxiliaire pour la cryptologie

Pour accéder aux fonctions de cryptologie depuis les deux fenêtres de votre application, sans avoir à récrire plusieurs fois le même code, vous allez les encapsuler dans une classe auxiliaire.

Ajoutez une nouvelle classe dans votre projet en cliquant du bouton droit sur le nom de votre projet et en sélectionnant **Classe** dans le sous-menu **Ajouter**. Appelez votre classe `Crypto`.

Votre classe contiendra trois constantes, qui serviront à configurer le chiffrement et le déchiffrement, ainsi que deux méthodes statiques, qui géreront les opérations proprement dites.

```
Const TailleSalt As Integer = 16
Const TailleCle As Integer = 32
Const TailleIV As Integer = 16
```

▲ Constantes servant à configurer le fonctionnement de la classe Crypto

Ces trois constantes servent à définir les tailles, en octets, de certains éléments nécessaires aux opérations de chiffrement et de déchiffrement :

- Le salt est un jeu aléatoire d'octets utilisé pour rendre plus difficile le décryptage du mot de passe employé pour la création de la clé de chiffrement.
- La clé de chiffrement sera celle utilisée pour encoder le message. Elle n'est pas égale au mot de passe, car elle doit avoir une longueur précise, mais elle est générée à partir de celui-ci. Le fait d'utiliser un salt pour générer la clé fait que celle-ci est différente chaque fois qu'elle est générée. Il faudra communiquer le salt pour permettre la génération de la bonne clé lors du déchiffrement.
- Le vecteur d'initialisation (IV) est un jeu aléatoire d'octets dont le but est de rendre plus difficile le décryptage d'un message chiffré. En effet, l'utilisation du IV empêche un bloc de texte brut donné, lorsqu'il apparaît à plusieurs reprises dans le message à chiffrer, d'être encodé de la même manière. Cela rend plus difficile l'analyse du texte chiffré.

Vient ensuite la méthode de chiffrement :

```
Public Shared Function Chiffrer( _
  ByRef messageClair As String, ByRef mdp As String) _
  As String
   Dim octetsClair() As Byte = _
     Encoding.Unicode.GetBytes(messageClair)
   Dim octetsMdp As New Rfc2898DeriveBytes(mdp, TailleSalt)

   Dim ms As New MemoryStream
```

```
    Dim algo As New RijndaelManaged
    algo.Key = octetsMdp.GetBytes(TailleCle)
    algo.IV = octetsMdp.GetBytes(TailleIV)

    Dim cs As New CryptoStream(ms, algo.CreateEncryptor(), _
      CryptoStreamMode.Write)
    cs.Write(octetsClair, 0, octetsClair.Length)
     cs.Close()

    Dim messageChiffre() As Byte = ms.ToArray()
    Dim resultat(octetsMdp.Salt.Length + _
      messageChiffre.Length - 1) As Byte

    Array.ConstrainedCopy(octetsMdp.Salt, 0, _
      resultat, 0, octetsMdp.Salt.Length)
    Array.ConstrainedCopy(messageChiffre, 0, resultat, _
      octetsMdp.Salt.Length, messageChiffre.Length)

    Return Convert.ToBase64String(resultat)
End Function
```

▲ Méthode de chiffrement

La méthode `Chiffrer` prend en paramètre une chaîne de caractères et le mot de passe servant à la chiffrer.

On utilise l'algorithme Rijndael pour chiffrer votre message. Mais tout d'abord, on transforme le message à encoder en un tableau d'octets à l'aide de la classe `Encoding`, tandis que la classe `Rfc2898DeriveBytes` sert à générer un jeu d'octets à partir duquel on extraira les octets correspondant à la clé de chiffrement et au IV. Le constructeur de cette classe génère en plus un salt de la taille indiquée qu'il faudra concaténer au message chiffré par la suite.

Les opérations de chiffrement et de déchiffrement sont réalisées non pas avec des données stockées dans des variables, mais avec des flux d'octets. Cette particularité permet d'encoder de grandes quantités de données et de rediriger les octets chiffrés vers un fichier ou vers le réseau de manière efficace. Mais elle implique aussi de créer un tel flux. Étant donné que l'on souhaite récupérer le message chiffré en mémoire pour constituer le message électronique, on doit créer un flux en mémoire avec la classe `MemoryStream`.

On doit ensuite créer un objet représentant l'algorithme de chiffrement choisi. On utilise la version managée de l'algorithme Rijndael, représenté par la classe `RijndaelManaged`. Rijndael est un algorithme de chiffrage symétrique, ce qui veut dire que la même clé est utilisée pour le chiffrement et pour le déchiffrement des données. Comme on génère la clé de chiffrement de manière pseudo-aléatoire en utilisant le mot de passe fourni et un salt, on doit fournir ce dernier avec le message chiffré pour générer de nouveau la clé et déchiffrer le message.

On configure la clé et le vecteur d'initialisation de l'objet `RijndaelManaged` en extrayant des octets de l'objet `Rfc2898DeriveBytes` avec sa méthode `GetBytes` et en les stockant dans les propriétés `Key` et `IV`. Il est désormais possible de chiffrer un message.

En ce sens, on a besoin d'un flux spécial, le `CryptoStream`, que l'on crée en spécifiant le flux qui recevra le message chiffré, un objet chiffreur créé avec la méthode `CreateEncryptor` de l'algorithme et la valeur `CryptoStreamMode.Write` pour spécifier que l'on souhaite écrire dans le premier paramètre. Il ne reste plus qu'à écrire le flux d'octets que l'on souhaite chiffrer dans le flux de chiffrement à l'aide de sa méthode `Write` et de le fermer avec sa méthode `Close`, une fois l'opération terminée.

Finalement, on récupère le message chiffré, qui se trouve dans l'objet `MemoryStream`, dans un tableau d'octets et l'on prépare un autre tableau d'octets pour y stocker le salt et le message chiffré. Comme le salt et le message se trouvent déjà sous forme de tableaux, on peut utiliser la méthode statique `ConstrainedCopy` pour copier ces deux valeurs dans le tableau `resultat`.

On peut retourner le résultat du chiffrement sous forme de chaîne de caractères après avoir converti le tableau `resultat` en utilisant la méthode `ToBase64String` de la classe `Convert`. Cette conversion assure que, quels que soient les octets résultant du chiffrement, ils seront tous représentables à l'écran, dans un fichier texte ou encore dans un courrier électronique.

La procédure de déchiffrement fonctionne à l'inverse du processus que vous venez d'implémenter.

```
Public Shared Function Dechiffrer( _
  ByRef messageChiffre As String, ByRef mdp As String) _
  As String
   Dim message() As Byte = _
     Convert.FromBase64String(messageChiffre)
   Dim salt(TailleSalt - 1) As Byte
   Dim octetsChiffres( _
     message.Length - TailleSalt - 1) As Byte
   Array.ConstrainedCopy(message, 0, salt, 0, TailleSalt)
   Array.ConstrainedCopy(message, TailleSalt, _
     octetsChiffres, 0, octetsChiffres.Length)

   Dim octetsMdp As New Rfc2898DeriveBytes(mdp, TailleSalt)
   octetsMdp.Salt = salt
   Dim ms As New MemoryStream
   Dim algo As New RijndaelManaged

   algo.Key = octetsMdp.GetBytes(TailleCle)
   algo.IV = octetsMdp.GetBytes(TailleIV)
```

```
    Dim cs As New CryptoStream(ms, algo.CreateDecryptor(), _
      CryptoStreamMode.Write)
    cs.Write(octetsChiffres, 0, octetsChiffres.Length)
    cs.Close()

    Return Encoding.Unicode.GetString(ms.ToArray())
End Function
```

▲ Méthode de déchiffrement

On récupère un tableau d'octets à partir d'une chaîne de caractères encodée en base 64 et on la sépare en deux autres tableaux d'octets pour récupérer le salt et le message chiffré.

On recrée ensuite la clé de chiffrage comme on l'a fait pour le chiffrement, et l'on prépare un objet `RijndaelManaged` et un flux `MemoryStream` pour les résultats, et un flux `CryptoStream` pour l'opération de déchiffrement. Cependant, lors de la création de ce dernier, il faut passer un déchiffreur en tant que second paramètre grâce à la méthode `CreateDecryptor` de l'algorithme.

Une fois le déchiffrement effectué, on peut récupérer une chaîne de caractères grâce à la classe `Encoding`.

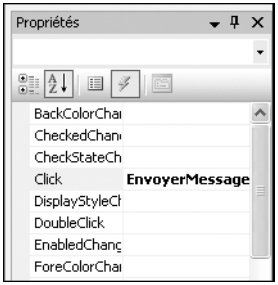
La classe auxiliaire est terminée.

Envoyer des messages chiffrés

Dans la fenêtre principale de votre application, vous avez ajouté deux boutons qui permettent d'envoyer des courriers électroniques chiffrés ou bruts.

Vous allez associer un gestionnaire d'événements à ces deux boutons pour permettre à l'utilisateur d'envoyer par courrier électronique le message qu'il a saisi, éventuellement chiffré avec le mot de passe s'il le désire.

Sélectionnez les deux boutons en cliquant dessus tout en appuyant sur la touche Ctrl. Dans le volet des propriétés, cliquez sur le bouton en forme d'éclair pour afficher les événements des deux boutons.



◄ **Figure 3-6** : Création d'un gestionnaire d'événements Click

Saisissez EnvoyerMessage comme nom du gestionnaire d'événements Click et appuyez sur la touche Entrée pour créer la méthode.

```
Private Sub EnvoyerMessage(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles Envoyer.Click, CrypterEnvoyer.Click
    Dim smtp As New SmtpClient("smtp.monserveur.com")
    AddHandler smtp.SendCompleted, _
      AddressOf Envoyer_SendCompleted

    Dim message As New MailMessage()
    message.From = New MailAddress(DeTextBox.Text)
    message.To.Add(New MailAddress(ATextBox.Text))
    message.Subject = ObjetTextBox.Text

    If sender.Equals(Envoyer) Then
        message.Body = MessageTextBox.Text
    Else
        message.Body = Crypto.Chiffrer( _
          MessageTextBox.Text, MdpTextBox.Text)
    End If

    smtp.SendAsync(message, message.Subject)
End Sub
```

▲ Préparer et envoyer un message électronique

On crée un client SMTP à l'aide de la classe SmtpClient. Le constructeur de cette classe a besoin de l'adresse du serveur SMTP pour envoyer des messages. On configure un gestionnaire d'événements SendCompleted de l'objet créé. Il permettra de savoir si le message a été envoyé avec succès.

On crée ensuite un nouvel objet de type MailMessage, qui représente le message électronique. On configure cet objet à l'aide des valeurs saisies par l'utilisateur dans le formulaire.

Il ne reste plus qu'à décider, en fonction du bouton sur lequel l'utilisateur a cliqué, si le corps du message doit être chiffré à l'aide de la classe auxiliaire Crypto, et à l'envoyer en utilisant la méthode SendAsync du client SMTP. Le premier paramètre fournit au client toutes les informations nécessaires pour délivrer le message, tandis que le deuxième n'est là que pour des raisons purement informatives. Il pourra être utilisé dans le gestionnaire d'événements SendCompleted.

```
Private Sub Envoyer_SendCompleted( -
  ByVal sender As System.Object, _
  ByVal e As System.ComponentModel.AsyncCompletedEventArgs)

    If e.Error IsNot Nothing Then
```

```
        Dim messageErreur As New StringBuilder()
        messageErreur.AppendFormat("Erreur lors de l'envoi _
          du message [{0}]. ", e.UserState.ToString())
        messageErreur.AppendLine(e.Error.Message)

        If (e.Error.InnerException IsNot Nothing) Then
            messageErreur.AppendFormat("Message : {0}", _
              e.Error.InnerException.Message)
        End If

        MessageBox.Show(messageErreur.ToString(), _
          "Erreur", MessageBoxButtons.OK, _
          MessageBoxIcon.Error)
    Else
        MessageBox.Show("Message envoyé !", "Succès", _
          MessageBoxButtons.OK, MessageBoxIcon.Information)
    End If
End Sub
```

▲ Gestionnaire d'événements SendCompleted

Dans ce gestionnaire, on vérifie simplement la présence d'une erreur lors de l'envoi du message et l'on affiche des détails sur celle-ci. S'il n'y a pas d'erreur, on affiche un message informant du succès de l'opération.

Déchiffrer les messages

Le deuxième formulaire permettra de déchiffrer les messages que l'on vous aura envoyés en utilisant votre application. Le code que vous devez écrire pour cela est simple.

Ouvrez le formulaire *Dechiffrer* et double-cliquez sur le bouton disponible. Complétez la méthode créée comme suit :

```
Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
Try
TexteClair.Text =
Crypto.Dechiffrer(TexteChiffre.Text, MotDePasse.Text)
Catch ex As Exception
Msgbox("Mot de passe incorrect ou message corrompu !",
"Erreur", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
End Sub
```

▲ Déchiffrer un message

On utilise la méthode `Dechiffrer` de la classe auxiliaire pour déchiffrer le texte qui se trouve dans le premier champ de texte. Ce code doit se trouver dans un bloc `Try… Catch`, car si le mot de passe fourni est incorrect ou si le texte chiffré

a été altéré, une exception sera levée. Il faut gérer cette exception pour informer l'utilisateur de l'échec.

Touches finales

Les deux formulaires sont désormais fonctionnels. Or vous ne pouvez pas accéder au deuxième à partir du formulaire principal. Toutefois, lors de la préparation de ce dernier, vous avez prévu des commandes dans les différents menus, qui permettent, entre autres, cet accès. Il ne reste plus qu'à implémenter leurs gestionnaires d'événements en double-cliquant sur chacun des boutons et en écrivant leur code comme suit :

```
' Menu Fichier
Private Sub NouveauToolStripMenuItem_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles NouveauToolStripMenuItem.Click
    DeTextBox.Text = String.Empty
    ATextBox.Text = String.Empty
    ObjetTextBox.Text = String.Empty
    MdpTextBox.Text = String.Empty
    MessageTextBox.Text = String.Empty
End Sub

Private Sub QuitterToolStripMenuItem_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles QuitterToolStripMenuItem.Click
    Close()
End Sub

' Menu Outils
Private Sub DechiffrerToolStripMenuItem_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles PersonnaliserToolStripMenuItem.Click
    Dim d As New Dechiffrer()
    d.Show()
End Sub
```

▲ Gestionnaires d'événements des différentes commandes des menus du formulaire principal

La commande **Nouveau** du menu **Fichier** servira donc à vider tous les champs du formulaire principal et la commande **Quitter** du même menu servira, logiquement, à le fermer. La commande **Déchiffrer** du menu **Outils** créera un nouveau formulaire de déchiffrement et l'affichera.

Grâce à ces touches finales, votre application est maintenant terminée.

3.4 Check-list

En développant cette application, vous avez appris à :

- utiliser les classes de l'espace de noms `System.Net.Mail` pour envoyer des courriers électroniques ;
- utiliser les classes et les méthodes de conversion de la plateforme .NET pour transformer et encoder des données de différentes manières ;
- utiliser les algorithmes de cryptologie proposés par la plateforme, en particulier celui de Rijndael, pour sécuriser l'échange de messages électroniques ;
- encapsuler des fonctionnalités dans une classe auxiliaire pour les rendre accessibles à d'autres classes de l'application.

Gestion d'un concours

Le but de l'application que vous allez développer dans ce chapitre est de permettre de gérer des données sans faire appel à un système de gestion de bases de données (SGBD).

Grâce aux Assistants et aux contrôles inclus dans Visual Basic 2005 Express, vous serez capable de connecter vos objets directement à votre interface graphique. Vous pourrez ensuite enregistrer vos données en stockant l'état de tous vos objets dans un fichier.

La transformation des objets en un format qui peut être stocké ou transporté s'appelle "sérialisation". Vous allez utiliser les classes fournies par la plate-forme .NET qui vous permettront de stocker vos objets sous forme binaire. Les classes pour le processus inverse, appelé "désérialisation", se trouvent aussi dans les bibliothèques .NET.

Votre application servira à la gestion d'un concours de pêche. Elle devra gérer l'inscription des participants, la saisie des résultats et l'affichage des classements.

Pour chaque participant, les résultats à enregistrer sont :

- la masse totale de poissons péchés ;
- la masse du plus gros poisson ;
- le nombre de poissons.

Les résultats pourront être classés soit par genre (hommes et femmes), soit par âge, avec une catégorie "junior" pour les moins de 17 ans.

4.1 Classes et espaces de noms utilisés

Comme pour toute application Windows, vous allez utiliser des classes de l'espace de noms `System.Windows.Forms`. Vous aurez aussi besoin de classes dans les espaces de noms `System.IO` pour la manipulation de fichiers et de la classe `BinaryFormatter` de l'espace de noms `System.Runtime.Serialization.Formatters.Binary`, qui se chargera de la sérialisation.

Toutefois, ces classes ne vous seront pas d'une grande utilité si vous ne créez pas les classes qui représenteront les objets du domaine de votre application.

Créez un nouveau projet de type *Application Windows* pour commencer à programmer.

La classe Personne

Chaque personne inscrite au concours sera caractérisée par un certain nombre d'attributs communs. En particulier, elles auront toutes un nom, un prénom, une

date de naissance et un genre. Vous allez encapsuler ces informations dans la classe `Personne`.

Cliquez du bouton droit sur le nom de votre projet et ajoutez la classe `Personne` à l'aide de la commande **Classe** du sous-menu **Ajouter**.

```
<Serializable()> _
Public Class Personne
    Implements IComparable(Of Personne)

    Public m_nom As String
    Public m_prenom As String
    Public m_dateN As DateTime
    Public m_feminin As Boolean

    Public Sub New(ByVal nom As String, _
      ByVal prenom As String, _
      ByVal dateNaissance As DateTime, _
      ByVal feminin As Boolean)
        Me.m_nom = nom
        Me.m_prenom = prenom
        Me.m_dateN = dateNaissance
        Me.m_feminin = feminin
    End Sub

    Public Property Nom() As String
        Get
            Return m_nom
        End Get
        Set(ByVal value As String)
            m_nom = value
        End Set
    End Property

    Public Property Prenom() As String
        Get
            Return m_prenom
        End Get
        Set(ByVal value As String)
            m_prenom = value
        End Set
    End Property

    Public Property DateDeNaissance() As DateTime
        Get
            Return m_dateN
        End Get
        Set(ByVal value As DateTime)
            m_dateN = value
        End Set
```

```
    End Property

    Public Property Feminin() As Boolean
        Get
            Return m_feminin
        End Get
        Set(ByVal value As Boolean)
            m_feminin = value
        End Set
    End Property

    Public ReadOnly Property NomComplet() As String
        Get
            Return Me.m_nom.ToUpper() + " " + Me.m_prenom
        End Get
    End Property

    Public ReadOnly Property Age() As Integer
        Get
            Return CInt( _
              Today.Subtract(Me.m_dateN).TotalDays / _
                                              365.25)
        End Get
    End Property

    Public Function CompareTo(ByVal obj As Personne) _
      As Integer _
      Implements IComparable(Of Personne).CompareTo
        Return Me.NomComplet.CompareTo(obj.NomComplet)
    End Function
End Class
```

▲ La classe Personne

L'attribut `Serializable`, placé au début de la déclaration de la classe, est nécessaire car on va sérialiser les données des personnes participant au concours.

On ajoute un constructeur à la classe, qui permet d'initialiser tous ses champs lors de l'instanciation des objets de type `Personne`.

On crée aussi des propriétés pour chacun des champs de la classe. Elles permettront d'utiliser automatiquement les objets comme source de données dans Visual Basic 2005 Express.

On crée deux propriétés supplémentaires en lecture seule : `NomComplet` et `Age`. La première retourne une chaîne de caractères composée du nom de famille de la personne en majuscules, suivi de son prénom ; cette propriété est utile pour l'affichage des informations relatives aux participants. La deuxième propriété

calcule l'âge des individus et est utilisée pour déterminer les membres de la catégorie "junior".

Enfin, on implémente une méthode `CompareTo`, puisque l'on va implémenter l'interface `IComparable(Of Personne)`. Parce que l'on utilise une interface générique, on n'a pas à se soucier du type du paramètre passé à la méthode : on a défini dès la compilation qu'il s'agit d'une `Personne`. Cette méthode sert à trier par ordre alphabétique une liste d'instances de la classe `Personne`.

La classe Participant

Vous avez déjà encapsulé certaines données relatives aux personnes participant au concours dans la classe `Personne`. Mais cela ne suffit pas : il faut associer à chaque participant ses résultats. Vous allez donc créer une classe `Participant`, qui dérivera de la classe `Personne` et contiendra les résultats en plus des données personnelles de chacun.

```
<Serializable()> _
Public Class Participant
    Inherits Personne

    Private m_nbPoissons As Integer
    Private m_masseTotalePoisson As Single
    Private m_massePlusGrosPoisson As Single

    Public Sub New(ByVal nom As String, _
      ByVal prenom As String, _
      ByVal dateNaissance As DateTime, _
      ByVal feminin As Boolean)
        MyBase.New(nom, prenom, dateNaissance, feminin)
        Me.NombrePoissons = 0
        Me.MassePlusGrosPoisson = 0
        Me.MasseTotalePoisson = 0
    End Sub

    Public Property NombrePoissons() As Integer
        Get
            Return Me.m_nbPoissons
        End Get
        Set(ByVal value As Integer)
            Me.m_nbPoissons = value
        End Set
    End Property

    Public Property MasseTotalePoisson() As Single
        Get
            Return Me.m_masseTotalePoisson
        End Get
```

```
        Set(ByVal value As Single)
            Me.m_masseTotalePoisson = value
        End Set
    End Property

    Public Property MassePlusGrosPoisson() As Single
        Get
            Return Me.m_massePlusGrosPoisson
        End Get
        Set(ByVal value As Single)
            Me.m_massePlusGrosPoisson = value
        End Set
    End Property
End Class
```

▲ La classe Participant

Tout comme la classe `Personne`, la classe `Participant` doit avoir l'attribut `Serializable` pour qu'il soit possible de stocker son état dans un fichier par la suite. Bien que cette classe n'implémente pas l'interface `IComparable`, ses instances peuvent être comparées entre elles grâce à l'implémentation de sa classe de base.

Ici on crée un constructeur qui permettra d'initialiser les données personnelles de chaque participant en appelant le constructeur de la classe de base à l'aide de `MyBase.New`.

On doit, une fois de plus, créer des propriétés pour accéder aux champs de la classe, afin que celle-ci puisse être utilisée comme source de données par la suite.

4.2 Interface utilisateur

Maintenant que vous avez créé les classes représentant les données que manipulera votre application, il ne vous reste plus qu'à créer l'interface graphique de celle-ci.

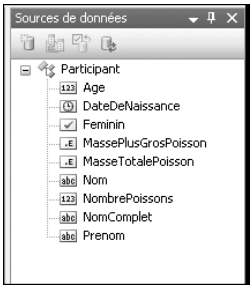
Les nouveaux Assistants et contrôles présents dans Visual Basic 2005 Express vous faciliteront la tâche.

Définition de la source de données

Commencez par générer la solution à l'aide de la commande **Générer** du menu de même nom. Cela a pour effet de compiler les types que vous avez créés précédemment et de les rendre visibles aux Assistants de Visual Basic 2005 Express.

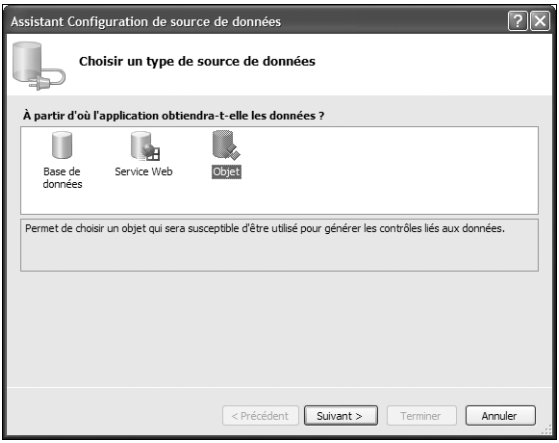
Affichez maintenant le volet Sources de données en cliquant sur la commande **Afficher les sources de données** dans le menu **Données**. Aucune source de données n'est associée au projet pour l'instant. Cliquez sur le lien *Ajouter une nouvelle source de données* pour en créer une.

◄ **Figure 4-1 :** Le volet Sources de données est vide



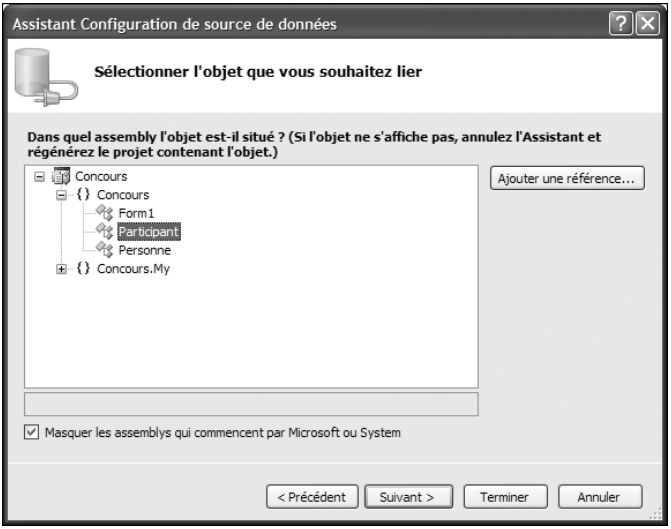
◄ **Figure 4-2 :** La source de données Participant

Dans l'Assistant qui se lance, sélectionnez le choix *Objet* et cliquez sur **Suivant**.



▲ **Figure 4-3 :** Assistant de Configuration de source de données

Maintenant, sélectionnez le type d'objet que vous souhaitez utiliser en tant que source de données. Développez l'arborescence pour trouver le type `Participant` et cliquez sur **Terminer**.

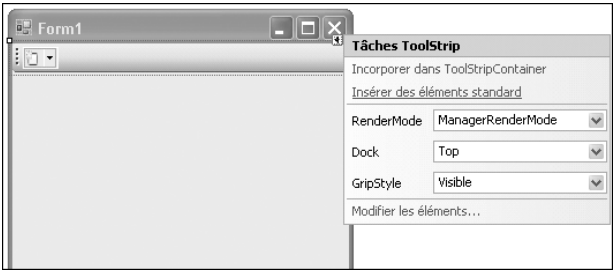


▲ **Figure 4-4 :** Choix de l'objet à lier

Maintenant que vous avez défini votre source de données, la création de votre formulaire sera presque un jeu d'enfants.

Création du formulaire

Commencez par déposer un contrôle de type `ToolStrip` dans votre formulaire et utilisez la flèche de balise active de celui-ci pour insérer des éléments standard.



▲ **Figure 4-5 :** Insérer des éléments standard dans la barre d'outils

Vous allez supprimer tous les éléments, sauf les boutons **Ouvrir** et **Enregistrer**.

Astuce

Création de barres d'outils et de menus

Visual Basic 2005 Express propose un certain nombre d'éléments standard, aussi bien pour les `ToolStrip` que pour les `MenuStrip` que vous pouvez insérer à l'aide des flèches de balise active des contrôles.

Vous êtes libre de créer vos barres de menus et d'outils manuellement si vous le souhaitez. Mais les éléments standard insérés par Visual Basic 2005 Express sont, pour la plupart, déjà configurés, avec les images et raccourcis clavier. Il ne vous reste plus qu'à supprimer ceux dont vous n'avez pas besoin.

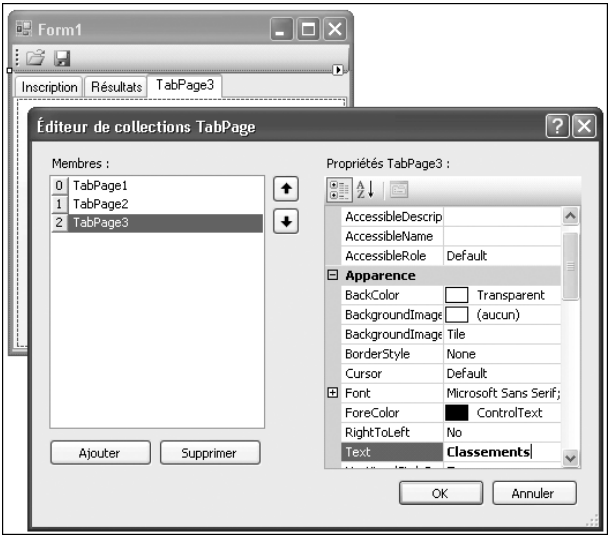
Vous allez maintenant diviser votre application en trois parties fonctionnelles : la gestion des inscriptions, l'enregistrement des résultats et l'affichage des classements. Pour cela, insérez un contrôle `TabControl` dans votre formulaire et réglez sa propriété `Dock` sur `Fill`.

Par défaut, le contrôle sera créé avec deux onglets. Ajoutez le troisième en cliquant sur la flèche de balise active et en choisissant **Ajouter un onglet**.

◀ **Figure 4-6 :** Ajouter un nouvel onglet

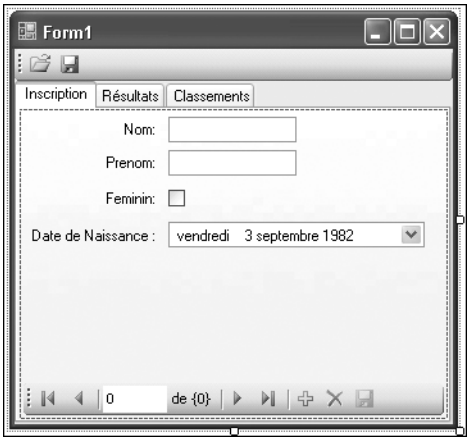
Utilisez la propriété `TabPages` du `TabControl` pour afficher l'Éditeur de collections TabPage. Servez-vous de l'Éditeur pour saisir les valeurs `Inscription`, `Résultats` et `Classements` dans la propriété `Text` de chaque onglet (voir Figure 4-7).

Maintenant, ajoutez les champs qui vous permettront de réaliser les inscriptions. Pour cela, glissez les champs `Nom`, `Prenom`, `Feminin` et `DateDeNaissance` à partir du volet Sources de données vers votre formulaire. Pour chacun d'entre eux, un `Label` et un contrôle permettant la saisie des données sont ajoutés au formulaire.



▲ **Figure 4-7 :** Éditeur de collections TabPage

Les contrôles `ParticipantBindingSource` et `ParticipantBindingNavigator` sont apparus dans la zone des contrôles non affichables. Il s'agit de deux contrôles qui permettront l'affichage et la navigation d'un jeu de données. Sélectionnez le contrôle `ParticipantBindingNavigator`. Coupez-le pour le retirer du haut du formulaire et collez-le sous l'onglet **Inscription**. Il viendra se placer vers le haut de celui-ci. Changez sa propriété `Dock` à `Bottom` afin qu'il se place en bas de l'onglet.



◄ **Figure 4-8 :** Onglet Inscription

Sélectionnez maintenant le deuxième onglet. Vous allez y ajouter un contrôle `ComboBox`, qui servira à sélectionner le participant pour saisir ses résultats. Pour lier ce contrôle à la source de données, utilisez la flèche de balise active et cochez la case *Utilisez des éléments liés aux données*. Sélectionnez `ParticipantBindingSource` comme *Source de données* et choisissez d'afficher le membre `NomComplet`.

▲ **Figure 4-9 :** Liaison d'un contrôle ComboBox à une source de données

Vous pourriez ajouter les champs correspondant aux résultats de chaque participant, comme vous l'avez fait sous l'onglet précédent. Mais si vous procédez ainsi, les champs insérés seront de type `TextBox` et personne ne peut garantir que les valeurs saisies seront des nombres, comme le requiert le type `Participant`.

Cliquez donc sur un champ tel que `NombrePoissons` dans le volet Sources de données et sélectionnez la commande **Personnaliser** du menu qui s'affiche lorsque vous cliquez sur le bouton fléché situé à côté du champ. Vérifiez, dans la fenêtre qui s'ouvre, que le contrôle de type `NumericUpDown` est coché. Faites cette opération pour les trois champs qui concernent les résultats.

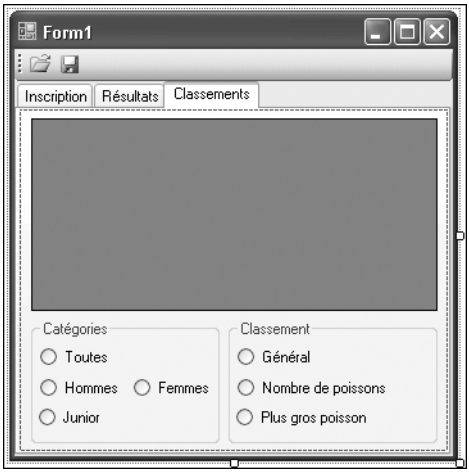
Désormais, vous pouvez utiliser le bouton fléché qui se trouve à côté de chaque champ dans le volet Sources de données pour sélectionner le type `NumericUpDown` à affecter aux champs de résultats. Glissez les trois champs qui concernent les résultats des participants.

◄ **Figure 4-10 :**
Onglet Résultats

Modifiez la propriété `DecimalPlaces` des champs relatifs aux masses des poissons pour pouvoir insérer des chiffres après la virgule.

Enfin, sous le troisième onglet, ajoutez un contrôle `DataGridView`. Décochez les options d'ajout, de modification et de suppression dans la balise active, car ce contrôle servira seulement à afficher des données. Une fois le contrôle `DataGridView` en place, vous pouvez modifier sa propriété `Anchor` pour "ancrer" ses côtés aux bords de son conteneur afin qu'il soit redimensionné avec celui-ci.

Insérez maintenant deux contrôles `GroupBox` et des boutons radio.



◄ **Figure 4-11 :**
Onglet Classements

Votre interface est maintenant terminée. Il ne reste que peu de code à écrire pour assurer le fonctionnement de l'application.

4.3 Réalisation

La gestion des données est faite par les contrôles `ParticipantBindingSource` et `ParticipantBindingNavigator`, par l'intermédiaire des contrôles liés et de la barre d'outils du `BindingNavigator`. Cependant, votre application n'est pas capable, telle qu'elle est, de charger et d'enregistrer des données.

Chargement et enregistrement des données

La classe `Participant` n'ayant pas de constructeur par défaut, le contrôle `ParticipantBindingSource` n'est pas capable par lui-même de créer de nouvelles instances. Vérifiez que la propriété `AllowNew` de la source est à `True` et créez le gestionnaire d'événements `AddingNew` du contrôle.

```
Private Sub ParticipantBindingSource_AddingNew( _
  ByVal sender As System.Object, -
  ByVal e As System.ComponentModel.AddingNewEventArgs) _
  Handles ParticipantBindingSource.AddingNew
    e.NewObject = New Participant(Me.NomTextBox.Text, _
      Me.PrenomTextBox.Text, _
      Me.DateNaissanceDateTimePicker.Value, _
      Me.FemininCheckBox.Checked)
End Sub
```

▲ Gestionnaire d'événements AddingNew

Désormais, vous pouvez gérer les événements des boutons **Ouvrir** et **Enregistrer** de la barre d'outils pour enregistrer et charger des listes de participants sérialisées. Double-cliquez sur les boutons pour créer leurs gestionnaires d'événements.

```
Private Sub OuvrirToolStripButton_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles OuvrirToolStripButton.Click
   Dim Formatter As New _
    Runtime.Serialization.Formatters.Binary.BinaryFormatter
   Dim ofd As New OpenFileDialog
   ofd.InitialDirectory = _
     My.Computer.FileSystem.SpecialDirectories.MyDocuments

   If ofd.ShowDialog(Me) = DialogResult.OK Then
       Dim fs As New IO.FileStream(ofd.FileName, _
         IO.FileMode.Open)

       Try
           Me.ParticipantBindingSource.DataSource = _
             CType(Formatter.Deserialize(fs), _
                   IList(Of Participant))
```

```
        Catch ex As Exception
            MessageBox.Show("La récupération a échouée : " _
              + vbCrLf + ex.Message, "Erreur", _
              MessageBoxButtons.OK, MessageBoxIcon.Error)
        Finally
            fs.Close()
        End Try
    End If
End Sub

Private Sub EnregistrerToolStripButton_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles EnregistrerToolStripButton.Click
   Dim Formatter As New _
    Runtime.Serialization.Formatters.Binary.BinaryFormatter
   Dim sfd As New SaveFileDialog
   sfd.InitialDirectory = _
     My.Computer.FileSystem.SpecialDirectories.MyDocuments

   If sfd.ShowDialog(Me) = DialogResult.OK Then
       Dim fs As New IO.FileStream(sfd.FileName, _
         IO.FileMode.Create)

       Try
           Formatter.Serialize(fs, _
             Me.ParticipantBindingSource.List)
       Catch ex As Exception
           MessageBox.Show("La sauvegarde a échoué : " _
             + vbCrLf + ex.Message, "Erreur", _
             MessageBoxButtons.OK, MessageBoxIcon.Error)
       Finally
           fs.Close()
       End Try
   End If
End Sub
```

▲ Sérialisation et désérialisation de la liste de participants

Dans chacun des gestionnaires d'événements, on crée un `BinaryFormatter` qui sera chargé de réaliser la sérialisation et la désérialisation.

Dans le gestionnaire du bouton **Ouvrir**, on utilise un `OpenFileDialog` pour récupérer le nom du fichier, tandis que, dans celui du bouton **Enregistrer**, on utilise un `SaveFileDialog`.

On emploie les méthodes `Serialize` et `Deserialize` du `BinaryFormatter` que l'on a créé pour sérialiser et désérialiser la liste de participants accessible par la propriété `List` de l'objet `ParticipantBindingSource`.

Classement des résultats

Pour finir, vous allez créer le gestionnaire d'événements des boutons radio du troisième onglet, qui permettra l'affichage des résultats classés dans le contrôle `DataGridView`.

Avant de créer le gestionnaire d'événements des contrôles `RadioButton`, vous devez implémenter des classes auxiliaires dans la classe `Participant` pour permettre la comparaison des résultats des participants.

```
Public Class ComparerParNombrePoissons
    Implements IComparer(Of Participant)

    Public Function Compare(ByVal x As Participant, _
      ByVal y As Participant) As Integer _
      Implements IComparer(Of Participant).Compare
        Return x.NombrePoissons.CompareTo(y.NombrePoissons)
    End Function
End Class

Public Class ComparerParMasseTotalePoisson
    Implements IComparer(Of Participant)

    Public Function Compare(ByVal x As Participant, _
      ByVal y As Participant) As Integer _
      Implements IComparer(Of Participant).Compare
        Return x.MasseTotalePoisson.CompareTo( _
          y.MasseTotalePoisson)
    End Function
End Class

Public Class ComparerParMassePlusGrosPoisson
    Implements IComparer(Of Participant)

    Public Function Compare(ByVal x As Participant, _
      ByVal y As Participant) As Integer _
      Implements IComparer(Of Participant).Compare
        Return x.MassePlusGrosPoisson.CompareTo( _
          y.MassePlusGrosPoisson)
    End Function
End Class
```

▲ Classes internes de la classe Participant

L'utilisation d'interfaces génériques `IComparer(Of T)` évite la conversion inutile des données et facilite la lecture des méthodes.

Sélectionnez maintenant tous les contrôles `RadioButton` en cliquant dessus en maintenant la touche Ctrl enfoncée. Saisissez `Classement_Click` comme valeur de l'événement `Click` dans le volet des propriétés et appuyez sur la touche Entrée pour créer le gestionnaire.

```
Private Sub Classement_Click( _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles optGros.Click, optNombre.Click, _
  optGeneral.Click, optJunior.Click, optFemmes.Click, _
  optHommes.Click, optTous.Click

    Dim ClassementConcours As New List(Of Participant)

    If Not optTous.Checked Then
        Dim participant As Participant
        For Each participant In _
          Me.ParticipantBindingSource.List
            If optJunior.Checked AndAlso _
              participant.Age < 17 Then
                ClassementConcours.Add(participant)
            ElseIf Me.optHommes.Checked AndAlso _
              participant.Feminin = False Then
                ClassementConcours.Add(participant)
            ElseIf Me.optFemmes.Checked AndAlso _
              participant.Feminin Then
                ClassementConcours.Add(participant)
            End If
        Next
    Else
        ClassementConcours.AddRange( _
          Me.ParticipantBindingSource.List)
    End If

    If Me.DataGridView1.DataSource IsNot Nothing Then
        Me.DataGridView1.DataSource = Nothing
    End If

    Select Case True
      Case Me.optGeneral.Checked
          ClassementConcours.Sort( _
            New Participant.ComparerParMasseTotalePoisson)
          With Me.DataGridView1
           .DataSource = ClassementConcours
           .Columns("NomComplet").DisplayIndex = 0
           .Columns("MasseTotalePoisson").Visible = True
           .Columns("MasseTotalePoisson").DisplayIndex = 1
           .Columns("MassePlusGrosPoisson").Visible = True
           .Columns("MassePlusGrosPoisson").DisplayIndex = 2
           .Columns("NombrePoissons").Visible = True
           .Columns("NombrePoissons").DisplayIndex = 3
          End With

      Case Me.optGros.Checked
         ClassementConcours.Sort( _
```

```
            New Participant.ComparerParMassePlusGrosPoisson)
          With Me.DataGridView1
           .DataSource = ClassementConcours
           .Columns("NomComplet").DisplayIndex = 0
           .Columns("MassePlusGrosPoisson").DisplayIndex = 1
           .Columns("MasseTotalePoisson").Visible = False
           .Columns("NombrePoissons").Visible = False
          End With

        Case Me.optNombre.Checked
          ClassementConcours.Sort( _
            New Participant.ComparerParNombrePoissons)
          With Me.DataGridView1
           .DataSource = ClassementConcours
           .Columns("NomComplet").DisplayIndex = 0
           .Columns("NombrePoissons").DisplayIndex = 1
           .Columns("MasseTotalePoisson").Visible = False
           .Columns("MassePlusGrosPoisson").Visible = False
          End With

    End Select

    With Me.DataGridView1
        .Columns("Age").Visible = False
        .Columns("DateDeNaissance").Visible = False
        .Columns("Feminin").Visible = False
        .Columns("Nom").Visible = False
        .Columns("Prenom").Visible = False
    End With
End Sub
```

▲ Méthode pour le classement

Cette méthode vérifie quel est le bouton radio du groupe *Catégories* qui est coché et génère une liste temporaire de type `List(Of Participant)` en vue d'un affichage dans le contrôle `DataGridView`. Si la case *Tous* est cochée, tous les éléments stockés dans la liste du contrôle `ParticipantBindingSource` seront ajoutés à la liste temporaire ; sinon, une sélection sera faite.

Enfin, en fonction du bouton radio du groupe *Classement* qui sera coché, une instance des classes internes de la classe `Participant` sera créée pour que la liste temporaire soit triée grâce à sa méthode `Sort`.

Toutes les fonctionnalités de votre application sont désormais implémentées.

4.4 Check-list

En développant cette application, vous avez appris à :

- créer des classes sérialisables et stocker des collections de celles-ci dans un fichier ;
- utiliser Visual Basic 2005 Express pour créer des sources de données à partir d'objets ;
- créer des interfaces graphiques liées à des sources de données ;
- implémenter des interfaces pour permettre la comparaison de classes personnalisées.

Outil de traitement de texte RTF

Vous allez créer, dans ce chapitre, une application qui vous permettra d'éditer des fichiers au format RTF (Rich Text Format, en français "format de texte enrichi").

RTF est fréquemment utilisé pour l'échange de données formatées aussi bien entre programmes qu'entre plateformes, car il est reconnu par beaucoup d'applications et, plus particulièrement, par les logiciels de traitement de texte des différents systèmes d'exploitation.

L'application que vous allez créer aura une interface multidocument (MDI), qui permettra d'ouvrir, simultanément, au sein d'un même formulaire, dit "parent", plusieurs sous-formulaires, dits "enfants".

Adobe Photoshop est un exemple d'application MDI, tout comme Microsoft Visual Basic 2005 Express Edition, qui utilise des onglets par défaut, mais qui peut être converti en application MDI grâce à la commande **Options** du menu **Outils**.

5.1 Classes et espaces de noms utilisés

Comme pour toute application Windows, vous aurez besoin de l'espace de noms `System.Windows.Forms`.

Dans cet espace de noms, vous trouverez la classe `RichTextBox`, qui sera le cœur de votre outil de traitement de texte.

Pensez aussi à importer l'espace de noms `System.IO`, qui vous servira pour certaines manipulations de fichiers.

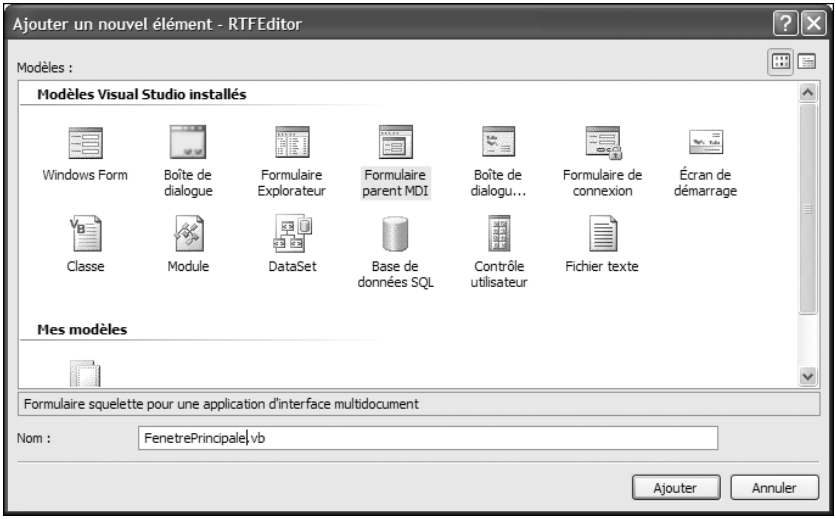
5.2 Interface utilisateur

Une grande partie de votre interface graphique est déjà prête. Vous allez comprendre pourquoi.

Formulaire principal

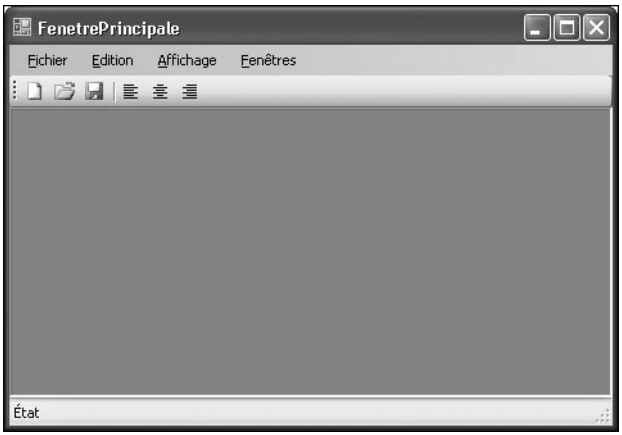
Vous allez commencer par créer le formulaire principal (celui qui hébergera les formulaires enfants) de votre application dans un nouveau projet.

1. Créez une nouvelle application Windows appelée `RTFEditor`.
2. Dans l'Explorateur de solutions, supprimez le fichier *Form1.vb* en cliquant sur celui-ci du bouton droit et en sélectionnant la commande **Supprimer**.
3. Pour créer la fenêtre principale de votre application, cliquez du bouton droit sur le nom de votre projet dans l'Explorateur de solutions et sélectionnez la commande **Nouvel élément** du sous-menu **Ajouter**.



▲ **Figure 5-1** : Ajouter un nouveau formulaire parent MDI

4 Dans l'Assistant, sélectionnez le modèle *Formulaire parent MDI* et, dans la zone de texte *Nom*, saisissez `FenetrePrincipale.vb`. Si vous cliquez sur le bouton **Ajouter**, Visual Studio crée une fenêtre complète, avec les menus et les barres d'outils les plus utilisés, et prête à recevoir des formulaires enfants.



▲ **Figure 5-2** : Fenêtre principale

Remarque

Vos propres conteneurs MDI

Le modèle de formulaire MDI proposé par Visual Studio peut vous faire gagner du temps, car, en quelques clics, vous obtenez un formulaire avec menus et barres d'outils, prêt à l'emploi.

Toutefois, vous n'êtes pas obligé d'utiliser ce modèle. En effet, tout formulaire est susceptible de devenir un conteneur MDI. Il suffit pour cela de changer la valeur de sa propriété `IsMdiContainer` à `True`. Vous remarquerez que sa bordure et sa couleur de fond changent, signalant qu'il est prêt à accueillir des formulaires enfants.

Le bouton **Aide** a été supprimé de la barre d'outils ainsi que les deux boutons qui concernent l'impression. Les menus **Aide** et **Outils**, et les commandes du menu **Fichier** qui concernent l'impression ont aussi été supprimés, car ces fonctionnalités ne seront pas implémentées.

Vous devez ajouter trois boutons à la barre d'outils :

- `GaucheToolStripButton` ;
- `CentreToolStripButton` ;
- `DroiteToolStripButton`.

◀ **Figure 5-3 :** Barre d'outils de la fenêtre principale

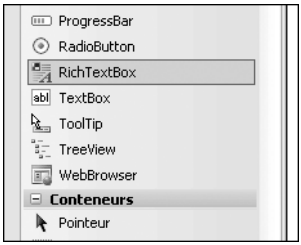
Le formulaire principal de votre application est prêt à recevoir les formulaires enfants que vous allez préparer dans la section suivante.

Formulaires enfants

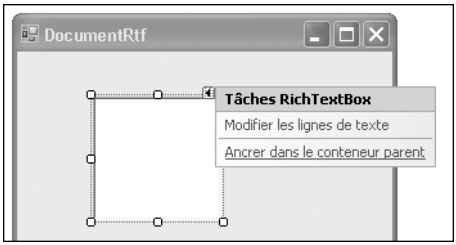
Maintenant que le formulaire principal est prêt, vous devez créer une nouvelle classe qui représentera les formulaires enfants : ceux qui vous permettront d'éditer vos documents RTF.

1. Pour cela, ajoutez un nouvel élément à votre projet, comme vous l'avez fait pour le formulaire principal. Choisissez, toutefois, *Windows Form* comme modèle et appelez-le *DocumentRtf.vb*.
2. Insérez maintenant un contrôle `RichTextBox` en double-cliquant sur celui-ci dans la boîte à outils.
3. Cliquez sur la flèche de balise active dans le coin supérieur droit du contrôle que vous venez d'insérer.

4 Cliquez sur **Ancrer dans le conteneur parent** pour attribuer à la propriété `Dock` la valeur `Fill`. Le contrôle `RichTextBox` s'ancre aux limites du formulaire `DocumentRtf`.



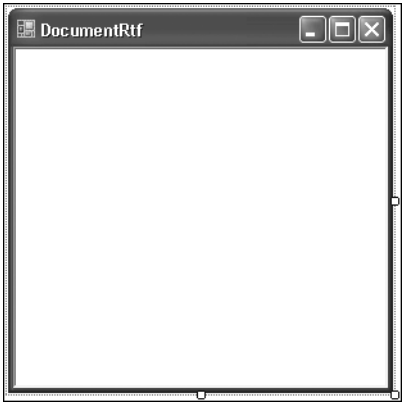
◄ **Figure 5-4 :** Insérer un RichTextBox



◄ **Figure 5-5 :** Ancrer le RichTextBox au conteneur parent

5 Dans la propriété `(name)` du contrôle `RichTextBox`, saisissez `RichTextBox`.

Votre formulaire d'édition est terminé. Ce formulaire, et plus particulièrement le contrôle `RichTextBox` qu'il contient, seront utilisés dans le formulaire principal pour éditer un ou plusieurs documents RTF simultanément.

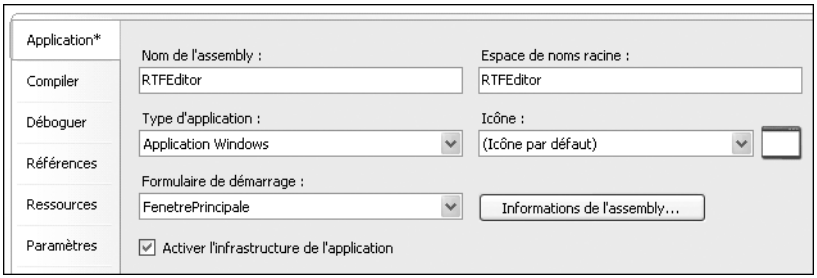


◄ **Figure 5-6 :** Formulaire d'édition

Définition du formulaire de démarrage

Les différents éléments de votre interface graphique sont prêts. Toutefois, si vous essayez de compiler et de lancer l'application, vous allez vous retrouver avec une erreur du compilateur. Cela est dû au fait que le formulaire par défaut, *Form1.vb*, a été supprimé. Vous devez donc spécifier, manuellement, le formulaire de démarrage de votre application.

1. Dans l'Explorateur de solutions, cliquez du bouton droit sur le nom de votre projet et sélectionnez **Propriétés**.
2. Sous l'onglet **Application**, sélectionnez `FenetrePrincipale` dans la liste déroulante *Formulaire de démarrage*.



▲ **Figure 5-7 :** Modifier le formulaire de démarrage de l'application

3. Fermez l'onglet à l'aide de la croix.

Votre interface graphique est maintenant prête. Il ne reste plus qu'à implémenter les fonctionnalités de base d'un outil de traitement de texte. Au préalable, vous pouvez la tester en appuyant sur la touche F5 de votre clavier.

5.3 Réalisation

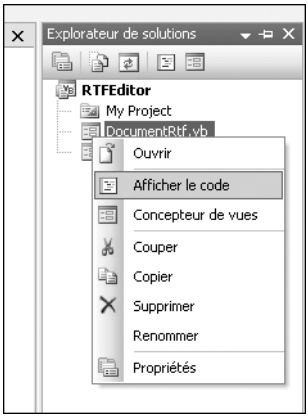
Visual Studio a préparé votre formulaire principal avec des éléments graphiques qui donnent l'accès à la plupart des fonctionnalités proposées par un outil de traitement de texte. Vous en avez ajouté d'autres lors de la préparation de celui-ci. Toutefois, ces fonctionnalités ne sont toujours pas implémentées.

Le modèle de formulaire MDI a déjà créé certains gestionnaires d'événements pour vous, par exemple celui qui permet de créer de nouveaux sous-formulaires à l'intérieur de la fenêtre principale ou encore celui qui affiche une boîte de dialogue pour ouvrir un document. Cependant, vous devez les modifier pour qu'ils correspondent exactement à vos besoins.

Compléter la classe DocumentRtf

Au niveau de l'interface graphique, la classe `DocumentRtf` est prête, mais vous devez encore écrire un peu de code afin de l'utiliser par la suite.

Affichez le code de la classe `DocumentRtf` en cliquant du bouton droit sur le fichier de même nom dans l'Explorateur de solutions et en sélectionnant la commande **Afficher le code**.



◄ **Figure 5-8 :** Afficher le code de la classe DocumentRtf

Écrivez le mot `Property` et appuyez sur la touche Tab de votre clavier. Cela a pour effet d'insérer un extrait de code correspondant à une propriété publique et son membre privé associé. Remplacez les noms et les types insérés par défaut pour obtenir le code suivant :

```
Private m_nomDuFichier As String = String.Empty

Public Property NomDuFichier() As String
    Get
        Return m_nomDuFichier
    End Get
    Set(ByVal value As String)
        m_nomDuFichier = value
    End Set
End Property
```

Ce champ et cette propriété serviront à identifier les différents documents ouverts dans votre application, ce qui permettra, par exemple, de n'ouvrir qu'une seule instance d'un document à la fois.

Vous pouvez fermer le code source et le Concepteur de vues du fichier *DocumentRtf.vb* car il n'a plus besoin d'être modifié.

Créer de nouveaux documents

Si vous testez votre interface graphique, vous pouvez constater que le bouton **Nouveau** de la barre d'outils et la commande **Nouveau** du menu **Fichier** vous permettent déjà de créer de nouvelles fenêtres à l'intérieur du formulaire principal de votre application. Toutefois, ces fenêtres ne sont que de simples instances de la classe `Form`, sans fonctionnalité supplémentaire.

Vous devez modifier la méthode `ShowNewForm` pour ouvrir de nouvelles fenêtres `DocumentRtf` dans votre application. Double-cliquez sur le bouton **Nouveau** de la barre d'outils afin d'accéder au code source de la méthode `ShowNewFrom`. Modifiez la déclaration de la variable `ChildForm` comme suit :

```
Private Sub ShowNewForm(ByVal sender As Object, _
                        ByVal e As EventArgs) _
  Handles NewToolStripMenuItem.Click, _
          NewToolStripButton.Click, _
          NewWindowToolStripMenuItem.Click
    ' Créez une nouvelle instance du formulaire enfant.
    Dim ChildForm As New DocumentRtf
    ' Configurez-la en tant qu'enfant de ce formulaire MDI
    ' avant de l'afficher.
    ChildForm.MdiParent = Me
    m_ChildFormNumber += 1
    ChildForm.Text = "Fenêtre " & m_ChildFormNumber
    ChildForm.Show()
End Sub
```

▲ Création d'un nouveau formulaire

Les nouveaux formulaires enfants seront de type `DocumentRtf`.

La méthode définit ensuite le formulaire courant, `Me`, comme parent MDI du formulaire que vous venez de créer en utilisant la propriété `MdiParent` de celui-ci, et incrémente un compteur qui contient le nombre de nouveaux enfants. Le titre du nouveau formulaire est ensuite modifié et celui-ci est affiché.

Ouvrir un document existant

Comme pour la commande **Nouveau**, il existe déjà un gestionnaire d'événements qui est appelé lorsque l'on clique sur le bouton **Ouvrir** ou lorsque l'on sélectionne la commande **Ouvrir** du menu **Fichier**. La méthode s'appelle `OpenFile` et elle est capable d'afficher une boîte de dialogue permettant de sélectionner un fichier à ouvrir et de récupérer le nom de ce fichier dans une variable. Mais c'est à vous de l'étendre pour qu'un nouveau formulaire de type `DocumentRtf` soit ouvert et affiche effectivement le fichier sélectionné.

Toutefois, avant d'ouvrir le fichier demandé, vous allez d'abord vérifier qu'il n'est pas déjà ouvert afin d'éviter d'afficher le même document plusieurs fois. Pour cela, vous allez créer une fonction privée au sein de la classe `FenetrePrincipale`.

```
Private Function RechercherDocument(ByRef nom As String) _
  As DocumentRtf
    For Each Document As DocumentRtf In MdiChildren
        If Document.NomDuFichier.Equals(nom) Then
            Return Document
        End If
    Next
    Return Nothing
End Function
```

▲ Recherche d'un document ouvert

Cette fonction prend une chaîne de caractères en paramètre, qui correspond au nom complet du fichier que l'on souhaite ouvrir. On parcourt la collection de formulaires enfants de la fenêtre principale, `MdiChildren`, tout en vérifiant si leur propriété `NomDuFichier` est égale au nom demandé. Si c'est le cas, le document correspondant est renvoyé. Si l'on arrive à la fin de la collection sans trouver le nom demandé, la valeur `Nothing` est renvoyée.

Vous pouvez à présent modifier la méthode `OpenFile` afin d'ouvrir le document souhaité ou ramener sa fenêtre vers le front s'il est déjà ouvert.

```
Private Sub OpenFile(ByVal sender As Object, _
                     ByVal e As EventArgs) _
  Handles OpenToolStripMenuItem.Click,
          OpenToolStripButton.Click
    Dim OpenFileDialog As New OpenFileDialog
    OpenFileDialog.InitialDirectory = _
      My.Computer.FileSystem.SpecialDirectories.MyDocuments
    OpenFileDialog.Filter = _
    "Fichiers RTF (*.rtf)|*.rtf|Tous les fichiers (*.*)|*.*"

    If (OpenFileDialog.ShowDialog(Me) = _
        System.Windows.Forms.DialogResult.OK) Then
        Dim FileName As String = OpenFileDialog.FileName
        Dim ChildForm As DocumentRtf = _
             RechercherDocument(FileName)
        If ChildForm IsNot Nothing Then
            ChildForm.BringToFront()
        Else
            ChildForm = New DocumentRtf
            ChildForm.RichTextBox.LoadFile(FileName)
            ChildForm.MdiParent = Me
            ChildForm.Text = Path.GetFileName(FileName)
            ChildForm.NomDuFichier = FileName
```

```
            ChildForm.Show()
        End If
    End If
End Sub
```

▲ Ouverture d'un fichier RTF

On commence par préparer une nouvelle boîte de dialogue de type `OpenFileDialog` pour sélectionner le fichier à ouvrir. On doit modifier le filtre utilisé dans cette boîte de dialogue afin qu'elle affiche les fichiers **.rtf*. La boîte de dialogue est ensuite affichée et, si l'utilisateur la valide grâce au bouton OK ou à la touche Entrée, on récupère le nom du fichier sélectionné dans la variable `FileName`.

On doit ensuite créer une variable appelée `ChildForm` à laquelle on affectera le résultat de la recherche du document souhaité parmi ceux qui sont déjà ouverts. Si le document demandé est déjà ouvert, la variable `ChildForm` contiendra une référence vers la fenêtre qui le contient, ce qui permettra de la ramener vers le front plus tard, sinon elle contiendra la valeur `Nothing`.

On teste si la recherche a abouti. Si c'est le cas, il suffit d'appeler la méthode `BringToFront` pour amener le document souhaité par-dessus les autres documents ouverts. Sinon, on stocke dans la variable `ChildFrom` une nouvelle instance de la classe `DocumentRtf` et l'on définit le formulaire courant en tant que son parent MDI, grâce à sa propriété `MdiParent`. On accède ensuite au contrôle `RichTextBox` du nouveau formulaire en utilisant la propriété de même nom et l'on appelle sa méthode `LoadFile` en passant le nom du fichier en argument pour le charger. On utilise ensuite la méthode statique `GetFileName` de la classe `Path` pour extraire le nom du fichier et l'utiliser comme titre de la nouvelle fenêtre que l'on affichera par la suite. Il faut stocker le nom du fichier, qui se trouve dans la variable `FileName`, dans la propriété `NomDuFichier` du document, car il permettra d'identifier le document si jamais on essaye de le rouvrir. Il permettra aussi d'enregistrer le document sans avoir à redemander l'emplacement de celui-ci à l'utilisateur.

Vous pouvez à présent tester votre application en ouvrant un fichier RTF créé avec une autre application, telle que Microsoft Word ou WordPad.

Enregistrer un document

Lorsque l'utilisateur veut enregistrer un document, il faut considérer trois cas possibles :

- L'utilisateur souhaite enregistrer un nouveau document et il clique sur la commande **Enregistrer** du menu **Fichier** ou sur le bouton **Enregistrer** de la barre d'outils.
- L'utilisateur clique sur la commande **Enregistrer sous** du menu **Fichier**.

- L'utilisateur clique sur la commande **Enregistrer** du menu **Fichier** ou sur le bouton **Enregistrer** de la barre d'outils pour enregistrer un fichier qui existe déjà.

Dans les deux premiers cas, une boîte de dialogue s'ouvre permettant à l'utilisateur de choisir le nom sous lequel il souhaite enregistrer le document. Dans le troisième cas, cela n'est pas nécessaire, car l'emplacement du fichier est connu.

Vous allez gérer les deux premiers cas ensemble en vous servant du gestionnaire d'événements `SaveAsToolStripMenuItem_Click`. Il a déjà été créé par le modèle que vous avez utilisé en concevant votre formulaire principal. Comme pour le gestionnaire de la commande **Ouvrir**, vous devez compléter cette méthode afin d'indiquer ce qui devra être enregistré.

```
Private Sub SaveAsToolStripMenuItem_Click( _
  ByVal sender As Object, ByVal e As EventArgs) _
  Handles SaveAsToolStripMenuItem.Click
    If ActiveMdiChild IsNot Nothing Then
      Dim SaveFileDialog As New SaveFileDialog
      SaveFileDialog.InitialDirectory = _
      My.Computer.FileSystem.SpecialDirectories.MyDocuments
      SaveFileDialog.Filter = _
      "Fichiers RTF (*.rtf)|*.rtf|Tous les fichiers (*.*)|*.*"

       If (SaveFileDialog.ShowDialog(Me) = _
          System.Windows.Forms.DialogResult.OK) Then
          Dim FileName As String = SaveFileDialog.FileName
          Dim FenetreActive As DocumentRtf = ActiveMdiChild
          FenetreActive.RichTextBox.SaveFile(FileName)
          FenetreActive.Text = Path.GetFileName(FileName)
       End If
    End If
End Sub
```

▲ Enregistrer un document RTF dans un nouveau fichier

L'utilisateur ne peut pas enregistrer de fichier s'il n'a pas ouvert de document. Pour cette raison, on commence par entourer le code existant par une clause `If… Else` qui teste l'existence d'un formulaire enfant en vérifiant que la propriété `ActiveMdiChild` contient une valeur.

Comme pour la commande **Ouvrir**, on prépare une boîte de dialogue, de type `SaveFileDialog` cette fois, pour récupérer le nom du fichier. On doit aussi modifier le filtre de cette boîte de dialogue pour afficher des fichiers de type *.rtf*.

Si l'utilisateur valide sa sélection, le nom du fichier est stocké dans une variable appelée `FileName`. On doit alors créer une autre variable appelée `FenetreActive` de type `DocumentRtf` qui référence le formulaire enfant actif, que l'on récupère

grâce à la propriété ActiveMdiChild. On procède ainsi pour accéder à la RichTextBox du DocumentRtf.

Il ne reste plus qu'à appeler la méthode SaveFile du contrôle RichTextBox de la fenêtre active, en lui passant le nom du fichier contenu dans la variable FileName, pour que celui-ci soit enregistré. On met ensuite à jour le titre de la fenêtre active pour refléter le nouveau nom du document.

Le troisième cas est plus rapide à écrire.

Double-cliquez sur le bouton **Enregistrer** du menu **Fichier** de votre fenêtre principale dans le Concepteur de vues et complétez le gestionnaire d'événements qui est créé.

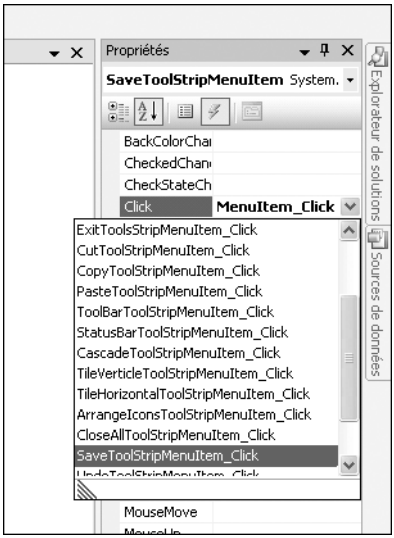
```
Private Sub SaveToolStripMenuItem_Click(ByVal sender As _
                            System.Object, _
                            ByVal e As System.EventArgs) _
 Handles SaveToolStripMenuItem.Click, _
        SaveToolStripButton.Click
  If ActiveMdiChild IsNot Nothing Then
        Dim FenetreActive As DocumentRtf = ActiveMdiChild

   If String.IsNullOrEmpty(FenetreActive.NomDuFichier) Then
     SaveAsToolStripMenuItem.PerformClick()
   Else
     FenetreActive.RichTextBox.SaveFile(FenetreActive.NomDuFichier)
   End If
  End If
End Sub
```

▲ Enregistrer le nom du fichier

Une fois de plus, on doit vérifier qu'il y a bien un document à enregistrer en s'assurant qu'il y a un formulaire enfant actif. Si c'est le cas, on stocke une référence vers la fenêtre active dans une variable FenetreActive.

À l'aide de la méthode IsNotNullOrEmpty de la classe String, on vérifie la propriété NomDuFichier du document à enregistrer. Si la méthode retourne True, cela veut dire qu'il s'agit d'un nouveau document pour lequel il faut demander un nom. Mais comme cette fonctionnalité a déjà été programmée pour la commande **Enregistrer sous**, il ne reste plus qu'à simuler un clic sur celle-ci en utilisant la méthode PerformClick. Dans le cas contraire, on appelle la méthode SaveFile du contrôle RichTextBox de la fenêtre active en lui passant la propriété NomDuFichier comme emplacement pour sauvegarder le fichier.

Afin d'attacher ce gestionnaire d'événements à la commande **Enregistrer** du menu **Fichier**, cliquez sur celle-ci puis sur le bouton en forme d'éclair dans le volet des propriétés pour afficher les événements disponibles. Cliquez sur l'événement Click puis sur le bouton fléché pour sélectionner la méthode SaveToolStripMenuItem_Click dans la liste.



◄ **Figure 5-9 :** Associer la méthode à l'événement Click

Maintenant que vous avez implémenté l'ouverture et l'enregistrement de fichiers, vous pouvez utiliser votre application pour créer de nouveaux fichiers RTF et éditer des fichiers créés par d'autres applications.

Menu Edition

Le modèle de formulaire parent que vous utilisez contient un menu **Edition** avec des commandes proposées par la plupart des applications Windows. Ces commandes, qui font souvent penser à des manipulations de piles ou à la gestion des différents types de données dans le Presse-papiers, sont gérées par la classe `RichTextBox`.

Les gestionnaires d'événements des six commandes seront virtuellement identiques, sauf pour la commande qui est appelée sur la `RichTextBox`. Ils respecteront tous le modèle suivant :

```
If ActiveMdiChild IsNot Nothing Then
    Dim FenetreActive As DocumentRtf = ActiveMdiChild
    FenetreActive.RichTextBox.[Commande à Executer]
End If
```

▲ Modèle de code des commandes du menu Edition

Pour créer chacun des gestionnaires, double-cliquez sur la commande que vous souhaitez implémenter et copiez ce modèle en remplaçant le texte entre crochets par l'appel à la méthode appropriée.

Ainsi, pour la commande **Annuler**, la méthode à appeler est `Undo` et le gestionnaire d'événements ressemblera à ceci :

```
Private Sub UndoToolStripMenuItem_Click(ByVal sender As _
                            System.Object, _
                            ByVal e As System.EventArgs) _
    Handles UndoToolStripMenuItem.Click
    If ActiveMdiChild IsNot Nothing Then
        Dim FenetreActive As DocumentRtf = ActiveMdiChild
        FenetreActive.RichTextBox.Undo()
    End If
End Sub
```

▲ Gestionnaire de la commande Annuler

Le tableau suivant indique les méthodes à appeler pour chacune des six commandes :

Méthodes du contrôle RichTextBox correspondant aux commandes du menu Edition

Commande	Méthode
Annuler	`Undo()`
Rétablir	`Redo()`
Couper	`Cut()`
Copier	`Copy()`
Coller	`Paste()`
Sélectionner tout	`SelectAll()`

Aligner le texte

Pour le moment, vous n'avez fait qu'implémenter des comportements pour les éléments de l'interface que Visual Studio a préparée pour vous. Or, lorsque vous avez créé la fenêtre principale de votre application, vous avez ajouté trois boutons à la barre d'outils permettant d'aligner horizontalement le texte du document.

Comme pour les commandes précédentes, ce ne sera pas à vous d'aligner directement le texte. Vous allez simplement modifier une propriété du contrôle `RichTextBox` de la fenêtre active pour changer l'alignement du texte sélectionné. Cependant, le code pour chacun des trois boutons étant quasiment identique, vous allez créer un seul gestionnaire d'événements que vous associerez aux trois boutons en question et qui déterminera automatiquement l'alignement à appliquer en fonction du bouton sur lequel l'utilisateur aura cliqué.

Commencez par écrire le gestionnaire d'événements pour les trois boutons :

```
Private Sub AlignerSelection(ByVal sender As Object, _
                             ByVal e As EventArgs)
    If ActiveMdiChild IsNot Nothing Then
        Dim FenetreActive As DocumentRtf = ActiveMdiChild
        Dim NouvelAlignement As HorizontalAlignment

        If sender.Equals(GaucheToolStripButton) Then
            NouvelAlignement = HorizontalAlignment.Left
        ElseIf sender.Equals(CentreToolStripButton) Then
            NouvelAlignement = HorizontalAlignment.Center
        ElseIf sender.Equals(DroiteToolStripButton) Then
            NouvelAlignement = HorizontalAlignment.Right
        End If

        FenetreActive.RichTextBox.SelectionAlignment = _
          NouvelAlignement
    End If
End Sub
```

▲ Aligner la sélection

Il faut vérifier qu'un document est ouvert avant de faire une quelconque modification.

Si un document est bien ouvert, on procède à la création de deux variables, `FenetreActive` et `NouvelAlignement`, qui contiendront respectivement une référence vers la fenêtre active, de type `DocumentRtf`, pour récupérer son contrôle `RichTextBox`, et l'alignement souhaité, de type `HorizontalAlignment`.

La suite de clauses `If… Else` permet d'affecter une valeur de l'énumération `HorizontalAlignment` à la variable `NouvelAlignement` en fonction du bouton qui a déclenché l'événement. Cette valeur est ensuite affectée à la propriété `SelectionAlignement` du contrôle `RichTextBox` de la fenêtre active.

Revenez dans le Concepteur de vues afin d'associer le gestionnaire d'événements que vous venez de créer aux trois boutons. Pour ce faire :

1. Sélectionnez les trois boutons en cliquant sur le premier puis sur les deux autres tout en maintenant la touche Ctrl enfoncée.
2. Cliquez sur le bouton en forme d'éclair dans le volet des propriétés pour afficher les événements disponibles pour les trois boutons.
3. Cliquez sur l'événement `Click` puis sur le bouton fléché pour sélectionner la méthode `AlignerSelection` dans la liste.

Votre application est désormais un outil de traitement de texte RTF à part entière. Certes, toutes les possibilités de formatage offertes par RTF ne sont pas exploitées, mais si vous le souhaitez, vous pouvez facilement étendre votre

application en suivant les modèles de code que vous avez déjà écrits et en modifiant d'autres propriétés du contrôle `RichTextBox`, par exemple la propriété `SelectionColor`, qui permet de changer la couleur du texte sélectionné.

5.4 Check-list

Dans ce chapitre vous avez appris à :

- utiliser les modèles fournis par Visual Basic 2005 Express Edition afin d'accélérer le développement de vos applications ;
- gérer plusieurs formulaires enfants dans une interface multidocument ;
- exécuter des commandes sur les formulaires enfants d'une application MDI à partir des contrôles du formulaire parent ;
- éditer un document au format de texte enrichi grâce au contrôle `RichTextBox`.

Site web personnel

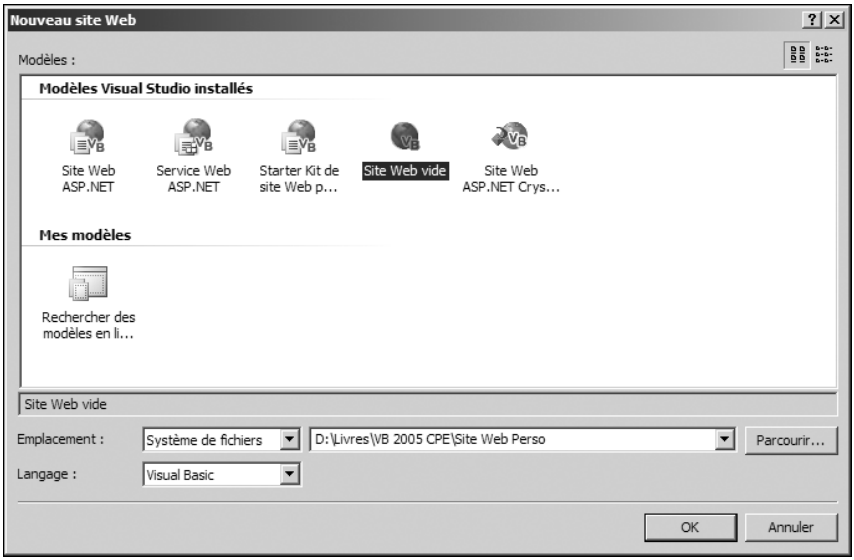
Les sites web personnels, qui permettent de créer un espace virtuel sur Internet, sont à la mode depuis quelques années. Vous allez créer le vôtre tout au long de ce chapitre pour vous présenter, présenter vos activités, votre CV, etc.

6.1 Création de l'interface

Commencez par créer votre projet web.

Pour cela, ouvrez Visual Web Developer 2005 Express et créez un nouveau projet via le menu **Fichier**.

Une boîte de dialogue s'affiche, permettant de créer différents types d'applications web. Sélectionnez le modèle de projet intitulé *Site Web vide*. Vérifiez que le langage sélectionné est le Visual Basic, nommez et placez le projet web où vous le souhaitez puis validez.

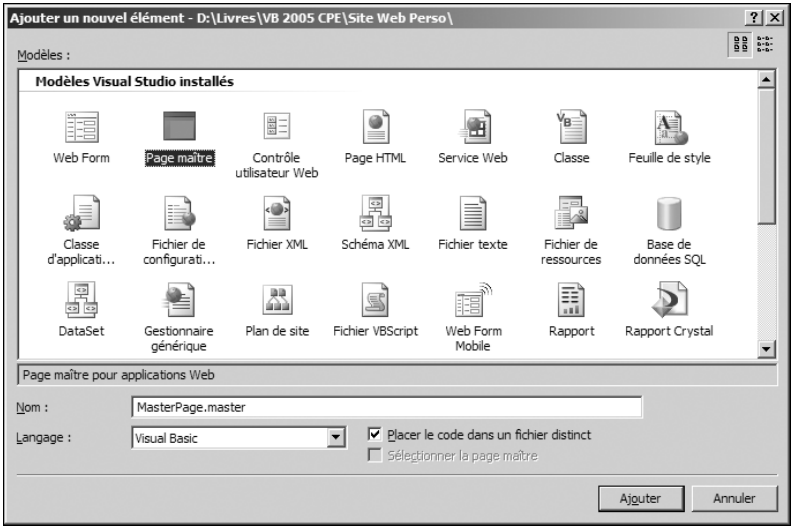


▲ **Figure 6-1 :** Modèles de projets web

Votre projet web est maintenant créé. Il est vide car il ne contient aucun fichier. L'Explorateur de solutions en rend compte.

Vous allez créer une page maître, c'est-à-dire un modèle de page à appliquer à un ensemble de pages web. Vous pouvez, par exemple, créer un modèle contenant l'interface commune à toutes les pages web d'un site, et le leur appliquer. Elles ne contiendront dès lors que le contenu variable.

Pour créer une page maître, cliquez sur le menu **Site Web** et sélectionnez **Ajouter un nouvel élément**. Dans la boîte de dialogue, sélectionnez l'élément *Page maître*.



▲ **Figure 6-2 :** Création d'une page maître

Après validation, le designer de Visual Web Developer affiche une page vierge blanche, qui contient un contrôle spécial : `ContentPlaceHolder`.

La page nouvellement créée ne sera pas accessible en tant que telle. Elle devra être utilisée par d'autres. Dans son code source, figurent, à la place de la directive `Page`, une directive `Master`, et un peu plus bas, le contrôle `ContentPlaceHolder`.

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
 Inherits="MasterPage" %>

<!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>Page sans titre</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:contentplaceholder id="ContentPlaceHolder1"
```

```
 runat="server">
        </asp:contentplaceholder>
    </div>
    </form>
</body>
</html>
```

L'objectif à présent est de définir dans la page maître l'interface qui sera commune à toutes les pages web. Celles-ci seront automatiquement affichées dans le contrôle `ContentPlaceHolder`.

Vous allez créer un tableau contenant une seule ligne et deux colonnes. La première colonne permettra de placer le menu du site web, la seconde colonne contiendra le contrôle `ContentPlaceHolder`.

Vous devez arriver à un résultat proche de celui-ci :

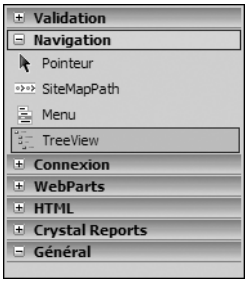
```
<table><tr><td>menu</td><td><asp:contentplaceholder
id="ContentPlaceHolder1" runat="server">
</asp:contentplaceholder></td></tr></table>
```

6.2 Création de menus

ASP .NET propose deux contrôles serveurs pour afficher les menus de sites web. Le premier est `Menu`, disponible dans la catégorie *Navigation* dans la boîte à outils. Il permet de créer des menus horizontaux et verticaux, et il est entièrement personnalisable via l'utilisation de styles CSS, d'images, etc.

Le second contrôle disponible pour la création de menus est `TreeView`. Il permet d'afficher des données de manière hiérarchique (ce qui est idéal pour un menu de site web), verticalement uniquement. Comme le contrôle `Menu`, il est personnalisable.

Sélectionnez donc le contrôle `TreeView` dans la boîte à outils et placez-le dans la première cellule du tableau précédemment créé.



◄ **Figure 6-3 :** Le contrôle TreeView dans la boîte à outils

Vous allez à présent définir l'apparence de ce contrôle. Pour ce faire, cliquez sur le smart tag lié à votre contrôle `TreeView` puis sur le lien *Mise en forme automatique*.

◄ **Figure 6-4 :** Affichage du smart tag du TreeView

Un clic sur ce lien ouvre une boîte de dialogue permettant de sélectionner différents types de mises en forme prédéfinies. Vous pouvez ainsi donner une apparence professionnelle à votre contrôle `TreeView` de manière rapide. Sélectionnez le thème Flèches, sobre et élégant.

Maintenant que le contrôle `TreeView` est disposé sur votre page maître et que son apparence est définie, il faut le remplir avec les informations concernant les différentes pages de votre site.

Il ne s'agit pas de remplir le contrôle `TreeView` "en dur" en ajoutant les différents éléments grâce à sa propriété `Nodes`. Vous allez créer un nouveau fichier de type "sitemap" dans votre projet.

Ajoutez un nouvel élément à votre projet et sélectionnez *Plan de site*. Une fois la création validée, vous disposez d'un fichier de type "sitemap", qui est en fait un simple fichier XML, avec un schéma particulier. Il permet de définir la hiérarchie d'un site web en ajoutant et en organisant des balises `siteMapNode`.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns=
"http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
    <siteMapNode url="" title=""  description="">
        <siteMapNode url="" title=""  description="" />
        <siteMapNode url="" title=""  description="" />
    </siteMapNode>
</siteMap>
```

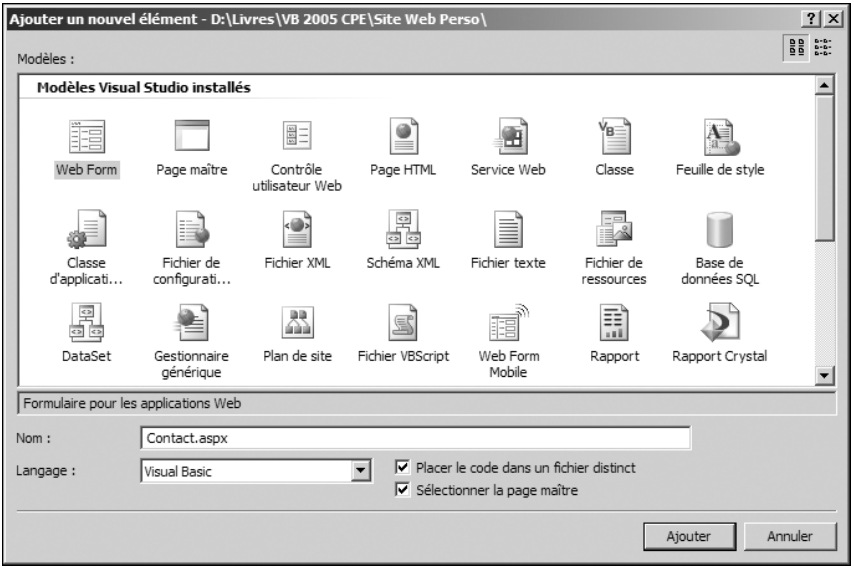
Après avoir personnalisé ce fichier, vous devez arriver à un résultat proche de celui-ci :

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns=
"http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
<siteMapNode url="default.aspx"
```

```
title="Mon Site web personnel"  description="">
<siteMapNode url="cv.aspx" title="CV" ></siteMapNode>
<siteMapNode url="links.aspx" title="Liens préférés"
 description="" />
<siteMapNode url="Contact.aspx" title="Contact"
 description="" />
</siteMapNode>
</siteMap>
```

Maintenant que le fichier est rempli, il faut le lier au contrôle `TreeView`. Le contrôle `SiteMapDataSource` est dédié à cela. Vous allez l'utiliser pour remplir le contrôle `TreeView` avec les informations présentes dans le fichier "sitemap".

En ce sens, placez un contrôle `SiteMapDataSource` sur votre page maître et cliquez sur le smart tag du contrôle `TreeView` pour le lier au contrôle `SiteMapDataSource` grâce à la liste déroulante *Choisir la source de données*. Une fois la source de données sélectionnée, le résultat final de votre menu apparaît en mode Conception dans le designer de Visual Web Developer.



▲ **Figure 6-5 :** Sélection de la source de données du contrôle TreeView

6.3 Gestion des liens

Les sites personnels proposent couramment de consulter une liste de liens favoris que l'auteur du site affectionne. Vous allez mettre en place cette fonctionnalité sur votre site personnel. Pour gérer cette liste de liens, vous

devez implémenter plusieurs fonctions : l'affichage de la liste en question, sa sauvegarde et son chargement.

Sérialisation XML

Une liste de liens contient généralement peu de liens, une cinquantaine maximum. Pour les gérer, nul besoin de base de données. Un simple fichier fera l'affaire, en l'occurrence un fichier XML généré automatiquement grâce à un mécanisme appelé "sérialisation". La sérialisation permet de sauvegarder un objet, plus précisément de le faire persister à un instant t et de le recharger plus tard, par exemple lors d'une utilisation ultérieure de l'application. Le Framework .NET propose deux types de sérialisations : la sérialisation binaire, qui stocke l'information dans un fichier binaire et donc non lisible avec un éditeur de texte, et la sérialisation XML, qui permet de générer un fichier XML contenant l'objet sérialisé.

Créez une classe `Lien` qui permettra de représenter et de stocker un lien en mémoire.

Pour cela, affichez le menu contextuel du projet et sélectionnez l'élément **Ajouter une nouvelle classe**. Nommez cette classe `lien`, et créez deux propriétés renvoyant des objets de type `String` : `Nom` et `Url`. Elles serviront d'accesseur aux champs privés correspondants. Vous devez arriver à un résultat proche de celui-ci :

```
Public Class Lien

    Private m_nom As String = String.Empty
    Private m_url As String = String.Empty

    Public Property Nom() As String
        Get
            Return m_nom
        End Get
        Set(ByVal value As String)
            m_nom = value
        End Set
    End Property

    Public Property URL() As String
        Get
            Return m_url
        End Get
        Set(ByVal value As String)
            m_url = value
        End Set
    End Property
End Class
```

Comme vous souhaitez gérer une liste de liens, et non pas un lien unique, créez une nouvelle classe que vous nommerez `Liens`. Elle contiendra une propriété `Liens` qui renvoie une liste de `Lien`.

```
Public Class Liens

    Private Shared m_liens As List(Of Lien)

    Public Shared Property Liens() As List(Of Lien)
        Get
            If m_liens Is Nothing Then
                m_liens = ChargerLiens()
            End If
            Return m_liens
        End Get
        Set(ByVal value As List(Of Lien))
            m_liens = value
        End Set
    End Property
End Class
```

Maintenant que vous pouvez stocker en mémoire une liste de liens, sauvegardez cette liste via le mécanisme de sérialisation XML.

Pour ce faire, vous devez utiliser trois espaces de noms : `System.IO` pour les objets liés à la manipulation de fichiers, `System.Xml` pour les objets liés à la manipulation de documents XML, et `System.Xml.Serialization` pour les objets dédiés à la sérialisation XML. La sérialisation se fait à l'aide d'un objet `XmlSerializer`. Cet objet devant écrire dans un fichier, il est nécessaire de le lier à un autre objet doté de cette capacité d'écriture, tel que `TextWriter`, qui permet d'écrire des documents de type texte :

```
    Public Shared Sub SauverLiens()
        Dim ser As XmlSerializer =
        New XmlSerializer(GetType(List(Of Lien)))
        Dim writer As TextWriter =
        New StreamWriter("links.xml")
        ser.Serialize(writer, m_liens)
        writer.Close()

    End Sub
```

Il ne reste plus qu'à mettre en place la désérialisation de l'objet pour charger la liste de liens sauvegardée. Ce mécanisme inverse de la sérialisation se fait de manière analogue, si ce n'est que, au lieu d'utiliser un `TextWriter` pour écrire un fichier, il faut utiliser un objet `TextReader` pour le lire :

```
    Public Shared Function ChargerLiens() As List(Of Lien)
        Dim returnLiens As List(Of Lien) = _
        New List(Of Lien)
        Dim deser As XmlSerializer = _
        New XmlSerializer(GetType(List(Of Lien)))
        Dim reader As TextReader
        reader = New StreamReader("links.xml")
        returnLiens = deser.Deserialize(reader)
        reader.Close()
        Return returnLiens
    End Function
```

Affichage des liens

Ajoutez une nouvelle page *Links.aspx*, qui contiendra les contrôles nécessaires à la gestion des liens. Pour ce faire, affichez le menu contextuel du projet et cliquez sur **Ajouter un nouvel élément**. Dans la boîte de dialogue qui s'ouvre, nommez votre fichier *Links.aspx*. Vérifiez que la case à cocher *Sélectionner la page maître* est cochée. Cela vous permettra de sélectionner la page maître que vous avez créée précédemment.

Pour afficher la liste de liens, vous allez utiliser un contrôle source de données qui permet de remplir des contrôles d'affichage avec une liste ou une collection d'objets. Ce contrôle non visuel s'appelle `ObjectDataSource`. Il permet d'afficher des données venant d'une liste d'objets directement dans un contrôle d'affichage tel que `GridView`. Il a besoin au minimum d'une méthode lui renvoyant cette liste d'objets.

Vous devez donc ajouter dans la classe `Liens` une méthode qui renvoie la liste de `Lien` :

```
    Public Shared Function GetLiens() As List(Of Lien)
        Return Liens
    End Function
```

Cela fait, placez sur votre formulaire le contrôle `ObjectDataSource`. Il s'affiche sous la forme d'un rectangle gris, non visible à l'exécution de la page.

Affichez le smart tag lié au contrôle et cliquez sur le lien *Configurer la source de données*.

Un nouvel Assistant apparaît, permettant de sélectionner la classe utilisée. Décochez la case *Afficher uniquement les composants*, et sélectionnez la classe `Liens` créée précédemment.

Cliquez sur le bouton **Suivant** pour passer à la prochaine étape. Vous devez sélectionner les méthodes de la classe `Liens` qui seront appelées par le contrôle `ObjectDataSource` lorsque l'utilisateur souhaitera afficher les données (onglet

Select), les modifier (onglet **Update**), les insérer (onglet **Insert**) ou encore les supprimer (onglet **Delete**).

Remarque

Classe ou méthode absente de la liste

Lors de la sélection de la classe `Liens` ou de la méthode `GetLiens` dans l'Assistant de configuration du contrôle `ObjectDataSource`, la classe ou la méthode risque d'être absente de la liste. Si tel est le cas, recompilez l'application, par exemple à l'aide de la combinaison de touches Ctrl+Maj+B, et relancez l'Assistant.

L'affichage des données est votre unique objectif pour le moment. Sous l'onglet **Select**, sélectionnez la méthode `GetLiens` que vous venez d'ajouter à la classe `Liens`.

Cliquez sur le bouton **Finish** pour valider. Vous avez à présent configuré le contrôle d'accès aux données, `ObjectDataSource`. Il ne reste plus qu'à afficher les données.

Pour cela, placez un contrôle `GridView` sur votre formulaire. Ensuite, cliquez sur son smart tag puis, dans la liste déroulante *Choisir la source de données*, sélectionnez le contrôle `ObjectDataSource` que vous avez ajouté à la page web. Dès la sélection de la source de données, Visual Web Developer modifie l'affichage du contrôle `GridView` et affiche automatiquement deux colonnes, une pour le nom du lien et une autre pour son URL.

Ces colonnes ne conviennent pas. En effet, il serait plus pratique de présenter aux visiteurs une seule colonne, avec des liens ayant comme texte le nom du lien et comme URL la propriété `Url` de l'objet `Lien`.

Pour arriver à ce résultat, cliquez sur le smart tag du contrôle, puis sur le lien *Modifier les colonnes*. Une boîte de dialogue s'affiche permettant de manipuler les différentes colonnes du `GridView`. Supprimez les colonnes existantes à l'aide du bouton ayant une croix rouge comme icône, et créez-en une nouvelle de type `HyperLinkField`. Ce type de colonne permet de créer une liste de liens. Définissez le texte affiché en en-tête de la colonne grâce à la propriété `HeaderText`. Définissez les données qui vont être affichées grâce à deux propriétés : `DataNavigateUrlFields` et `DataTextField`. La première permet d'indiquer le champ utilisé pour définir le lien, c'est-à-dire l'URL vers laquelle le lien va pointer. Définissez sa valeur à `Url` puisqu'il s'agit du nom de la propriété de la classe `Lien` qu'il faut utiliser. Concernant la propriété `DataTextField`, définissez sa valeur à `Nom` puisqu'il s'agit du nom de la propriété de la classe `Lien` qui contient le nom du lien.

Ajout des liens

La procédure d'ajout de liens est assez triviale à réaliser. Ajoutez deux contrôles `TextBox` à la page web, un pour la saisie du nom et un second pour la saisie de l'URL, ainsi qu'un bouton.

Dans l'événement `Click` du bouton, ajoutez le lien à la collection de liens présente en mémoire, puis rafraîchissez l'affichage du contrôle `GridView` :

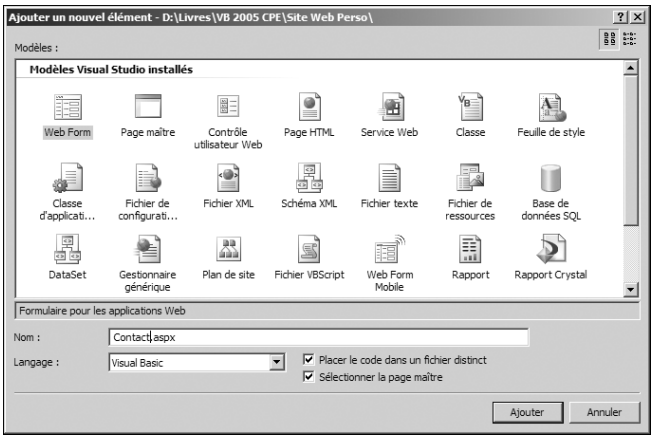
```
Protected Sub Button1_Click(ByVal sender As Object,_
 ByVal e As System.EventArgs) Handles Button1.Click
        Dim nouveauLien As Lien = New Lien
        nouveauLien.Nom = txtUrl.Text
        nouveauLien.URL = txtNom.Text

        Liens.Liens.Add(nouveauLien)
        GridView1.DataBind()
        Liens.SauverLiens()
    End Sub
```

Enregistrez, puis compilez et affichez le résultat grâce à la touche F5.

6.4 Page Contact

Passons à présent à la page **Contact**. Elle permettre à votre visiteur de prendre contact avec vous grâce à l'envoi d'un courrier électronique.

Pour créer cette page **Contact**, utilisez la commande **Ajouter un nouvel élément** et ajoutez une WebForm. Vérifiez que le langage sélectionné est Visual Basic et que la case *Sélectionner la page maître* est cochée.

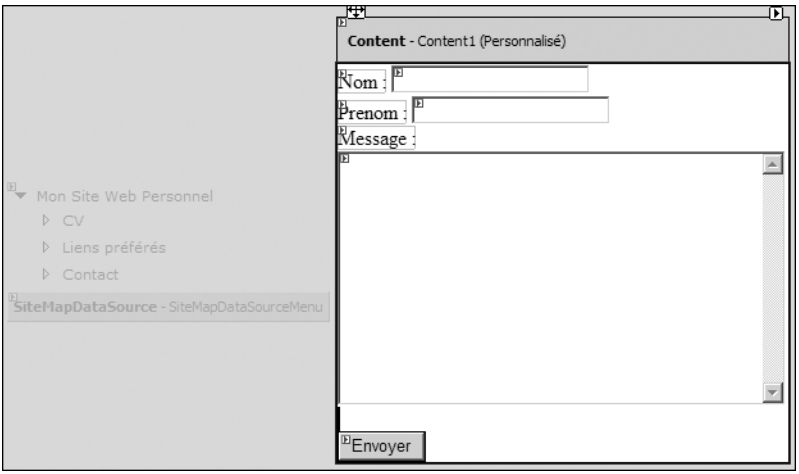


▲ **Figure 6-6 :** Ajout d'une page web

Validez. Sélectionnez la page maître créée précédemment pour l'appliquer à votre page **Contact**.

Ajoutez trois `TextBox` dans votre page *Contact.aspx*, la première ayant un `ID` égal à `txtNom`, la deuxième un `ID` égal à `txtMail`, et la dernière un `ID` égal à `txtMessage`. Modifiez également la propriété `TextMode` de cette dernière `TextBox` pour définir sa valeur à `Multiline`.

Ajoutez quelques contrôles `Label` pour décrire les différents champs de saisie, ainsi qu'un bouton permettant d'envoyer le message.



▲ **Figure 6-7** : Interface de la page Contact

Il faut à présent écrire un peu de code pour que le visiteur puisse envoyer son e-mail lorsqu'il clique sur le bouton **Envoyer**.

Les objets liés à l'envoi d'e-mails se situent dans l'espace de noms `System.Net.Mail`. Il est donc utile d'écrire un `Imports System.Net.Mail` en début de classe pour accéder directement à tous ces objets.

Écrivez le code suivant dans l'événement `Click` du bouton **Envoyer** :

```
Protected Sub btnEnvoyer_Click(ByVal sender As Object,_
 ByVal e As System.EventArgs) Handles btnEnvoyer.Click
Dim mail As New MailMessage()
mail.From = _
New MailAddress("adressemail@fai.fr", "Contact")
mail.To.Add(New MailAddress("monadressemail@monfai.fr"))

mail.Body = "Mail envoyé par " & txtNom.Text & vbCrLf
mail.Body &= "Adresse mail : " & txtMail.Text & vbCrLf
mail.Body &= "Message :" & vbCrLf
```

```
mail.Body &= txtMessage.Text

Dim smtp As New SmtpClient
smtp.Host = "smtp.wanadoo.fr"
smtp.Credentials = _
New Net.NetworkCredential("userName", "password")
smtp.Send(mail) 'envoi du message

End Sub
```

L'application est à présent terminée. Il ne vous reste plus qu'à créer une page contenant votre CV par exemple, pour compléter le site web.

6.5 Check-list

Dans ce chapitre, vous avez appris à :

- utiliser une page maître au sein de votre site web personnel ;
- créer un menu grâce à un fichier "sitemap" et à un contrôle `Treeview` ;
- sauvegarder des objets grâce à la sérialisation XML ;
- utiliser le contrôle `ObjectDataSource` pour remplir un contrôle d'affichage des données avec des objets ;
- envoyer des e-mails.

Site web familial

Dans ce chapitre, vous allez créer un site web personnel qui vous permettra de présenter votre famille ainsi que les photos de vos vacances.

Vous utiliserez des composants clés d'ASP .NET 2.0, comme les pages maîtres, les thèmes et contrôles liés aux données.

7.1 Classes et espaces de noms utilisés

Il s'agit ici, une fois de plus, d'une application web. Vous devrez donc utiliser des clases dans l'espace de noms `System.Web`.

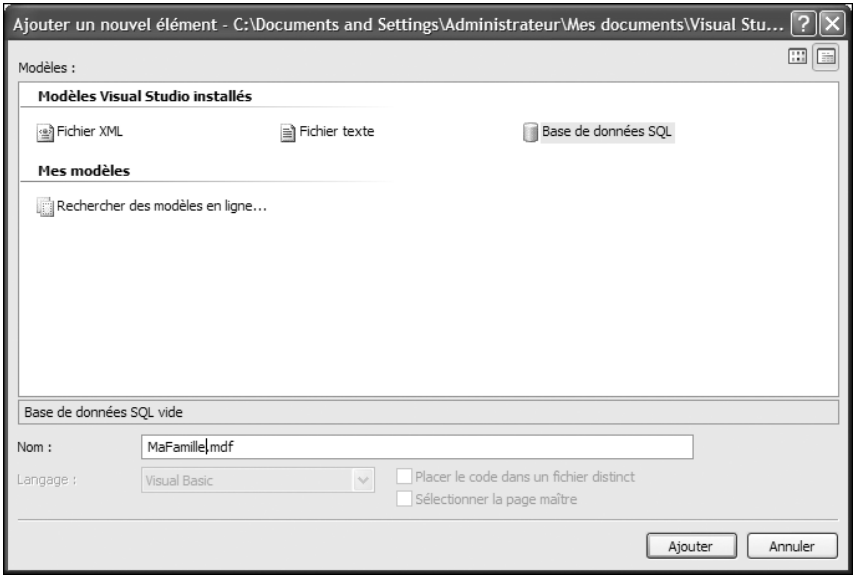
De plus, vous manipulerez des fichiers avec les classes qui se trouvent dans `System.IO`, et vous vous connecterez à votre base de données grâce à l'espace de noms `System.Data.SqlClient`.

7.2 Accès aux données

Vous aurez besoin d'une base de données pour stocker les photos de l'album que vous présenterez sur votre site web. Pour cela, créez votre nouveau site web dans Visual Web Developer 2005 Express.

◄ **Figure 7-1** : Création d'une nouvelle base de données pour un site web ASP .NET

Sous le nom de votre site figure un dossier *App_Data* dans lequel vous stockerez votre base de données. Cliquez du bouton droit sur ce dossier et sélectionnez la commande **Ajouter un nouvel élément**. Créez une base de données SQL que vous appellerez `MaFamille.mdf`.



▲ **Figure 7-2 :** La nouvelle base de données

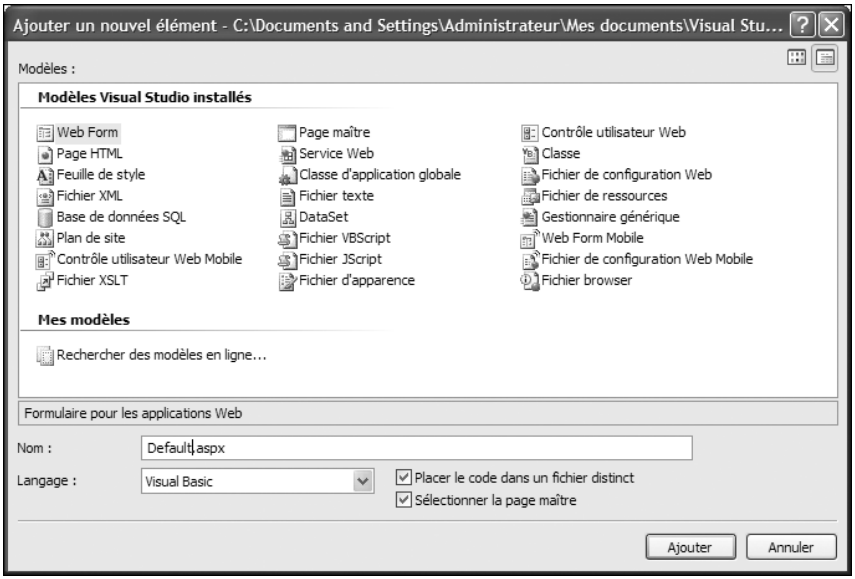
Dans l'Explorateur de bases de données qui s'ouvre, cliquez du bouton droit sur le dossier *Tables* pour créer une nouvelle table dans la base. Créez la table `Photos` en vous basant sur la figure suivante :

Nom de la colonne	Type de données	Null autorisé
PhotoID	int	☐
Titre	nvarchar(50)	☐
Description	nvarchar(MAX)	☐
Photo	image	☐

◀ **Figure 7-3 :** Structure de la table Photos

7.3 Interface utilisateur

Votre site web consistera en deux pages : la page principale, *Default.aspx*, qui servira à présenter votre famille, et l'album de vos photos, *Album.aspx*. Vous pourrez par la suite, ajouter d'autres pages si vous le souhaitez. Pour le moment, pour assurer un style uniforme dans toutes les pages du site, vous allez créer une page maître : cliquez du bouton droit sur le nom de votre site web, sélectionnez **Ajouter un nouvel élément** et puis le modèle *Page maître*.



▲ **Figure 7-4 :** Création d'une page maître

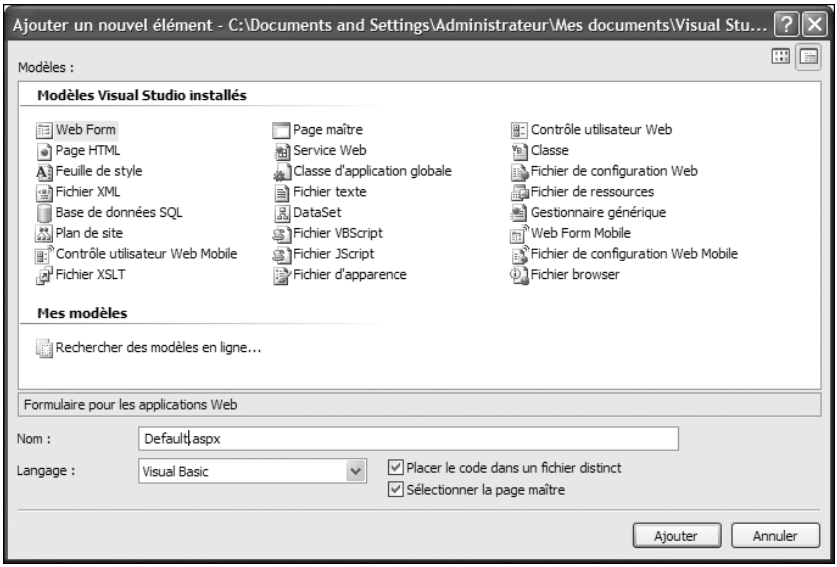
Votre page maître servira de modèle pour toutes les autres pages de votre site. Elle contient un `ContentPlaceHolder` par défaut, qui correspond à la région qui pourra être modifiée dans les autres pages. Remplissez votre page maître avec le texte que vous voulez, en dehors du `ContentPlaceHolder`. Par exemple, écrivez le nom de votre famille comme titre de la page. Ne définissez pas les couleurs ou l'emplacement des éléments. Cela sera fait grâce aux thèmes que vous préparerez par la suite.

Enfin, supprimez le fichier *Default.aspx* qui a été créé par défaut. En effet, celui-ci ne tient pas compte de la page maître que vous venez de construire. Vous allez donc récréer un fichier qui aura le même nom, mais qui se basera sur votre page maître.

Ajoutez un nouvel élément à votre site web, comme vous l'avez fait pour la page maître, mais sélectionnez le modèle *Web Form* (voir Figure 7-5).

Les cases *Placer le code dans un fichier distinct* et *Sélectionner la page maître* doivent être cochées. La première permet de séparer le code Visual Basic du code ASP .NET, ce qui apporte de la clarté, et la deuxième vous oblige à sélectionner la page maître que vous venez de créer comme modèle de votre nouveau formulaire.

De la même manière, créez un deuxième formulaire que vous appellerez *Album.aspx*.



▲ **Figure 7-5 :** Création d'un formulaire web basé sur une page maître

Chaque page basée sur une page maître contient un contrôle `Content`, qui correspond au `ContentPlaceHolder` de celle-ci. Dans le contrôle `Content` de la page *Default.aspx*, présentez votre famille et créez un lien vers l'album de photos.

Votre interface graphique est presque terminée. Vous allez la compléter dans la prochaine section en écrivant du code ASP .NET.

7.4 Réalisation

Vous avez, pour l'instant, deux formulaires web vides, qui se basent sur une page maître. Cependant, aucune de ces pages n'a de mise en forme ou de mise en page, car cela sera fait à l'aide de thèmes. Vous allez, d'ailleurs, créer un thème pour votre site web tout de suite.

Associer un thème au site

Tout d'abord, créez un dossier ASP .NET qui sera dédié aux thèmes. Cliquez du bouton droit sur le nom de votre projet et sélectionnez, dans le sous-menu **Ajouter le dossier ASP .NET**, **Thème**. Appelez votre premier thème `Theme1`. Dans ce dossier, vous stockerez des images de fond et des feuilles de style CSS qui seront appliquées à vos formulaires, dès que vous aurez spécifié ce thème comme celui par défaut.

Pour ce faire, ouvrez le fichier *Web.config* et cherchez la section `pages`. Ajoutez dans cette balise un attribut `theme` auquel vous affecterez comme valeur le nom de votre thème.

Source de données

Avant de commencer votre album de photos, glissez un objet `SqlDataSource` dans votre formulaire. Il servira de source de données pour le contrôle `FormView` qui gérera votre album. Utilisez la balise active pour configurer votre objet afin qu'il se connecte à votre base de données. Les options par défaut sont appropriées pour les deux premiers écrans de l'Assistant. En revanche, sur l'écran **Configurer l'instruction Select**, en plus de cocher la case à côté de l'astérisque pour que l'objet `SqlDataSource` récupère toutes les données de la table, cliquez sur le bouton **Options avancées** pour spécifier que les insertions et les mises à jour devront aussi être gérées. Il suffit pour cela de cocher les deux cases de la fenêtre qui s'ouvre.

Malheureusement, comme votre base de données utilise le type `image` pour la colonne qui stocke les photos, la configuration créée par l'Assistant ne convient pas complètement à votre application. Vous devez donc modifier le code ASP .NET à la main pour supprimer les références aux paramètres correspondant à la colonne `Photo` :

```
<asp:SqlDataSource ID="SqlDataSource" runat="server"
    ConflictDetection="CompareAllValues"
    ConnectionString="<%$ ConnectionStrings:ConnectionString %>"
    DeleteCommand="DELETE FROM [Photos]
    WHERE [PhotoID] = @original_PhotoID"
    InsertCommand="INSERT INTO [Photos]
    ([Titre], [Description], [Photo])
    VALUES (@Titre, @Description, @Photo)"
    OldValuesParameterFormatString="original_{0}"
    SelectCommand="SELECT * FROM [Photos]"
    UpdateCommand="UPDATE [Photos]
    SET [Titre] = @Titre, [Description] = @Description
    WHERE [PhotoID] = @original_PhotoID">

    <DeleteParameters>
      <asp:Parameter Name="original_PhotoID" Type="Int32" />
    </DeleteParameters>
    <UpdateParameters>
      <asp:Parameter Name="Titre" Type="String" />
      <asp:Parameter Name="Description" Type="String" />
      <asp:Parameter Name="original_PhotoID" Type="Int32" />
    </UpdateParameters>
    <InsertParameters>
      <asp:Parameter Name="Titre" Type="String" />
      <asp:Parameter Name="Description" Type="String" />
```

```
    </InsertParameters>
</asp:SqlDataSource>
```

▲ SqlDataSource modifié sans paramètre pour la colonne Photo

Étant donné que vous avez supprimé les paramètres correspondant à la colonne `Photo`, vous ne pouvez pas stocker les images dans votre base de données, à moins de gérer les événements `Inserting` et `Updating` de `SqlDataSource`.

```
Protected Sub SqlDataSource_Inserting( _
  ByVal sender As Object, _
  ByVal e As SqlDataSourceCommandEventArgs) _
  Handles SqlDataSource.Inserting
    Dim stateTextBox As FileUpload = _
      FormView1.FindControl("FichierPhoto")
    Dim inputStream As Stream = _
      stateTextBox.PostedFile.InputStream
    Dim imageLength As Integer = _
      stateTextBox.PostedFile.ContentLength
    Dim imageBinary(imageLength) As Byte
    Dim inputRead As Integer = _
      inputStream.Read(imageBinary, 0, imageLength)
    Dim imageData() As Byte = imageBinary
    Dim param As New SqlParameter("@photo", _
      System.Data.SqlDbType.Image)
    param.Value = imageData
    e.Command.Parameters.Add(param)
End Sub

Protected Sub SqlDataSource_Updating( _
  ByVal sender As Object, _
  ByVal e As SqlDataSourceCommandEventArgs) –
  Handles SqlDataSource.Updating
    Dim stateTextBox As FileUpload = _
      FormView1.FindControl("FichierPhoto")
    If stateTextBox.FileName.Length > 0 Then
        e.Command.CommandText = "UPDATE [Photos] _
    SET [Titre] = @Titre, [Description] = @Description, _
    [Photo] = @Photo WHERE [PhotoID] = @original_PhotoID"
        Dim inputStream As Stream = _
          stateTextBox.PostedFile.InputStream
        Dim imageLength As Integer = _
          stateTextBox.PostedFile.ContentLength
        Dim imageBinary(imageLength) As Byte
        Dim inputRead As Integer = _
          inputStream.Read(imageBinary, 0, imageLength)
        Dim imageData() As Byte = imageBinary
        Dim param As New SqlParameter("@photo", _
          System.Data.SqlDbType.Image)
        param.Value = imageData
```

```
        e.Command.Parameters.Add(param)
    End If
End Sub
```

▲ Gestionnaires des événements Inserting et Updating du contrôle SqlDataSource

Dans ces gestionnaires, on crée le paramètre correspondant à la colonne `Photo`. On utilise, comme valeur de ce paramètre, l'image qui se trouve dans le champ `FichierPhoto` du contrôle `FormView` que l'on créera par la suite.

Vous allez maintenant créer le contrôle `FormView`, une nouveauté d'ASP .NET 2.0, qui gérera votre album photos.

Album de photos

Déposez un contrôle de type `FormView` dans votre fichier *Album.aspx* pour la gestion de votre album de photos. Utilisez la balise active de `FormView` pour spécifier le contrôle `SqlDataSource` comme source de données de celui-ci et pour activer la pagination.

Modifiez les modèles du contrôle `FormView` correspondant à l'édition des données, à leur insertion et à la page qui sera affichée si la source de données est vide. Voici un exemple de contrôle `FormView` pour la gestion de l'album de photos :

```
<asp:FormView ID="FormView1" runat="server"
    AllowPaging="True" DataKeyNames="PhotoID"
    DataSourceID="SqlDataSource">
    <EditItemTemplate>
        Titre:
        <br />
        <asp:TextBox ID="TitreTextBox" runat="server"
            Text='<%# Bind("Titre") %>'></asp:TextBox><br />
        Description:
        <br />
        <asp:TextBox ID="DescriptionTextBox" runat="server"
            Text='<%# Bind("Description") %>'
            Rows="5" TextMode="MultiLine"></asp:TextBox>
        <br />
        Photo:
        <br />
        <asp:FileUpload ID="FichierPhoto" runat="server" />
        <br />
        <asp:LinkButton ID="UpdateButton" runat="server"
            CausesValidation="True" CommandName="Update"
            Text="Mettre à jour">
        </asp:LinkButton>
        <asp:LinkButton ID="UpdateCancelButton"
            runat="server" CausesValidation="False"
```

```
            CommandName="Cancel" Text="Annuler">
        </asp:LinkButton>
    </EditItemTemplate>

    <InsertItemTemplate>
        Titre:
        <br />
        <asp:TextBox ID="TitreTextBox" runat="server"
            Text='<%# Bind("Titre") %>'></asp:TextBox><br />
        Description:
        <br />
        <asp:TextBox ID="DescriptionTextBox" runat="server"
            Text='<%# Bind("Description") %>'
            Rows="4" TextMode="MultiLine"></asp:TextBox>
        <br />
        Photo:
        <br />
        <asp:FileUpload ID="FichierPhoto" runat="server" />
        <br />
        <asp:LinkButton ID="InsertButton" runat="server"
            CausesValidation="True" CommandName="Insert"
            Text="Insérer">
        </asp:LinkButton>
        <asp:LinkButton ID="InsertCancelButton"
            runat="server" CausesValidation="False"
            CommandName="Cancel" Text="Annuler">
        </asp:LinkButton>
    </InsertItemTemplate>
    <ItemTemplate>
        <h2>
            <asp:Label ID="TitreLabel" runat="server"
                Text='<%# Bind("Titre") %>'></asp:Label>
        </h2>
        <asp:Image ID="Image2" runat="server"
        ImageUrl='<%# Eval("PhotoID", "Photo.ashx?id={0}") %>' />
        <br />
        <asp:Label ID="DescriptionLabel" runat="server"
            Text='<%# Bind("Description") %>'></asp:Label>
        <p>
            <asp:LinkButton ID="EditButton" runat="server"
                CausesValidation="False" CommandName="Edit"
                Text="Modifier"></asp:LinkButton>
            <asp:LinkButton ID="DeleteButton" runat="server"
                CausesValidation="False"
                CommandName="Delete" Text="Supprimer">
            </asp:LinkButton>
            <asp:LinkButton ID="NewButton" runat="server"
                CausesValidation="False" CommandName="New"
                Text="Nouveau"></asp:LinkButton>
        </p>
```

```
    </ItemTemplate>

    <EmptyDataTemplate>
        <p>L'album photos est vide, cliquez
            <asp:LinkButton ID="NewButton" runat="server"
                CausesValidation="False" CommandName="New"
                Text="ici"></asp:LinkButton>
            pour ajouter une photo.</p>
    </EmptyDataTemplate>
</asp:FormView>
```

▲ Exemple de contrôle FormView pour la gestion de l'album de photos

Vous pouvez personnaliser vos modèles à votre guise. La seule contrainte vitale à respecter est le nom des contrôles `FileUpload`. En effet, il faut absolument que celui-ci corresponde au nom utilisé dans les gestionnaires d'événements de l'objet `SqlDataSource`.

Tous vos contrôles sont directement liés à votre source de données, à l'exception de celui qui affiche l'image stockée dans la base de données. En effet, cette liaison n'est pas possible et vous devrez utiliser un gestionnaire générique qui retrouvera les images dans la base.

Gestionnaire générique

Ajoutez un fichier de type *Gestionnaire générique* à votre projet. Pour être conforme aux modèles de l'exemple de contrôle `FormView` précédent, appelez-le *Photo.ashx*.

Un gestionnaire générique est une simple classe capable de répondre à une requête HTTP. En effet, la méthode `ProcessRequest` du gestionnaire sera exécutée à chaque appel de celui-ci et son résultat sera renvoyé au client qui a fait la requête. Vous allez donc utiliser l'identifiant de la photo que vous souhaitez afficher pour récupérer celle-ci dans la base de données et la renvoyer au client.

Voici un gestionnaire générique qui cherche l'image demandée dans la base de données et la renvoie au client :

```
Public Class Photo : Implements IHttpHandler

    Public Sub ProcessRequest( _
      ByVal context As HttpContext) _
      Implements IHttpHandler.ProcessRequest
        context.Response.ContentType = "image/jpeg"
        context.Response.Cache.SetCacheability( _
          HttpCacheability.Public)
        context.Response.BufferOutput = False
        Dim id As Integer = -1
```

```
        Dim stream As Stream
        id = Convert.ToInt32( _
          context.Request.QueryString("id"))
        stream = GetPhoto(id)

        Const buffersize As Integer = 1024 * 16
        Dim buffer(buffersize) As Byte
        Dim count As Integer = stream.Read( _
          buffer, 0, buffersize)
        While (count > 0)
            context.Response.OutputStream.Write( _
              buffer, 0, count)
            count = stream.Read(buffer, 0, buffersize)
        End While
    End Sub

    Private Shared Function GetPhoto( _
      ByVal photoid As Integer) As Stream
        Using connection As New SqlConnection( _
          ConfigurationManager.ConnectionStrings( _
          "ConnectionString").ConnectionString)
            Using myCommand = New SqlCommand( _
              "SELECT [photo] FROM [photos] _
               WHERE ([photoid]=@id)", connection)
                myCommand.Parameters.Add(New SqlParameter( _
                  "@id", photoid))
                connection.Open()
                Dim result As Object = _
                  myCommand.ExecuteScalar()
                Try
                    Return New MemoryStream( _
                      CType(result, Byte()))
                Catch
                    Return Nothing
                End Try
            End Using
        End Using
    End Function

Public ReadOnly Property IsReusable() As Boolean _
      Implements IHttpHandler.IsReusable
        Get
            Return True
        End Get
    End Property

End Class
```

▲ Gestionnaire générique pour récupérer des images stockées dans une base de données

On utilise simplement les objets `SqlConnection` et `SqlCommand` pour se connecter à la base de données et récupérer l'image souhaitée. Celle-ci est transformée en flux `MemeoryStream` et renvoyée au client via la méthode `Response.Output Stream.Write` de l'objet `context`, qui est le seul paramètre de la méthode `ProcessRequest` et qui correspond à la session HTTP courante.

Maintenant que vous pouvez récupérer les images qui se trouvent dans votre base de données, votre site est fonctionnel. Il ne reste plus qu'à personnaliser la page maître et le thème créés précédemment pour que le site illustre au mieux vos vacances en famille.

7.5 Check-list

Dans ce chapitre, vous avez appris à :

- créer un site web avec un style uniforme en utilisant les pages maîtres et les thèmes ;
- stocker des images dans une base de données SQL Server 2005 ;
- utiliser les contrôles liés aux données de Visual Basic 2005 ;
- utiliser un `HTTPHandler` pour récupérer des fichiers stockés dans une base de données.

Site web d'une association

Vous allez créez, dans ce chapitre, le site web d'une association. La France compte près d'un million d'associations, qu'elles soient humanitaires, sportives, religieuse, etc.

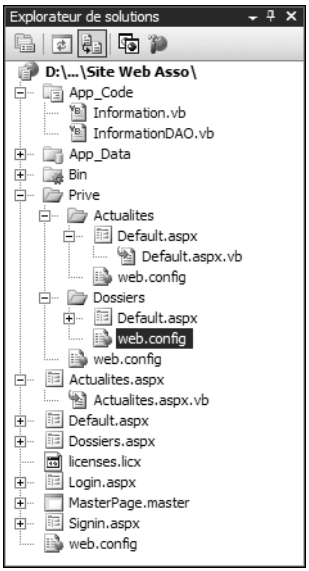
La communication est souvent un outil indispensable à la vie d'une association. Pour présenter les activités proposées, vous allez créer un site permettant de consulter l'actualité de l'association, et des dossiers. Pour vivre, un site doit être dynamique. Vous allez donc inviter les membres du site à saisir eux-mêmes leurs actualités ou leurs dossiers pour qu'ils participent activement à la vie du site.

Cela implique une gestion de la sécurité pour éviter que n'importe qui puisse modifier le site. Vous allez apprendre à mettre en place ce genre de fonctionnalité grâce à ASP .NET 2, Visual Basic 2005 et Visual Web Developer Express.

8.1 Création de la hiérarchie des pages

Le chapitre ***Site web familial*** *explique comment créer une page maître, des pages web utilisant cette page maître et comment créer des menus.*

Pour débuter ce nouveau projet, créez un nouveau site web et ajoutez la hiérarchie de pages suivantes :



◄ **Figure 8-1 :** Hiérarchie des pages

Laissez ces pages vides pour le moment. Vous allez les remplir tout au long de ce chapitre. Vous n'avez qu'à appliquer la page maître créée à toutes les autres pages et le travail sera terminé pour le moment.

8.2 Gestion des informations

Il faut à présent stocker une information en mémoire. Pour cela, créez une classe `Information` qui va définir ce qu'est une information. Celle-ci peut être caractérisée par un identifiant (`id`), un titre, son contenu, sa date de publication, ainsi que son type. Une information peut être une actualité ou un dossier. Il est donc pertinent d'utiliser une énumération pour définir le type d'information :

```
Public Enum TypeInformation
    Actualite = 0
    Dossier = 1
End Enum
```

En programmation orientée objet, les caractéristiques d'un élément sont souvent représentées par des propriétés. Implémentez donc autant de propriétés que de données liées à une information.

Vous devez arriver à un résultat proche de celui-ci :

```
Public Class Information
    Private m_id As Integer
    Private m_titre As String
    Private m_contenu As String
    Private m_typeInformation As TypeInformation
    Private m_datePublication As Date

    Public Property Id() As Integer
        Get
            Return m_id
        End Get
        Set(ByVal value As Integer)
            m_id = value
        End Set
    End Property

    Public Property Titre() As String
        Get
            Return m_titre
        End Get
        Set(ByVal value As String)
            m_titre = value
        End Set
    End Property
```

```
    Public Property Contenu() As String
        Get
            Return m_contenu
        End Get
        Set(ByVal value As String)
            m_contenu = value
        End Set
    End Property

    Public Property Type() As TypeInformation
        Get
            Return m_typeInformation
        End Get
        Set(ByVal value As TypeInformation)
            m_typeInformation = value
        End Set
    End Property

    Public Property DatePublication() As Date
        Get
            Return m_datePublication
        End Get
        Set(ByVal value As Date)
            m_datePublication = value
        End Set
    End Property
End Class
```

Création de la base de données

La gestion des informations consiste à enregistrer des données dans la base. Pour cela, vous allez utiliser SQL Server Express Edition. Démarrez l'application SQL Management Studio Express Edition et créez une nouvelle base de données que vous appellerez *SiteAssoc*.

Dans cette base, ajoutez une table nommée *Informations* et créez-la selon le schéma suivant :

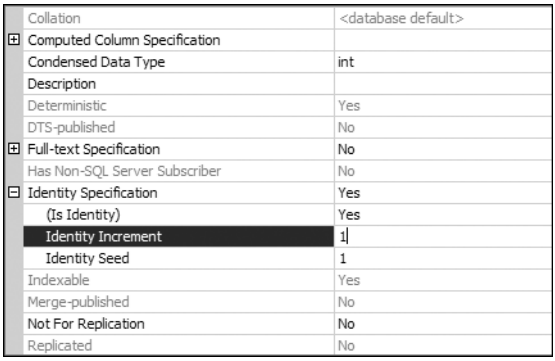
Table - dbo.Informations | Summary

Column Name	Data Type	Allow Nulls
id_information	int	☐
titre_information	varchar(50)	☐
contenu_information	varchar(MAX)	☐
date_information	nchar(10)	☐
type_information	tinyint	☐
		☐

◄ **Figure 8-2 :** Schéma de la table Informations

Prêtez attention aux points suivants :

- Le champ `id_information` doit être défini en tant que clé primaire. Pour cela, cliquez du bouton droit sur la ligne correspondant au champ et sélectionnez **Créer clé primaire** dans le menu contextuel.
- Le champ `id_information` doit être défini en tant que champ en auto-incrémentation. Cela vous évitera d'incrémenter manuellement la valeur du champ à chaque insertion d'une information. L'opération sera effectuée automatiquement par SQL Server. Pour cela, dans la zone des propriétés du champ, sélectionnez la propriété *Identity Specification* et définissez la propriété *IsIdentity* à *Yes* et la propriété *Identity Increment* à 1.



◄ **Figure 8-3 :** Auto-incrémentation d'un champ

Voici le script SQL permettant de créer directement la table *Informations* :

```
CREATE TABLE [dbo].[Informations](
  [id_information] [int] IDENTITY(1,1) NOT NULL,
  [titre_information] [varchar](50)
COLLATE French_CI_AS NOT NULL,
  [contenu_information] [varchar](max)
COLLATE French_CI_AS NOT NULL,
  [date_information] [nchar](10)
COLLATE French_CI_AS NOT NULL
CONSTRAINT [DF_Informations_date_information]
 DEFAULT (getdate()),
  [type_information] [tinyint] NOT NULL,
 CONSTRAINT [PK_Informations] PRIMARY KEY CLUSTERED
(
  [id_information] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]

GO
```

Insérer les informations

Il faut à présent implémenter la fonctionnalité d'insertion des informations dans la base de données.

La classe d'accès aux données se nommera `InformationDAO`. Ajoutez donc cette nouvelle classe dans votre projet. Celle-ci se retrouve dans le dossier spécial d'ASP .NET *App_Code*.

Vous utiliserez deux espaces de noms relatifs à la gestion des données en base SQL Server : `System.Data` et `System.Data.SqlClient`. N'oubliez donc pas de placer des instructions d'importation de ces espaces de noms en tout début du fichier contenant votre classe.

Pour insérer des données dans une base de données SQL Server, vous devez utiliser au moins deux objets : un objet `SqlConnection`, qui permet d'établir une connexion à la base, et un objet `SqlCommand`, qui permet d'exécuter une requête SQL.

Vous ouvrirez la connexion en spécifiant une chaîne de connexion qui désigne la base à utiliser ainsi que le mode d'authentification.

Attributs de la chaîne de connexion à définir

Attribut	Description
`server`	Définit l'adresse du serveur ainsi que son instance. Vous pouvez indiquer ici le nom de la machine et le nom d'instance, ou alors l'adresse IP de la machine et le nom d'instance.
`Database`	Nom de la base de données à laquelle se connecter.
`User ;password`	Permet de définir le nom d'utilisateur et le mot de passe à utiliser en cas d'activation du mode d'authentification SQL.
`Integrated Security`	Doit être défini et égal à `SSPI` en cas d'activation du mode d'authentification Windows.

Ouvrez la connexion dans un bloc `Try… Catch` car cette ouverture peut échouer en cas d'indisponibilité du serveur par exemple.

Ajoutez les paramètres d'entrée (et même de sortie) à l'objet `SqlCommand` en utilisant la méthode `Add` de la collection `Parameters` de celui-ci, puis en définissant le nom du paramètre à ajouter définir et sa valeur :

```
Imports System.Data
Imports System.Data.SqlClient
Public Class InformationDAO
```

```
    Public Shared Sub CreateInformation(ByVal
        information As Information)
        Dim connection As New SqlConnection
("server=.\SQLEXPRESS;database=siteassoc;integrated security=SSPI")
        Try
connection.Open()
Dim cmdCreate As SqlCommand
cmdCreate = connection.CreateCommand
cmdCreate.CommandText = "insert into informations"  & _
"(titre_information, contenu_information, " & _
& "type_information) values (@titre,@contenu,@type)"
With cmdCreate.Parameters
.AddWithValue("@titre", information.Titre)
.AddWithValue("@contenu", information.Contenu)
.AddWithValue("@type", information.Type)
End With
cmdCreate.ExecuteNonQuery()
        Catch ex As Exception
            Throw
        Finally
            connection.Dispose()
        End Try
    End Sub
End Class
```

Insérer des actualités et des dossiers

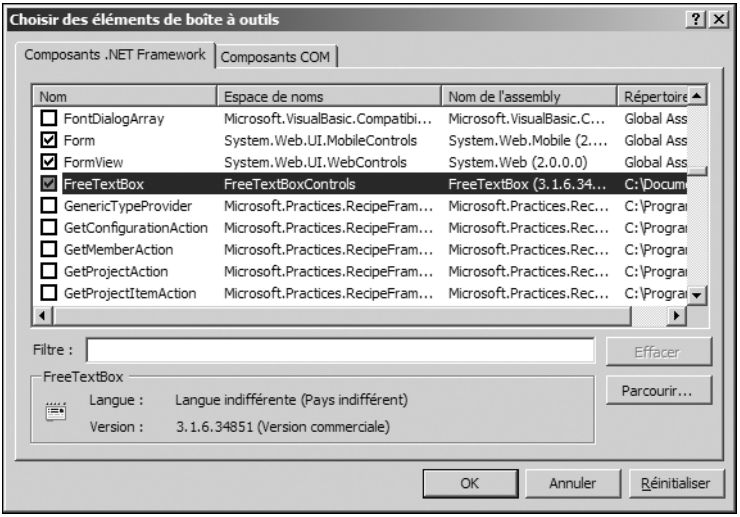
Vous pouvez à présent passer à la réalisation des pages d'édition des actualités et des dossiers situées dans le dossier *Prive*.

La réalisation de ces pages est simple. Pour permettre aux rédacteurs d'actualités et de dossiers de rédiger leurs informations et d'appliquer une mise en page et une mise en forme au texte saisi, il est nécessaire d'utiliser un composant externe à ASP .NET, bien plus riche que le contrôle `TextBox` proposé en standard.

Utilisez par exemple le contrôle `FreeTextBox`, qui est gratuit et disponible en téléchargement à l'adresse **http://freetextbox.com/default.aspx**, sous la forme d'un fichier ZIP.

Pour l'ajouter dans la boîte à outils, dézippez le fichier dans un dossier grâce à un outil de compression/décompression.

Cliquez du bouton droit dans la boîte à outils. Dans le menu contextuel qui s'affiche, sélectionnez **Choisir les éléments**. Dans la boîte de dialogue qui s'affiche, cliquez sur le bouton **Parcourir** pour rechercher le fichier *FreeTextBox.dll* présent dans le dossier *Framework-2.0*.



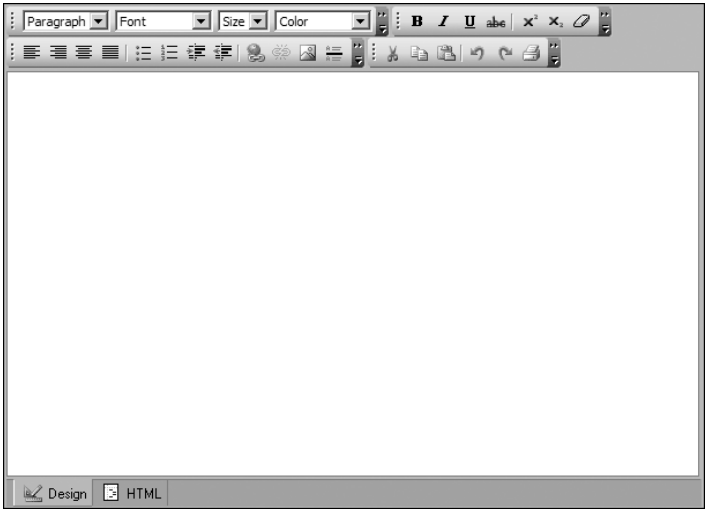
▲ **Figure 8-4 :** Ajout de FreeTextBox dans la boîte à outils

Validez pour afficher le contrôle dans la boîte à outils.

Il ne reste plus qu'à glisser le contrôle sur la page de rédaction des actualités pour pouvoir l'utiliser. Ajoutez également un contrôle `TextBox`, que vous nommerez `txtTitre`, qui permettra de saisir le titre de l'actualité, et un bouton qui permettra d'effectuer l'insertion.

Pour effectuer l'insertion, appelez la méthode `CreateInformation` écrite auparavant, en lui passant en paramètre un objet `Information` :

```
Protected Sub Button1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles Button1.Click
    Dim information As New Information
    information.Contenu = FreeTextBox1.Text
    information.Titre = txtTitre.Text
    information.Type = TypeInformation.Actualite
    InformationDAO.CreateInformation(information)
End Sub
```



▲ **Figure 8-5 :** Le contrôle FreeTextBox dans une page web en cours d'exécution

Remarque

Sécurité ASP .NET

Dès que le rédacteur saisira une information en appliquant une mise en forme (mise en gras, italique…) et cliquera sur le bouton, cela provoquera une erreur "Une valeur Request.Form potentiellement dangereuse a été détectée à partir du client". Cette erreur vient du fait que l'on essaie d'envoyer du code HTML au serveur alors que ASP .NET bloque cette possibilité par défaut pour des raisons de sécurité.

Pour éviter une telle erreur, vous devez ajouter l'attribut `ValidateRequest` dans la directive `Page` en lui attribuant la valeur `false`. Pour cela, affichez le code source HTML de votre page dans Visual Web Developer, et dans la première ligne (où se trouve la directive `Page`), ajoutez l'attribut et sa valeur.

Répétez les mêmes opérations pour la page de rédaction des dossiers, en prenant soin de modifier le type de l'information dans l'événement `Click` du bouton d'insertion pour insérer un dossier, et non une actualité, dans la base.

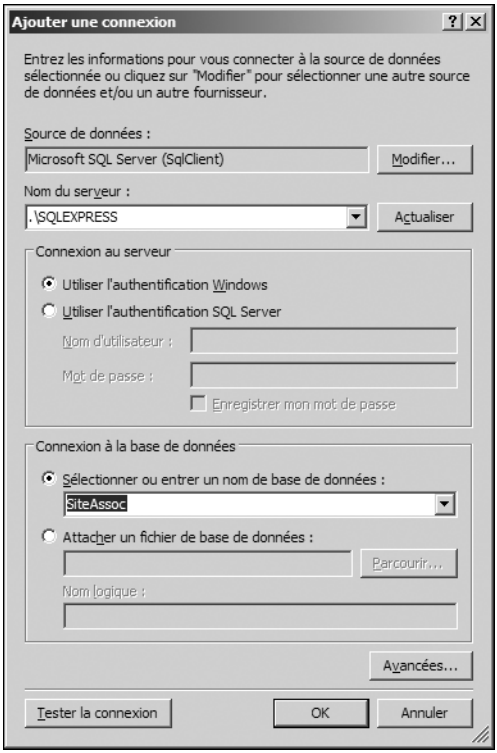
Afficher les actualités et les dossiers

Maintenant qu'il est possible d'ajouter des données dans la base et d'afficher des informations, vous allez créer la page d'affichage des actualités, l'affichage des dossiers pouvant se faire exactement de la même manière.

Placez un contrôle `SqlDataSource` sur la page *Actualites.aspx*. Il s'agit d'un contrôle source de données, qui permet de remplir un contrôle d'affichage avec des données issues d'une base de données, et tout cela sans écrire une ligne de code.

Cliquez sur le smart tag du contrôle puis sur le lien *Configurer la source de données*.

Dans l'Assistant qui s'affiche, cliquez sur le bouton **Nouvelle Connexion** pour créer la chaîne de connexion qui sera utilisée par le contrôle. Sélectionnez une base de type SQL Server puis saisissez les informations fournies dans la chaîne de connexion définie précédemment.

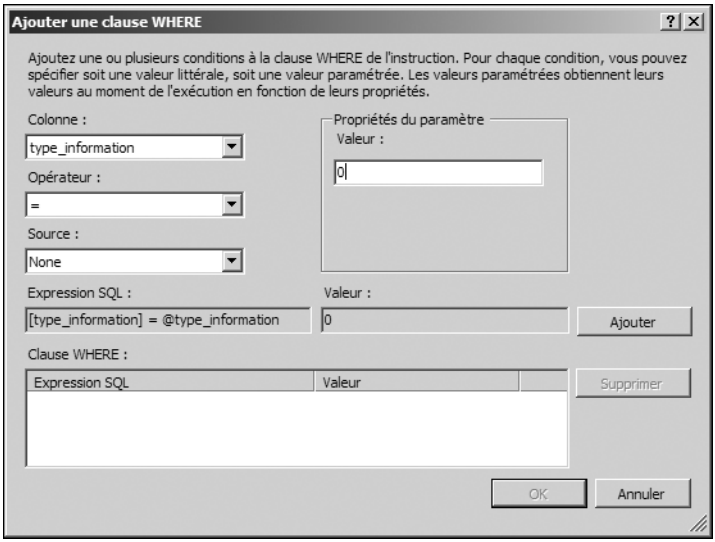


◀ **Figure 8-6 :** Paramètres Connexion

Validez et cliquez sur **Suivant** dans l'Assistant pour sélectionner les données à afficher. La base n'ayant qu'une seule table, la table *Informations* est alors sélectionnée par défaut. Sélectionnez les champs que vous souhaitez afficher. Sélectionnez-les tous, excepté le champ `type_information`.

Vous devez à présent définir un critère de sélection puisque toutes les informations sont stockées dans la table *Informations* : les actualités comme les dossiers.

Pour ce faire, cliquez sur le bouton **Where**, sélectionnez le champ sur lequel portera le critère, c'est-à-dire le champ `type_information`. Dans la zone de sélection *Source*, sélectionnez *None*, et affectez la valeur 0 au paramètre. Cette valeur 0 est la valeur définie par l'énumération `TypeInformation` créée en début de chapitre. Validez et fermez l'Assistant pour afficher les données.



▲ **Figure 8-7 :** Clause Where

Ajoutez à présent un contrôle `DataList` sur votre page, affichez son smart tag pour sélectionner le contrôle `SqlDataSource` qui vient d'être créé comme source de données.

Dès que l'on affecte une source de données au contrôle, celui-ci reconnaît les différents éléments à afficher.

Toujours dans le smart tag, cliquez sur le lien *Mise en forme automatique*, pour sélectionner un style d'affichage plus propre et plus professionnel que celui par défaut.

Vous pouvez maintenant exécuter la page à l'aide de la touche F5 pour vérifier que les actualités s'affichent correctement.

L'affichage des dossiers peut se faire exactement de la même manière. Vous devez simplement modifier la valeur du paramètre de la clause `Where` et la définir à 1 pour récupérer les dossiers, et non les actualités.

8.3 Gestion de la sécurité

ASP .NET 2 propose une gestion de la sécurité simple à mettre en œuvre. Nul besoin d'écrire du code pour implémenter une gestion des membres, des rôles, des autorisations, de l'authentification, vous pouvez tout faire directement, en modifiant quelques paramètres de configuration de l'application, et en utilisant les contrôles web de sécurité, tels que le contrôle de login, le contrôle de création de compte, etc.

Création de la base de données

Créez la base de données qui va contenir les informations nécessaires à la gestion des membres de votre site. Habituellement, cette étape prend du temps, car il est nécessaire de réfléchir au schéma de la base, aux données à stocker, etc.

Pour vous faciliter la vie et mettre en place rapidement une gestion des membres dans une application web, ASP .NET 2 propose un Assistant capable de créer automatiquement une base de données, les différentes tables et procédures stockées nécessaires à une gestion solide des membres.

Pour lancer cet Assistant, exécutez le programme *aspnet_regsql.exe* présent dans le dossier d'installation du Framework .NET 2.

Remarque

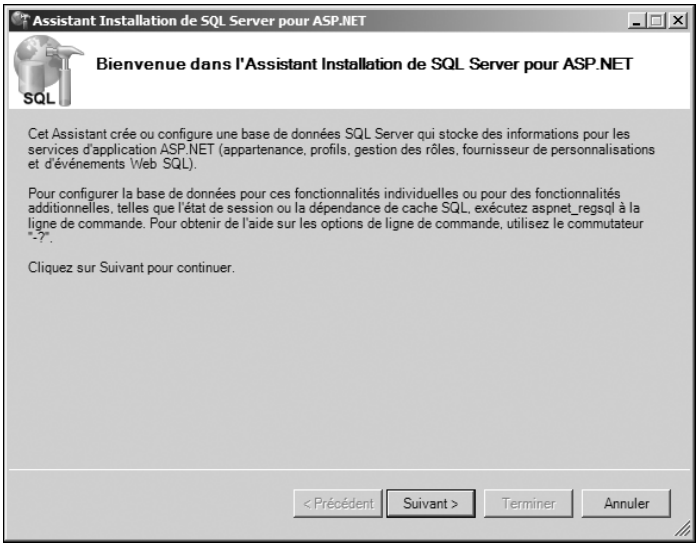
Dossier du Framework .NET 2

Le Framework .NET se situe dans le dossier *Windows/Microsoft.NET/Framework*. Vous y trouverez un dossier par version du Framework .NET installé sur votre machine. Rendez-vous dans le dossier de la version 2. Exemple : *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

Une fois l'application lancée, l'Assistant s'affiche (voir Figure 8-8).

Cliquez sur **Suivant** pour passer à l'étape suivante et sélectionnez l'option *Configurer SQL Server pour les services d'applications*.

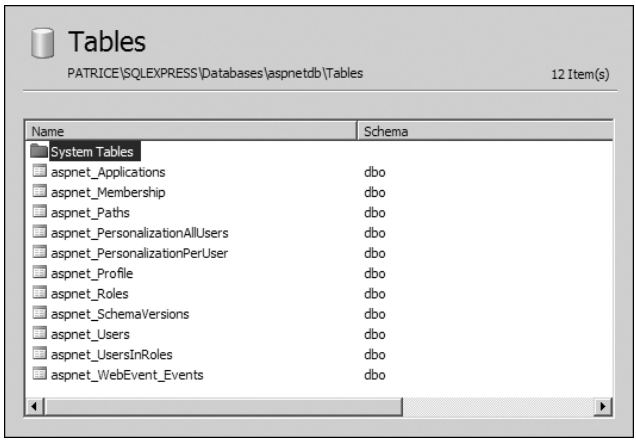
Cliquez de nouveau sur le bouton **Suivant** pour saisir les paramètres de connexion à votre serveur SQL Server. Si votre serveur est installé sur votre poste de développement, vous pouvez saisir `.\SQLEXPRESS` dans le champ *Serveur*, le point désignant la machine locale, et `SQLEXPRESS` désignant l'instance créée lors de l'installation de SQL Server 2005 Express.



▲ **Figure 8-8 :** Assistant de création de base de données de gestion des membres

Cliquez de nouveau plusieurs fois sur le bouton **Suivant** pour valider la création de la base de données. Cette base s'appelle par défaut *aspnetdb* et peut être utilisée par plusieurs applications.

Si vous ouvrez à présent SQL Server Management Studio Express, vous trouverez une base de données *aspnetdb* contenant plusieurs tables, comme *aspnet_Membership*, *aspnet_Users*, *aspnet_UserInRoles*, etc.



▲ **Figure 8-9 :** Tables générées par l'Assistant

Examinez la liste des procédures stockées qui ont été générées et qui accèdent à ces tables (plus d'une cinquantaine !) : l'Assistant a fait une bonne partie du travail. Cette tâche aurait nécessité plusieurs heures si elle avait du être réalisée manuellement.

Configuration de l'application

L'étape suivante consiste à configurer l'application web pour utiliser tout ce qui a été généré.

Pour cela, affichez l'Explorateur de solutions et cliquez sur l'icône d'accès au site d'administration de l'application.

◄ **Figure 8-10 :** Icône d'accès

Cliquez sur l'onglet **Sécurité** puis sur le lien *Sélectionner le type d'authentification*.

Vous arrivez sur une page permettant de sélectionner le type d'accès au site : depuis un réseau local ou depuis Internet.

Sélectionnez l'accès depuis Internet. Cela a pour effet de configurer l'application pour qu'elle utilise une authentification basée sur un formulaire (comme la grande majorité des sites web), et non basée sur les comptes Windows.

Rendez-vous ensuite sous l'onglet **Fournisseur**. Il permet de sélectionner le fournisseur d'accès aux données à utiliser pour accéder aux données relatives à la gestion des membres. Cliquez sur le lien *Sélectionner un fournisseur d'accès unique pour toutes les données de gestion de site* et sélectionnez le fournisseur *AspNetSqlProvider*. Ce fournisseur permet d'utiliser les données générées par l'Assistant précédemment utilisé. Cliquez sur le lien *Test* pour vérifier que la connexion à la base de données se fait correctement et validez.

Création des rôles

Vous allez maintenant créer les rôles utilisateurs de votre application.

Le site web accepte plusieurs types d'utilisateurs :

- Les utilisateurs anonymes, qui viennent consulter le site web pour récupérer des informations sur l'association.
- Les rédacteurs d'actualités, authentifiés sur le site et qui peuvent ajouter des informations sur le site.
- Les rédacteurs de dossiers, authentifiés sur le site et qui peuvent ajouter des dossiers.

Pour créer ces deux rôles, rendez-vous sous l'onglet **Sécurité** du site web d'administration et cliquez sur le lien *Activez les rôles*.

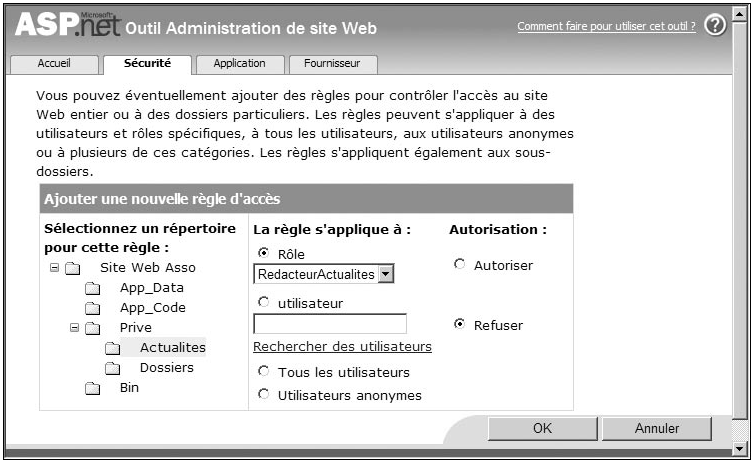
Cliquez ensuite sur le lien *Créer ou Gérer des rôles* pour créer les deux rôles que vous nommerez *RedacteurActualites*, *RedacteurDossiers*.

Création des règles d'accès

Il faut à présent créer les règles d'accès aux différents dossiers du site web. Pour cela, toujours sous l'onglet **Sécurité** du site web d'administration, cliquez sur le lien *Créer des règles d'accès*. Ces règles d'accès correspondent aux autorisations que vous souhaitez affecter aux membres en fonction de leurs rôles :

Liste des règles d'accès à mettre en place		
Dossier	**Rôle**	**Règle**
Prive	Tout le monde	Refuser
Prive\Actualites	*RedacteurActualites*	Autoriser
Prive\Dossiers	*RedacteurDossiers*	Autoriser

La création est intuitive : il suffit de cliquer sur le dossier souhaité, de sélectionner le rôle souhaité ainsi que la règle à créer pour que les autorisations soient automatiquement gérées.

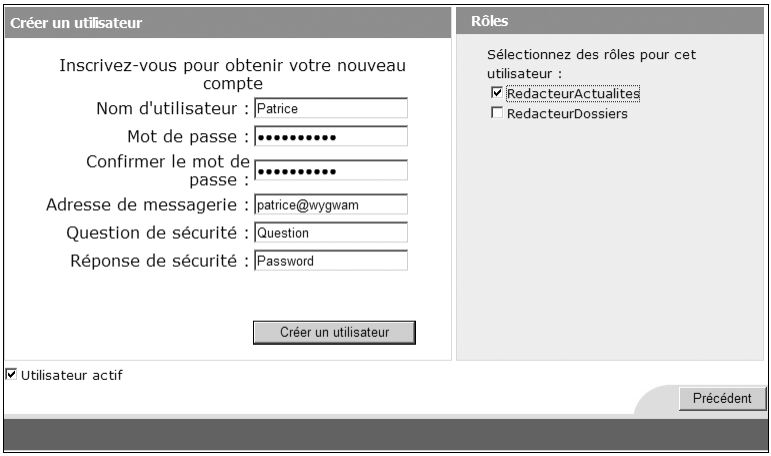


▲ **Figure 8-11 :** Création de règles d'accès

Création d'un utilisateur

Il ne reste plus qu'à créer le compte utilisateur qui sera, par exemple, rédacteur d'actualités pour enfin être capable de mettre en place le mécanisme de gestion des membres et des rôles au sein de l'application.

Pour cela, sous l'onglet *Sécurité*, cliquez sur le lien *Créer un utilisateur*. Puis dans la fenêtre qui s'ouvre, saisissez toutes les informations nécessaires à la création du compte, sans oublier de sélectionner le rôle *RedacteurActualites*.



▲ **Figure 8-12 :** Création de l'utilisateur RedacteurActualites

Implémentation du site web

Il ne vous reste plus qu'à réaliser les pages de login et de création de compte pour que la sécurité soit fonctionnelle sur le site web.

Pour ce faire, rien de plus simple : ouvrez la page *Login.aspx* et placez un contrôle `Login`.

◄ **Figure 8-13 :** Contrôle Login dans la boîte à outils

Le contrôle `Login` représente une boîte de dialogue de connexion. Cette boîte est semblable à toutes celles que l'on a l'habitude de voir sur les sites web : elle permet au visiteur de saisir son nom d'utilisateur, son mot de passe et de spécifier s'il souhaite que son identification soit mémorisée ou non.

Il ne s'agit pas d'un simple contrôle graphique. Lorsque vous l'insérez sur une page web, vous indiquez à ASP .NET 2 que vous souhaitez utiliser les services de gestion des membres qu'il propose. En effet, ce contrôle a un comportement prédéfini, ce qui le rend prêt à l'emploi.

Vous pouvez dès à présent le tester en exécutant la page et en essayant de vous connecter sous le compte utilisateur *Administrateur* que vous avez créé, et sous un compte qui n'existe pas pour vérifier que ASP .NET renvoie bien une erreur d'identification dans ce cas.

Le contrôle étant prêt à l'emploi, il suffit d'appliquer une petite modification de mise en forme afin que le résultat soit parfait. Pour cela, cliquez sur le smart tag puis sur le lien *Mise en forme automatique*. Une boîte de dialogue s'affiche. Elle propose différents styles à appliquer au contrôle `Login`. Sélectionnez celui qui vous convient.

◄ **Figure 8-14 :** Appliquer un style au contrôle Login

Il faut également modifier la page *Signin.aspx*, qui permet de créer un compte utilisateur,. Placez un contrôle `CreateUserWizard`, qui est un Assistant de création de compte. Il permet de créer des utilisateurs sans écrire une seule ligne de code.

De plus, des validations sont effectués pour vérifier que le nom d'utilisateur est bien unique, que le mot de passe est assez complexe, etc.

Remarque

Mot de passe

Lors de la création d'utilisateurs avec `CreateUserWizard`, une erreur peut intervenir au niveau de la validation du mot de passe. En effet, les règles de sécurité d'ASP .NET imposent de saisir des mots de passe de 7 caractères minimum, avec au moins un caractère non alphanumérique. Il suffit de saisir un mot de passe satisfaisant ces règles pour que la création de l'utilisateur soit effective.

8.4 Check-list

Dans ce chapitre, vous avez appris à :

- insérer des données dans une base de données SQL Server Express 2005 ;
- afficher des données provenant d'une table grâce à un contrôle `SqlDataSource` ;
- implémenter des fonctions de sécurité telles que la gestion des membres et des rôles dans un site web.

Moniteur de performances

Les utilitaires de diagnostic permettant d'afficher des informations précises sur l'ordinateur que l'on est en train d'utiliser et de mesurer son activité sont de plus en plus légion, et sont de plus utilisés.

Ces outils permettent de s'assurer que tout fonctionne correctement, et de savoir le taux d'occupation de la machine en regardant des informations telles que l'occupation de la mémoire vive, l'espace disque utilisé sur le ou les disques dur, le taux d'occupation du microprocesseur, etc.

Pour avoir un outil qui vous convienne parfaitement, vous allez créer votre propre moniteur de performances. Il affichera les caractéristiques détaillées de l'ordinateur et surveillera les informations les plus pertinentes en les affichant sous forme de graphique.

9.1 Classes et espaces de noms utilisés

Vous allez utiliser plusieurs espaces noms.

L'espace de noms `My` est spécial. Il est apparu avec la version 2005 de Visual Basic et n'existait pas dans les versions 2002 et 2003. Il permet d'accéder à un ensemble de fonctionnalités souvent utilisées.

Vous allez créer une application Windows et utiliserez donc fréquemment l'espace de noms `System.Windows.Forms`, qui contient toutes les classes permettant de créer des interfaces graphiques grâce à des formulaires et à des contrôles.

`System.Management` donne accès à la bibliothèque WMI, qui permet de gérer un ordinateur et de récupérer des informations détaillés à son propos.

9.2 Espace de noms My

Vous allez commencer par récupérer quelques informations simples à propos de l'ordinateur grâce aux fonctionnalités proposées par l'espace de noms `My`. Il permet d'accéder rapidement à un ensemble de fonctionnalités utiles, sans recourir aux nombreux espaces de noms présents dans le Framework .NET 2.

`My` contient les classes suivantes :

Fonctionnalités proposées par l'espace de noms My	
Classe	**Fonctionnalité**
`Application`	Informations sur l'application : son titre, sa version, sa description, son auteur, etc.
`Computer`	Informations et accès à certaines fonctionnalités liées à l'ordinateur : manipulation de la Base de registre, accès au système de fichiers, informations sur les composants, etc.

Fonctionnalités proposées par l'espace de noms My	
Classe	**Fonctionnalité**
`Forms`	Permet d'accéder à la collection des formulaires de l'application.
`Ressources`	Permet d'accéder aux ressources de l'application : icônes, images, etc.
`Settings`	Paramètres de l'application et paramètres utilisateurs
`User`	Information sur l'utilisateur qui exécute l'application : nom d'utilisateur, groupe, domaine auquel il est rattaché, etc.
`WebServices`	Permet d'accéder à la collection des services web référencés au niveau du projet.

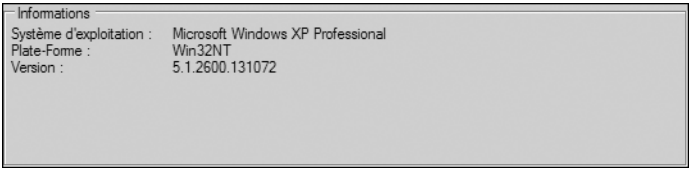
Vous allez dans un premier temps récupérer les informations proposées par la classe `Computer`, présente dans l'espace de noms `My`.

Informations sur le système d'exploitation

Vous allez commencer par afficher des informations sur le système d'exploitation installé sur votre ordinateur. Pour cela, et dans le but de les regrouper de manière logique dans la fenêtre, ajoutez un contrôle `GroupBox` puis trois contrôles `Label` dans `GroupBox`. Ces contrôles `Label` vont permettre d'afficher le nom de système d'exploitation, son type et sa version. Changez la propriété `Name` de ces contrôles `Label` et utilisez les valeurs suivantes : `lblOS`, `lblPlateForme` et `lblVersion`. Ajoutez trois autres contrôles `Label` pour décrire ces champs.

Il ne reste plus qu'à afficher la valeur réelle de ces informations dans les contrôles `Label` que vous venez de créer. Ces informations sont accessibles via la propriété `Info` de l'objet `Computer` présent dans l'espace de noms `My`. Vous allez donc utiliser `My.Computer.Info` pour afficher tous les éléments nécessaires dans l'événement `Load` du formulaire afin que les données soit affichées dès son ouverture.

```
Private Sub Form1_Load(ByVal sender As System.Object, _
       ByVal e As System.EventArgs) Handles MyBase.Load
        With My.Computer
            lblOS.Text = .Info.OSFullName
            lblPlateForme.Text = .Info.OSPlatform
            lblVersion.Text = .Info.OSVersion
        End With
    End Sub
```



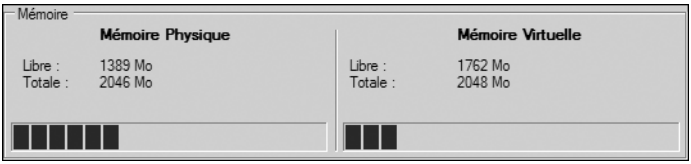
▲ **Figure 9-1 :** Affichage des informations sur le système d'exploitation

Information sur l'utilisation de la mémoire

Vous allez à présent afficher des informations sur l'utilisation de la mémoire de l'ordinateur. Il s'agit de données concernant la mémoire physique de la machine (la mémoire vive, également appelée RAM) et la mémoire virtuelle.

En ce sens, vous devez ajouter quelques contrôles à l'interface de la fenêtre. Commencez comme précédemment par placer un contrôle `GroupBox` pour délimiter le groupe de contrôles qui afficheront des données sur la mémoire. Ajoutez ensuite quatre contrôles `Label`. Modifiez leur propriété `Name` en affectant les valeurs `lblMemPhyFree`, `lblMemPhyTotal`, `lblMemVirtFree` et `lblMemVirtTotal`, pour afficher la mémoire physique libre, le volume total de mémoire physique, la mémoire virtuelle libre et le volume total de la mémoire virtuelle. Ajoutez quelques contrôles `Label` "compagnons" pour décrire les champs précédemment insérés dans la fenêtre.

Ajoutez enfin deux contrôles `ProgressBar` pour afficher le pourcentage de mémoire physique et virtuelle occupée.



▲ **Figure 9-2 :** Affichage des informations sur la mémoire

Pour afficher des informations sur la mémoire, vous allez utiliser comme précédemment `My.Computer.Info`. Mais pour actualiser ces données en quasi "temps réel", vous utiliserez un contrôle `Timer`.

Propriétés fournissant des informations sur la mémoire	
Propriété	**Description**
`AvailablePhysicalMemory`	Mémoire vive physique (RAM) disponible sur l'ordinateur. Valeur exprimée en octets.
`AvailableVirtualMemory`	Mémoire virtuelle disponible sur l'ordinateur. Valeur exprimée en octets.

Propriétés fournissant des informations sur la mémoire	
Propriété	**Description**
`TotalPhysicalMemory`	Total de la mémoire vive présente sur le système (libre et occupée). Valeur exprimée en octets.
`TotalVirtualMemory`	Total de la mémoire virtuelle présente sur le système (libre et occupée). Valeur exprimée en octets.

Les valeurs sont toutes exprimées en octets. Il est donc nécessaire d'effectuer une opération arithmétique pour convertir cette valeur en mégaoctets, plus lisible pour l'utilisateur.

Un kilo-octet étant égal à 1 024 octets, et un mégaoctet étant égal à 1 024 Ko, il faut donc diviser la valeur renvoyée par les propriétés par le carré de 1 024 :

```
With My.Computer.Info
lblMemPhyFree.Text = CType(.AvailablePhysicalMemory
/(1024*1024),Int16)
lblMemPhyFree.Text += " Mo"
End With
```

De même, pour afficher le pourcentage de mémoire occupée, il est nécessaire d'effectuer quelques opérations. La première étape consiste à récupérer le pourcentage de mémoire libre et à effectuer un simple calcul de pourcentage.

```
Dim physical As Double
physical = My.Computer.Info.AvailablePhysicalMemory/
➥ My.Computer.Info.TotalPhysicalMemory
physical *= 100
```

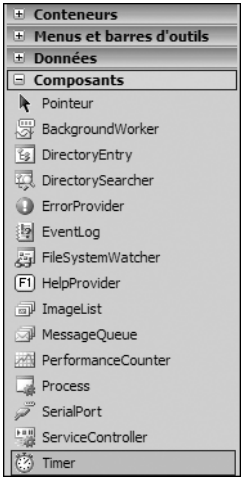
Vous avez à présent le pourcentage de mémoire libre. Pour avoir le pourcentage de mémoire occupée, il suffit de soustraire le pourcentage obtenu à 100.

```
        prgPhysical.Value = 100 - physical
```

`Timer` est un contrôle non visuel, qui permet d'exécuter du code en boucle par intervalles de temps régulier, par exemple toutes les secondes (voir Figure 9-3).

Placez un contrôle `Timer` sur votre formulaire. Celui-ci apparaît alors dans la zone des composants non visuels. Sélectionnez-le pour modifier ses propriétés.

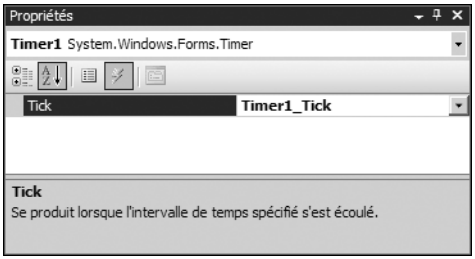
Modifiez sa propriété `Interval`, définie en millisecondes : donnez-lui une valeur de 500. `Interval` permet de définir l'intervalle de temps qui sépare deux exécutions de code.



◄ **Figure 9-3 :** Contrôle Timer dans la boîte à outils

Modifiez ensuite sa propriété `Enabled` et définissez-la à `True` pour activer le contrôle `Timer` dès l'ouverture de la fenêtre.

Il ne reste plus qu'à placer le code dans l'événement `Tick` du `Timer`, déclenché en boucle à intervalles de temps régulier.



◄ **Figure 9-4 :** Événement Tick

```
Private Sub Timer1_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Timer1.Tick
Dim physical As Double
With My.Computer.Info
lblMemPhyFree.Text = CType(.AvailablePhysicalMemory
/ (1024 * 1024), Int16)
lblMemPhyFree.Text += " Mo"
physical =.AvailablePhysicalMemory/.TotalPhysicalMemory
physical *= 100
prgPhysical.Value = 100 - physical

Dim virtual As Double
lblMemVirtFree.Text=CType(AvailableVirtualMemory
```

```
/(1024*1024),Int16)
lblMemVirtFree.Text += " Mo"
virtual =.AvailableVirtualMemory
virtual /=.TotalVirtualMemory
virtual *= 100
End With
prgVirtual.Value = 100 - virtual

    End Sub
```

Il ne manque plus qu'a afficher les informations concernant le volume total de mémoire libre et virtuelle. Comme elles sont fixes (on n'ajoute pas de mémoire vive lorsque l'ordinateur est en cours d'utilisation), vous allez afficher ces données une fois pour toutes, lors de l'ouverture de la fenêtre. Pour cela, il est nécessaire d'ajouter quelques lignes de code à l'événement `Load` du formulaire :

```
  With My.Computer
           lblOS.Text = .Info.OSFullName
           lblPlateForme.Text = .Info.OSPlatform
           lblVersion.Text = .Info.OSVersion
       End With
With My.Computer.Info
lblMemPhyTotal.Text=CType(.TotalPhysicalMemory
/(1024*1024),Int16)
lblMemPhyTotal.Text += " Mo"
lblMemVirtTotal.Text = CType(.TotalVirtualMemory
/(1024 * 1024), Int16)
End With
lblMemVirtTotal.Text += " Mo"
```

9.3 Récupérer des informations grâce à l'espace de noms System.Management

Il est possible de récupérer, grâce à l'espace de noms `My`, quasiment toutes les informations possibles à propos d'un système via une API appelée Windows Management Instrumentation (WMI). En outre, cette API permet de modifier des paramètres système pour maîtriser totalement l'environnement Windows. Des outils complets tels que Microsoft Operation Manager s'appuient sur WMI. Bien que développé en COM et donc non en .NET, WMI peut s'utiliser depuis une application développée en Visual Basic 2005 grâce à l'espace de noms `System.Management`, qui permet de consulter les informations offertes par WMI tout en cachant toute la complexité sous-jacente.

Vous allez dans cet exemple créer une classe permettant d'afficher des informations à propos du ou des processeurs présents dans votre ordinateur.

Cela n'est pas possible via l'espace de noms `My`. Vous allez donc vous tournez vers l'espace de noms `System.Management`.

Pour débuter, créez une classe nommée `Processeur`. Elle permettra de récupérer des informations telles que le fabriquant, le modèle, la fréquence, la plage d'adressage (32 ou 64 bits), ainsi que le nombre de processeurs :

```
Imports System.Management

Public Class Processeur

    Private m_fabriquant As String
    Dim m_modele As String
    Dim m_frequence As String
    Dim m_adressage As String
    Dim m_nombre As Integer

    Public Property Fabriquant() As String
        Get
            Return m_fabriquant
        End Get
        Set(ByVal value As String)
            m_fabriquant = value
        End Set
    End Property

    Public Property Modele() As String
        Get
            Return m_modele
        End Get
        Set(ByVal value As String)
            m_modele = value
        End Set
    End Property

    Public Property Frequence() As String
        Get
            Return m_frequence
        End Get
        Set(ByVal value As String)
            m_frequence = value
        End Set
    End Property

    Public Property Adressage() As String
        Get
            Return m_adressage
        End Get
        Set(ByVal value As String)
            m_adressage = value
```

```
        End Set
    End Property

    Public Property Nombre() As Integer
        Get
            Return m_nombre
        End Get
        Set(ByVal value As Integer)
            m_nombre = value
        End Set
    End Property
```

La classe `Processeur` permet de représenter un processeur via quelques caractéristiques implémentées à l'aide de propriétés. Il ne reste plus qu'à définir les valeurs de ces propriétés grâce à des objets présents dans l'espace de noms `System.Management`.

```
Public Sub New()
Dim query As New SelectQuery("Win32_Processor")
Dim search As New ManagementObjectSearcher(query)
For Each info As ManagementObject In search.Get
m_fabriquant = info("Manufacturer").ToString
m_modele = info("Name").ToString
m_frequence = (CDbl(info("CurrentClockSpeed")
/ 1000)).ToString & " GHz"
m_adressage = info("AddressWidth").ToString & " bits"
m_nombre = Environment.ProcessorCount
Next
search.Dispose()
End Sub

End Class
```

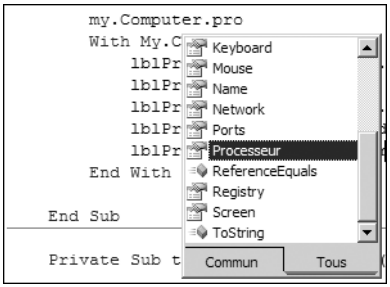
Extension de l'espace de noms My

Vous avez dans un premier temps utilisé l'espace de noms `My` pour récupérer des informations sur l'ordinateur, telles que le nom du système d'exploitation, le volume de mémoire physique libre. D'autres espaces de noms tels que `System.Management` ont permis d'aller plus loin et de récupérer par exemple des informations sur le processeur de l'ordinateur.

La classe `Processeur` pourrait être logiquement rattachée à l'objet `Computer` présent dans l'espace de noms `My` si celui-ci était extensible. La bonne nouvelle est qu'il l'est. Vous allez donc l'enrichir grâce à la classe `Processeur`. Pour cela, créez une nouvelle classe `MyComputer`. `MyComputer` est le nom interne de l'objet pointé par la propriété `Computer`. Définissez son espace de noms comme étant l'espace de noms `My`, puis ajoutez une propriété `Processeur` renvoyant un objet

de type `Processeur` :

```
Namespace My
    Friend Class MyComputer
        Private m_processeur As Processeur
        Public ReadOnly Property Processeur() As Processeur
            Get
                If m_processeur Is Nothing Then
                    m_processeur = New Processeur
                End If
                Return m_processeur
            End Get
        End Property
    End Class
End Namespace
```



◄ **Figure 9-5 :** Affichage via l'Intellisense de la propriété Computer étendue

9.4 Afficher des informations sur le processeur

Vous êtes à présent capable d'afficher les informations sur le processeur.

Insérez un contrôle `GroupBox` puis ajoutez-y cinq contrôles `Label` pour afficher les informations liées au processeur. Nommez-les `lblProcAdressage`, `lblProcFabriquant`, `lblProcFrequence`, `lblProcModele`, `lblProcNombre`.

Ajoutez ensuite le code suivant dans l'événement `Load` du formulaire pour afficher les informations :

```
With My.Computer.Processeur
            lblProcAdressage.Text = .Adressage
            lblProcFabriquant.Text = .Fabriquant
            lblProcFrequence.Text = .Frequence
            lblProcModele.Text = .Modele
            lblProcNombre.Text = .Nombre
        End With
```

9.5 Créer une vue synthétique

Une grande partie des outils de surveillance des performances proposent une vue synthétique des données qu'ils récupèrent pour les afficher tout le temps en premier plan sur l'écran de l'utilisateur. Ces vues sont souvent partiellement transparentes pour ne pas masquer la ou les fenêtres qui se situent en arrière-plan afin de ne pas gêner l'espace de travail.

Vous allez donc faire de même pour visualiser le pourcentage du processeur occupé, ainsi que le pourcentage de mémoire vive occupée, et cela en permanence via une fenêtre transparente de petite taille.

En ce sens, créez une nouvelle fenêtre que vous appellerez **Monitor**. Cliquez du bouton droit sur votre projet pour afficher le menu contextuel et sélectionnez l'élément de menu intitulé **Ajouter un nouvel élément**. Dans la boîte de dialogue qui s'affiche, sélectionnez l'icône *Windows Form*.

▲ **Figure 9-6 :** Création d'un nouveau formulaire Windows

Validez pour débuter la construction du formulaire. Vous vous retrouvez à présent devant le Concepteur de formulaires proposé par Visual Basic 2005 Express Edition. La fenêtre d'affichage devant être le plus discrète possible, vous allez la configurer pour la rendre beaucoup moins imposante que la majorité des formulaires qui composent les applications Windows.

Commencez par supprimer la barre de titre de la fenêtre, contenant le nom de l'application, son icône ainsi que les boutons système de réduction, d'agrandissement et de fermeture de l'application. Ces informations sont inutiles dans ce cas. Pour opérer la suppression, modifiez la propriété `FormBorderStyle` de la

fenêtre. Cette propriété définit le type de bordure de la fenêtre et peut prendre plusieurs valeurs :

Valeurs possibles de la propriété FormBorderStyle	
Valeur	**Description**
`None`	Définit une fenêtre sans bordure ni barre de titre. Cette valeur est rarement utilisée mais elle est pratique pour créer des applications qui doivent être affichées en plein écran, comme un économiseur d'écran ou une fenêtre qui doit être la plus discrète possible.
`FixedSingle`	Définit une fenêtre avec une bordure de taille fixe composée d'un seul trait.
`Fixed3D`	Définit une fenêtre avec une bordure de taille fixe, ayant un effet 3D grâce à l'utilisation de plusieurs traits.
`FixedDialog`	Définit une fenêtre avec une bordure. Cette valeur est couramment utilisée pour les boîtes de dialogue.
`Sizable`	Définit une fenêtre redimensionnable par l'utilisateur à l'aide de la souris.
`FixedToolWindow`	Définit une fenêtre de taille fixe, avec une barre de titre réduite et une bordure fine. Cette valeur est utilisée pour les barres d'outils flottantes.
`SizableToolWindow`	Définit une fenêtre que l'utilisateur peut redimensionner, avec une barre de titre réduite. Cette valeur est utilisée pour les barres d'outils flottantes redimensionnables.

Vous allez définir cette propriété `FormBorderStyle` en lui affectant une valeur égale à `None`.

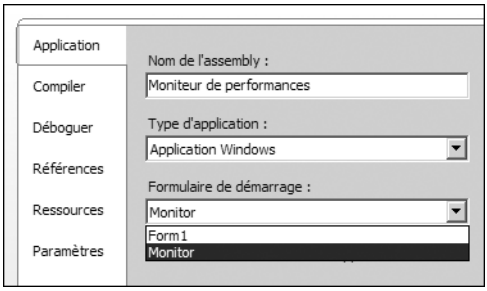
De plus, pour rendre encore plus discrète cette fenêtre, réduisez fortement sa taille, définissez une couleur d'arrière-plan qui passe inaperçue, et surtout activez la transparence en modifiant la propriété `Opacity`. `Opacity` permet de définir le pourcentage d'opacité de la fenêtre, 0 % correspondant à une fenêtre totalement transparente, et 100 % à une fenêtre totalement opaque. Choisissez une transparence assez forte pour ne pas gêner l'espace de travail, en l'occurrence une valeur de 20 %.

Ajoutez ensuite deux contrôles `ProgressBar`. Ils permettront d'afficher le pourcentage de CPU occupée ainsi que le pourcentage de mémoire vive occupée.

◄ **Figure 9-7 :** Exemple de fenêtre synthétique

Pour définir cette nouvelle fenêtre en tant que formulaire principal de l'application, vous devez modifier une propriété du projet. Pour cela, cliquez du bouton droit sur votre projet dans l'Explorateur de solutions puis sélectionnez **Propriétés**.

Sous l'onglet qui s'affiche, modifiez le formulaire de démarrage et sélectionnez *Monitor*.



◀ **Figure 9-8 :** Sélection du formulaire de démarrage

Ajoutez à présent un menu contextuel au formulaire *Monitor* pour permettre à l'utilisateur de quitter l'application et d'afficher la fenêtre d'informations détaillées créée en début de chapitre. Pour cela, utilisez un contrôle `ContextMenuStrip` et insérez deux éléments de menu : un pour afficher la fenêtre détaillée grâce à la ligne de code suivante :

```
My.Forms.Form1.Show()
```

et un second pour quitter l'application :

```
Private Sub QuitterToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs) Handles
➥ QuitterToolStripMenuItem.Click
        Application.Exit()
    End Sub
```

Liez à présent ce menu contextuel au formulaire en modifiant la propriété `ContextMenuStrip` et en sélectionnant le menu que vous venez de créer.

Déplacement d'une fenêtre sans bordure

La fenêtre est à présent sans bordure, avec une transparence qui lui permet d'être discrète, mais la suppression de ladite bordure entraîne un problème épineux : on ne peut plus déplacer la fenêtre car la barre de titre est absente.

Il est donc nécessaire de coder par programmation les déplacements, qui s'opéreront, non plus via la barre de titre, mais via la surface de la fenêtre elle-même. Il s'agira d'appuyer sur le bouton gauche de la souris tout en

déplaçant le pointeur, à la manière d'un glisser-lâcher. Ainsi, les habitudes de l'utilisateur ne seront pas troublées.

Pour ce faire, vous avez besoin de deux événements : `MouseDown`, pour stocker la position originale de la souris au départ du glisser-lâcher, et `MouseMove`, pour déplacer la fenêtre en même temps que la souris.

Pour stocker la position de départ de la souris, déclarez une variable nommée `m_positionDepart` de type `Point` :

```
Dim m_positionDepart As Point
```

Dans l'événement `MouseDown` du formulaire, initialisez cette valeur :

```
Private Sub Monitor_MouseDown(ByVal sender
      As System.Object, ByVal e As
      System.Windows.Forms.MouseEventArgs)
      Handles MyBase.MouseDown
      m_positionDepart = e.Location
  End Sub
```

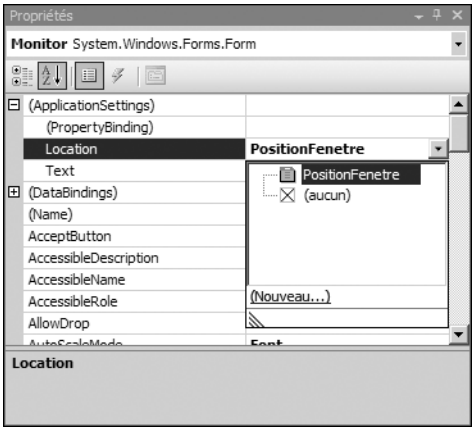
Dans l'événement `MouseMove`, déplacez la fenêtre en modifiant les propriétés `Left` et `Top` du formulaire :

```
Private Sub Monitor_MouseMove(ByVal sender As
      System.Object, ByVal e As
      System.Windows.Forms.MouseEventArgs)
      Handles MyBase.MouseMove
      If e.Button = Windows.Forms.MouseButtons.Left Then
          Me.Left += e.X - m_positionDepart.X
          Me.Top += e.Y - m_positionDepart.Y
      End If
  End Sub
```

Enregistrer des paramètres d'application

L'utilisateur est à présent capable de déplacer la fenêtre où il le souhaite, dans un endroit où elle est le moins gênante vis-à-vis de la surface de travail. Le résultat est satisfaisant. Il reste un problème à régler pour compléter cette gestion : la sauvegarde de la position de la fenêtre. En effet, dès que l'on quitte l'application, la position de la fenêtre est perdue et l'utilisateur doit la replacer manuellement. Ce comportement n'est pas celui que l'on attend d'une application Windows. Vous allez corriger ce problème de manière simple, sans écrire une ligne de code, grâce à une fonctionnalité offerte par Visual Basic 2005 Express : les paramètres d'application (*Application Settings*).

Pour stocker la position de la fenêtre, sélectionnez celle-ci, et dans la fenêtre des propriétés, repérez l'élément intitulé *(Application Settings)*. Dépliez-le pour afficher la propriété `Location`. Cliquez dans la zone de valeur de cette propriété, puis sur le lien *Nouveau* pour définir la valeur, et intitulez-la `PositionFenetre`. En effectuant cela, vous indiquez qu'il faut stocker, dans un fichier de ressources propre à l'utilisateur courant, la position de la fenêtre dans une variable nommée `PositionFenetre`.



◄ **Figure 9-9 :** Sauvegarde de la position de la fenêtre

À présent, lorsque l'on quitte l'application, la position de la fenêtre est automatiquement mémorisée dans un profil lié à l'utilisateur courant (donc, s'il y a plusieurs utilisateurs, les paramètres pourront être différents). Ce profil sera automatiquement chargé lors du démarrage de l'application.

9.6 Check-list

Dans ce chapitre, vous avez appris à :

- récupérer des informations sur le système grâce aux espaces de noms `My` et `System.Management` ;
- étendre l'espace de noms `My` ;
- utiliser le contrôle `Timer` ;
- modifier les propriétés des fenêtres (suppression de bordure, transparence…) ;
- déplacer une fenêtre sans bordure ;
- sauvegarder des paramètres d'application.

Client MSN Messenger avec onglets

Au cours des chapitres précédents, vous avez écrit plusieurs applications en vous servant des bibliothèques de classes de la plateforme .NET et, évidemment, du code que vous avez écrit vous-même. Or, comme avec tout langage de programmation, il est possible avec Visual Basic .NET de faire appel à des bibliothèques écrites par des tiers, éventuellement dans d'autres langages pris en charge par la plateforme .NET. La bibliothèque DotMSN que vous utiliserez dans cette application, par exemple, a été écrite en C#.

Vous utiliserez cette bibliothèque pour écrire un client capable de se connecter au service de messagerie MSN. Cette application sera différente des autres clients qui existent déjà, car elle ne consistera qu'en une seule fenêtre dans laquelle vous verrez votre liste de contacts et toutes vos conversations sous la forme d'onglets.

L'application que vous obtiendrez à la fin de ce chapitre vous permettra d'échanger des messages avec d'autres clients MSN. Toutefois, sachez que le développement d'un client de messagerie complet n'est pas une tâche facile. Le but de ce chapitre est de vous mettre sur la bonne voie pour vos projets futurs.

10.1 Classes et espaces de noms utilisés

Comme pour toute application Windows, la plupart des classes dont vous aurez besoin se trouvent dans l'espace de noms `System.Windows.Forms`.

De plus, les classes de la bibliothèque DotMSN, dont vous aurez besoin pour interagir avec les serveurs du service de messagerie, se trouvent dans l'espace de noms `XihSolutions.DotMSN`.

10.2 Configuration

Une fois que vous aurez créé un projet de type *Application Windows*, vous devez le configurer pour pouvoir accéder aux classes de la bibliothèque DotMSN.

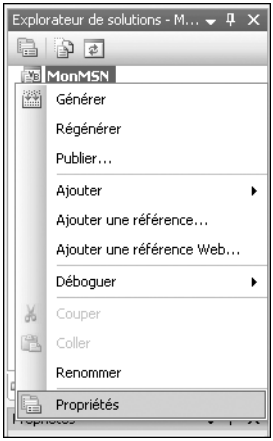
Commencez par télécharger DotMSN sur le site **www.xihsolutions.net/DotMSN**. La version de la bibliothèque utilisée dans ce chapitre est la 2.0.1. Bien que le code source de cette bibliothèque soit disponible en téléchargement, vous n'aurez besoin que de la version compilée sous forme de fichier *.dll*. Enregistrez ce fichier dans le dossier racine de votre projet qui se trouve, par défaut, dans le répertoire *Mes documents\Visual Studio 2005\Projects*.

Remarque

Dossier racine de votre projet

Lorsque vous créez un nouveau projet avec Visual Basic 2005 Express, tous les fichiers sont stockés dans un premier temps dans un répertoire temporaire. Vous devez utiliser la commande **Enregistrer tout** du menu **Fichier**, ou le raccourci clavier Ctrl+Maj+S, pour enregistrer votre projet à son emplacement définitif.

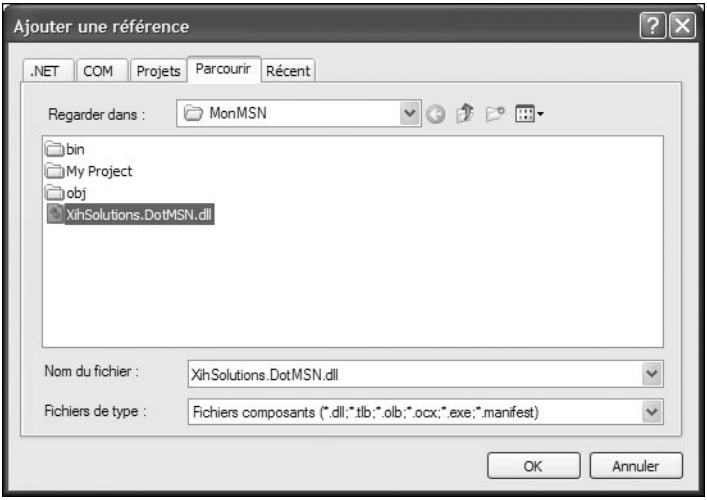
Malheureusement, le fait d'enregistrer le fichier *.dll* à la racine de votre projet ne suffit pas pour que celui-ci soit pris en compte. Cliquez du bouton droit sur le nom de votre projet dans l'Explorateur de solutions et sélectionnez la commande **Propriétés**.



◄ **Figure 10-1 :** Accéder aux propriétés d'un projet

Dans la fenêtre qui s'ouvre, sélectionnez l'onglet **Références** à gauche et cliquez sur le bouton **Ajouter** pour ajouter une référence vers la bibliothèque DotMSN. Dans la boîte de dialogue qui s'ouvre, sélectionnez l'onglet **Parcourir**. Par défaut, vous vous retrouvez dans le répertoire racine de votre application, dans lequel figure le fichier *.dll* correspondant à la bibliothèque DotMSN. Sélectionnez-le et validez (voir Figure 10-2).

La référence `XihSolutions.DotMSN` apparaît à présent dans la liste de références de votre projet.



▲ **Figure 10-2 :** Ajouter une référence à un projet

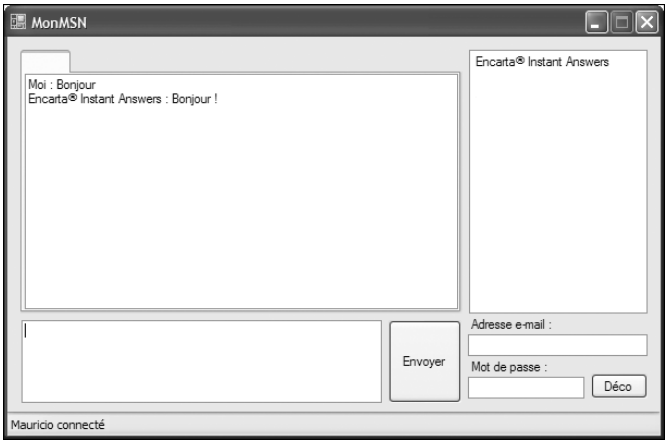
Pour finir, et pour éviter de devoir saisir le nom complet de l'espace de noms à chaque fois que vous souhaitez utiliser une classe de celui-ci, importez-le en cochant la case à côté de `XihSolutions.DotMSN` dans la liste *Espaces de noms importés*.

▲ **Figure 10-3 :** Importer un espace de noms

Fermez l'onglet de configuration de votre projet pour enregistrer les modifications.

10.3 Interface utilisateur

L'interface utilisateur que vous allez créer consiste en un seul formulaire qui contiendra la liste de contacts ainsi que les conversations sous forme d'onglets.

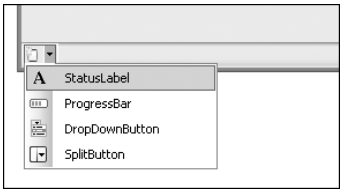


▲ **Figure 10-4 :** Interface utilisateur terminée

Formulaire principal

Le formulaire principal est simple, tout en étant capable de gérer plusieurs conversations simultanées avec les contacts MSN.

Commencez par glisser un contrôle `StatusStrip` dans votre formulaire. Il viendra se placer, automatiquement, en bas de celui-ci. Insérez dans la barre d'état un contrôle de type `ToolStripStatusLabel`, que vous appellerez `Status` (en modifiant sa propriété `(Name)`). Videz sa propriété `Text` afin qu'il n'affiche rien.



◄ **Figure 10-5 :** Insertion d'un ToolStripStatusLabel

Insérez dans votre formulaire un `TabControl` que vous appellerez `Conversations` et supprimez tous ses onglets à l'aide de sa balise active. Ce contrôle viendra se placer en haut à gauche du formulaire.

À côté du `TabControl`, glissez un contrôle `ListView` qui s'appellera `Contacts`. Changez ses propriétés `MultiSelect` à `False`, `Sorting` à `Ascending` et `View` à `SmallIcon`. De cette manière, vos contacts seront rangés par ordre alphabétique et vous ne pourrez en sélectionner qu'un seul à la fois. De plus, le fait d'utiliser la vue `SmallIcon` fait que vous pourrez, à l'avenir, associer une icône à chacun de vos contacts, ce que l'on ne fait pas dans cet exemple.

Ajoutez un contrôle TextBox en dessous du TabControl. Appelez-le MonTexte et affectez la valeur True à sa propriété Multiline pour pouvoir le redimensionner en hauteur et y écrire des messages qui s'étendent sur plusieurs lignes. À côté de ce champ de texte, insérez un contrôle Button que vous appellerez Envoyer et modifiez sa propriété Text afin qu'il affiche cette même valeur.

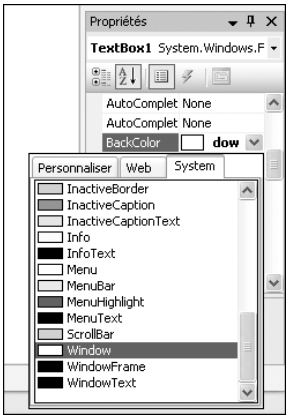
Enfin, insérez deux contrôles TextBox et un contrôle Button que vous appellerez respectivement Adresse, MotDePasse et Connexion. Vous pouvez aussi insérer des contrôles Label pour identifier les champs de texte si vous le souhaitez. Modifiez la propriété Text du bouton afin qu'il affiche OK et changez la valeur de la propriété UseSystemPasswordChar du champ MotDePasse à True afin que le texte saisi ne soit pas visible.

Votre interface graphique est maintenant terminée, à l'exception d'un détail : vous avez peut-être remarqué qu'il n'y a pas de champ de texte prévu pour l'affichage des conversations.

Contrôle Dialogue

Vous devez créer un contrôle utilisateur pour afficher vos conversations avec vos contacts MSN. Pour cela, cliquez du bouton droit sur le nom de votre projet dans l'Explorateur de solutions, et choisissez la commande **Contrôle utilisateur** du sous-menu **Ajouter**. Appelez votre contrôle Dialogue.

Dans votre nouveau contrôle, insérez une TextBox. Modifiez ses propriétés Multiline et ReadOnly à True et sa propriété Dock à Fill. Ainsi, votre champ de texte occupera tout le contrôle et il ne pourra pas être modifié par l'utilisateur. En revanche, son texte sera toujours accessible grâce à sa propriété Text. Pour finir, changez sa couleur de fond en modifiant sa propriété BackColor et en lui attribuant la valeur Window.



◄ **Figure 10-6 :** Sélectionner une couleur système pour le fond d'un contrôle

Votre contrôle utilisateur fera beaucoup plus qu'afficher un simple champ de texte, mais les fonctionnalités supplémentaires devront être ajoutées par la suite via du code.

10.4 Réalisation

Maintenant que votre interface graphique est prête à se connecter au service de messagerie MSN pour envoyer et recevoir des messages, vous pouvez commencer à écrire le code qui utilise les classes de la bibliothèque DotMSN.

Connexion au service de messagerie

Vous allez commencer par configurer votre application afin qu'elle se connecte au service de messagerie MSN. Pour cela, vous devez afficher le code source de votre formulaire principal en appuyant sur la touche F7 de votre clavier.

```
Public Class Form1
    Private Delegate Sub _
      UpdateControlTextDelegate(ByVal text As String)
    Private Delegate Sub UpdateControlDelegate()
    Private Delegate Sub _
      CreateConversationTabDelegate( _
        ByVal conversation As Conversation)

    Friend Messenger As Messenger

    Public Sub New()
      ' Cet appel est requis par le Concepteur
      InitializeComponent()

      Messenger = New Messenger()

      ' Imiter MSN Messenger
      Messenger.Credentials.ClientID = "msmsgs@msnmsgr.com"
      Messenger.Credentials.ClientCode = "Q1P7W2E4J9R8U3S5"

      AddHandler _
      Messenger.NameserverProcessor.ConnectionEstablished, _
      AddressOf NameserverProcessor_ConnectionEstablished

      AddHandler Messenger.Nameserver.SignedIn, _
      AddressOf Nameserver_SignedIn

      AddHandler Messenger.Nameserver.SignedOff, _
      AddressOf Nameserver_SignedOff

      AddHandler Messenger.Nameserver.AuthenticationError, _
```

```
      AddressOf Nameserver_AuthenticationError

      AddHandler Messenger.ConversationCreated, _
      AddressOf Messenger_ConversationCreated
    End Sub
End Class
```

▲ Nouveau constructeur du formulaire principal

On commence par créer trois délégués. Ils serviront à appeler des méthodes à travers différents threads pour mettre à jour l'interface graphique de l'application. Ensuite on crée un objet appelé `Messenger` de type `XihSolutions.DotMSN.Messenger`, qui assurera la communication avec le service de messagerie.

On crée aussi un constructeur pour le formulaire afin d'initialiser l'objet `Messenger`. Il faut configurer les paramètres avec lesquels le client va se présenter auprès du service de messagerie. Ces paramètres permettent d'identifier l'application, et non pas l'utilisateur qui veut se connecter au service. Dans cet exemple, on personnifie le client MSN Messenger de Microsoft.

Si vous souhaitez déployer une application qui se connecte au service de messagerie MSN, vous devrez vous procurer vos propres identifiants auprès de MSN.

Enfin, on associe des méthodes à certains événements qui peuvent être déclenchés par le nouveau client.

Vous allez écrire des gestionnaires pour les événements de base (il ne s'agit pas d'une liste exhaustive).

Gestion des événements du service de messagerie

Le premier événement que vous allez gérer est `ConnectionEstablished`. Ce n'est pas un événement de l'objet `Messenger`, mais de la propriété `NameserverProcessor` de celui-ci.

```
Private Sub NameserverProcessor_ConnectionEstablished( _
  ByVal sender As Object, ByVal e As EventArgs)
    Update_Status("Connecté au serveur")
End Sub
```

▲ Gestionnaire de l'événement ConnectionEstablished

Cet événement ne semble pas intéressant, car il est seulement déclenché lors de la connexion initiale au serveur du service de messagerie et tout ce qu'il fait est de mettre à jour le texte de la barre d'état pour informer l'utilisateur.

Or, cette opération est loin d'être anodine. En effet, lorsque vous démarrez la connexion au service, la bibliothèque DotMSN crée un nouveau thread pour ne

pas bloquer l'application. Le gestionnaire étant appelé à partir de ce nouveau thread, différent de celui dans lequel ont été créés les éléments de l'interface graphique, la mise à jour de ces derniers ne peut pas se faire directement à partir de celui-ci.

Vous allez donc créer deux méthodes dans votre classe : `Update_Status` et `Update_Status_Synchro`.

```
Private Sub Update_Status(ByVal status As String)
  Invoke(New UpdateControlTextDelegate( _
   AddressOf Update_Status_Synchro), New Object() {status})
End Sub

Private Sub Update_Status_Synchro(ByVal status As String)
    Me.Status.Text = status
End Sub
```

▲ Mise à jour d'un élément de l'interface graphique à travers différents threads

La deuxième méthode sert à mettre à jour la propriété `Text` du contrôle appelé `Status` et ne requiert pas d'explication supplémentaire.

La première méthode s'assurera que la mise à jour du contrôle se fera dans le thread approprié. Pour cela, elle fait appel à la méthode `Invoke` en lui passant un délégué et un tableau de paramètres.

Le délégué a été préparé lors des premières modifications de la classe du formulaire principal. Il permet l'appel indirect à une méthode qui prend une chaîne de caractères en paramètre. Dans ce cas précis, il s'agit de la méthode `Update_Status_Synchro`. La chaîne de caractères à passer sera insérée dans d'un tableau d'objets afin que la méthode `Invoke` la transmette à la méthode appelée.

Le même modèle de code sera utilisé tout au long du développement de l'application pour mettre à jour les éléments de l'interface graphique à travers différents threads.

Vous allez maintenant gérer les événements `SignedIn` et `SignedOff`, déclenchés respectivement lorsque le client a authentifié l'utilisateur auprès des serveurs de messagerie et lorsque l'utilisateur s'est déconnecté. Ils appartiennent à la propriété `Nameserver` de `Messenger`.

```
Private Sub Nameserver_SignedIn(ByVal sender As Object, _
  ByVal e As EventArgs)
    Messenger.Owner.Status = PresenceStatus.Online
    Update_Contacts()
    Update_Status(Messenger.Owner.Name & " connecté")
    Update_Bouton("Déco")
End Sub

Private Sub Nameserver_SignedOff(ByVal sender As Object, _
```

```
  ByVal e As SignedOffEventArgs)
    Messenger.Disconnect()
    CloseAllTabs()
    Update_Contacts()
    Update_Status("Déconnecté")
    Update_Bouton("OK")
End Sub
```

▲ Gestion des événements SignedIn et SignedOff

Dans le premier gestionnaire, on indique que le statut de l'utilisateur authentifié est "en ligne", via la propriété `Owner` de `Messenger`. Ensuite, on fait appel à des méthodes auxiliaires, telles que `Update_Status`, pour mettre à jour l'interface graphique. Dans le deuxième gestionnaire, on fait de même, sauf que l'on déconnecte l'objet `Messenger` au lieu de changer le statut de l'utilisateur. Voici le code des méthodes auxiliaires nécessaires :

```
Private Sub Update_Bouton(ByVal text As String)
    Invoke(New UpdateControlTextDelegate( _
     AddressOf Update_Bouton_Synchro), New Object() {text})
End Sub

Private Sub Update_Bouton_Synchro(ByVal text As String)
    Connexion.Text = text
End Sub

Private Sub Update_Contacts()
    Invoke(New UpdateControlDelegate( _
      AddressOf Update_Contacts_Synchro))
End Sub

Private Sub Update_Contacts_Synchro()
   Contacts.SuspendLayout()
   Contacts.Items.Clear()
   If Messenger.Connected Then
   For Each Contact As Contact In Messenger.ContactList.All
        Dim item As New ListViewItem
        item.Text = Contact.Name
        item.Tag = Contact
        Contacts.Items.Add(item)
   Next
   End If
   Contacts.ResumeLayout()
End Sub

Private Sub CloseAllTabs()
    Invoke(New UpdateControlDelegate( _
      AddressOf CloseAllTabs_Synchro))
End Sub
```

```
Private Sub CloseAllTabs_Synchro()
    For Each tab As TabPage In Conversations.TabPages
        Conversations.TabPages.Remove(tab)
    Next
End Sub
```

▲ Méthodes auxiliaires pour la mise à jour de l'interface graphique

La méthode `Update_Bouton` est presque identique à `Update_Status`, sauf qu'elle met à jour le texte du bouton appelé `Connexion`.

La méthode `Update_Contacts` vide la liste `Contacts` et la remplit, seulement si l'objet `Messenger` est connecté, avec la liste d'utilisateurs contenue dans la propriété `ContactList` de `Messenger`. Aucune vérification du statut de l'utilisateur n'est faite. On peut la faire en testant la valeur de `Contact.Status`. Comme les contacts sont stockés un à un dans la propriété `Tag` de chaque élément de la liste, on peut récupérer l'objet `Contact`, et pas seulement son nom, lorsque l'on voudra initier une conversation avec celui-ci.

On appelle la dernière méthode lorsque l'utilisateur est déconnecté pour itérer dans la liste des onglets de conversation ouverts et les supprimer.

Le dernier événement lié à la connexion que l'on va gérer est celui qui est déclenché lorsque les paramètres d'authentification sont incorrects :

```
Private Sub Nameserver_AuthenticationError( _
  ByVal sender As Object, ByVal e As ExceptionEventArgs)
  MessageBox.Show( _
  "Vérifiez votre adresse e-mail et votre mot de passe.", _
  "Erreur d'authentification")
    Messenger.Disconnect()
End Sub
```

▲ Gestion de l'événement AuthenticationError

Il suffit d'afficher un message d'alerte et de déconnecter l'objet `Messenger`.

Enfin, on gère l'événement déclenché lorsqu'une conversation est créée :

```
Private Sub Messenger_ConversationCreated( _
  ByVal sender As Object, _
  ByVal e As ConversationCreatedEventArgs)
    If e.Initiator Is Nothing Then
        Invoke(New CreateConversationTabDelegate( _
          AddressOf CreateConversationTab), _
          New Object() {e.Conversation})
    End If
End Sub
```

▲ Gestion des nouvelles conversations

Cette méthode teste s'il s'agit d'une nouvelle conversation en vérifiant la propriété `Initator`, qui est différente de `Nothing` si la conversation est déjà ouverte. Si c'est le cas, la méthode qui crée les onglets de conversation est appelée. Une fois de plus, on doit utiliser `Invoke` ici car `CreateConversationTab` modifie des éléments de l'interface graphique.

```
Private Sub CreateConversationTab( _
  ByVal conversation As Conversation)
   Dim newTabPage As New TabPage
   Dim newDialogue As New Dialogue(conversation)

   If conversation.SwitchboardProcessor Is Nothing OrElse _
    Not conversation.SwitchboardProcessor.Connected Then
    conversation.Messenger.Nameserver.RequestSwitchboard( _
    conversation.Switchboard, conversation.Messenger)
    Threading.Thread.Sleep(1000)
   End If

   Dim contacts As New System.Text.StringBuilder
   For Each contact As DictionaryEntry In _
     conversation.Switchboard.Contacts
       contacts.AppendFormat("{0}, ", _
         CType(contact.Value, Contact).Name)
   Next
   If (contacts.Length > 2) Then
       contacts.Remove(contacts.Length - 2, 2)
   End If

   newTabPage.Text = contacts.ToString

   newDialogue.Dock = DockStyle.Fill

   newTabPage.Controls.Add(newDialogue)

   Conversations.TabPages.Add(newTabPage)
   Conversations.SelectTab(newTabPage)
End Sub
```

▲ Création de nouveaux onglets de conversation

On crée un nouvel objet `TabPage`, qui correspond à l'onglet que l'on va insérer dans le contrôle `Conversations`, ainsi qu'un objet `Dialogue`, le contrôle utilisateur. Le constructeur du contrôle utilisateur prend une conversation en paramètre.

Vous écrirez ce constructeur dans la section suivante.

Ensuite, on vérifie l'existence d'un `SwitchboardProcessor` dans la conversation. Cet objet "route" la conversation vers le bon destinataire. S'il n'existe pas, on le crée puis on attend 1 seconde avant de s'assurer qu'il a été initialisé correctement.

On peut maintenant itérer dans la liste de contacts associés à la conversation pour utiliser leurs noms comme titre de l'onglet grâce à la propriété `Text` de ce dernier.

Il ne reste plus qu'à attribuer la valeur `Fill` à la propriété `Dock` du contrôle `Dialogue` afin qu'il occupe tout l'onglet une fois ajouté dans celui-ci. On procède à cet ajout juste avant d'ajouter l'onglet dans le contrôle `Conversations`.

Contrôle Dialogue

Le contrôle utilisateur `Dialogue` doit être modifié pour être utilisé avec l'application. En effet, vous allez écrire du code qui ressemble beaucoup à celui que vous avez écrit pour la classe de la fenêtre principale. Voici le code complet de la classe `Dialogue` :

```
Public Class Dialogue
    Private Delegate Sub UpdateConversationText( _
      ByVal text As String)

    Private m_conversation As Conversation

    Public Sub New(ByVal conversation As Conversation)
        ' Cet appel est requis par le Concepteur
        InitializeComponent()

        m_conversation = conversation
        AddHandler _
          conversation.Switchboard.TextMessageReceived, _
          AddressOf Switchboard_TextMessageReceived
    End Sub

    Public ReadOnly Property Conversation() As Conversation
        Get
            Return m_conversation
        End Get
    End Property

    Public Sub AppendText(ByVal text As String)
        Invoke(New UpdateConversationText( _
          AddressOf AppendText_Synch), New Object() {text})
    End Sub
```

```
    Private Sub AppendText_Synch(ByVal text As String)
        TextBox1.AppendText(text)
    End Sub

    Private Sub Switchboard_TextMessageReceived( _
    ByVal sender As Object, ByVal e As TextMessageEventArgs)
        AppendText(String.Format("{0} : {1}{2}", _
          e.Sender.Name, e.Message.Text, vbCrLf))
    End Sub
End Class
```

▲ Classe Dialogue

On crée, dans cette classe aussi, un délégué qui facilitera les mises à jour de l'interface graphique entre les threads. On ne stocke pas de `Messenger` dans le contrôle, mais seulement une `Conversation`, que l'on initialise dans le constructeur et que l'on rend accessible grâce à une propriété en lecture seule.

Cette classe propose aussi deux méthodes auxiliaires, qui servent à faire les modifications de l'interface graphique, la première appelant la seconde à l'aide de la méthode `Invoke` et du délégué.

Finalement, on écrit le gestionnaire d'événements `TextMessageReceived` de l'objet `Conversation` stocké dans la classe. On utilise les méthodes auxiliaires pour ajouter dans la `TextBox` du contrôle utilisateur les messages reçus.

Maintenant que le contrôle utilisateur est capable de gérer des conversations, il ne reste plus qu'à utiliser des méthodes de l'objet `Messenger` à partir des contrôles de la fenêtre principale pour commencer à dialoguer avec les contacts MSN.

Gestion des événements de la fenêtre principale

Bien que votre application sache réagir lorsqu'elle est connectée au service de messagerie, elle ne sait pas encore s'y connecter. Vous allez donc implémenter le gestionnaire d'événements `Click` du bouton `Connexion` pour utiliser les valeurs des champs `Adresse` et `MotDePasse` en vue de l'authentification auprès du service.

```
Private Sub Connexion_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Connexion.Click
    If Not Messenger.Connected Then
        If Adresse.Text.Length > 0 Then
            Messenger.Credentials.Account = Adresse.Text
            Messenger.Credentials.Password = MotDePasse.Text

            Messenger.Connect()
```

```
            Update_Status("Connexion…")
        End If
    Else
        Messenger.Disconnect()
    End If
End Sub
```

▲ Connexion au service de messagerie

Ce bouton a deux fonctions : si, lorsque l'utilisateur clique dessus, l'objet `Messenger` n'est pas connecté au service, il est configuré avec l'adresse e-mail et le mot de passe fournis, et connecté ; sinon, il est déconnecté.

Maintenant que vous êtes connecté au réseau, vous voudrez sûrement initier des conversations avec vos contacts. Pour cela, vous double-cliquerez sur un contact dans la liste et le gestionnaire d'événements créera l'onglet approprié :

```
Private Sub Contacts_DoubleClick(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles Contacts.DoubleClick
    If Contacts.SelectedItems.Count = 1 Then
        Dim contactSelectionne As Contact = _
          CType(Contacts.SelectedItems(0).Tag, Contact)

        If contactSelectionne IsNot Nothing AndAlso _
           contactSelectionne.Online Then
            Dim conversation As Conversation = _
              Messenger.CreateConversation()
            conversation.Invite(contactSelectionne)
            CreateConversationTab(conversation)
        End If
    End If
End Sub
```

▲ Ouverture d'un onglet de conversation

On vérifie qu'un élément du `ListView Contacts` est bien sélectionné et on récupère l'objet `Contact` stocké dans sa propriété `Tag`. Après avoir vérifié que le contact est en ligne, on demande à l'objet `Messenger` de créer une conversation grâce à la méthode `CreateConversation`, et on invite le contact grâce à la méthode `Invite`. Enfin, on appelle la méthode `CreateConversationTab` pour créer l'onglet correspondant à la conversation.

Vous avez prévu un champ de texte et un bouton pour envoyer des messages à vos contacts. Vous allez donc gérer l'événement `Click` du bouton **Envoyer** pour envoyer le message au sein de la conversation de l'onglet sélectionné. De plus, vous allez gérer l'événement `KeyDown` du contrôle `MonTexte` afin que la touche Entrée serve aussi à envoyer le message.

```
Private Sub Envoyer_Click(ByVal sender As System.Object, _
  ByVal e As System.EventArgs) Handles Envoyer.Click
    If Conversations.SelectedTab IsNot Nothing And _
       MonTexte.Text.Length > 0 Then
     Dim dialogue As Dialogue = _
       Conversations.SelectedTab.Controls(0)
     Dim conversation As Conversation = _
       dialogue.Conversation

     If Not conversation.SwitchboardProcessor.Connected Then
     conversation.Messenger.Nameserver.RequestSwitchboard( _
       conversation.Switchboard, Messenger)
     End If

     Dim message As New TextMessage(MonTexte.Text)

     conversation.Switchboard.SendTextMessage(message)
     MonTexte.Clear()
     dialogue.AppendText("Moi : " & message.Text & vbCrLf)
    End If
End Sub

Private Sub MonTexte_KeyDown(ByVal sender As Object, _
  ByVal e As System.Windows.Forms.KeyEventArgs) _
  Handles MonTexte.KeyDown
    If e.KeyCode = Keys.Enter Then
        Envoyer.PerformClick()
        e.SuppressKeyPress = True
    End If
End Sub
```

▲ Gestionnaires chargés de l'envoi des messages dans la conversation courante

Lorsque l'utilisateur clique sur bouton **Envoyer**, on vérifie qu'il y a bien un onglet sélectionné et que du texte a été saisi dans le champ `MonTexte`. Si c'est le cas, on récupère le contrôle `Dialogue` courant en utilisant la propriété `Controls` de l'onglet sélectionné, ainsi que la conversation courante en utilisant la propriété de même nom du contrôle `Dialogue` courant. On recourt ensuite à la méthode `SendTextMessage` de la propriété `Switchboard` de la conversation pour envoyer le message et l'on met à jour le contrôle `Dialogue` avec sa méthode `AppendText`.

Pour permettre l'envoi des messages avec la touche Entrée, on teste simplement si la touche a été utilisée en vérifiant la valeur de la propriété `KeyCode`. Si c'est le cas, on fait appel à la méthode `PerformClick` du bouton `Envoyer` pour simuler un clic et l'on supprime le retour à la ligne en utilisant la propriété `SuppressKeyPress`.

Finalement, vous fermerez les conversations que vous ne désirez plus en gérant l'événement `DoubleClick` du `TabControl` :

```
Private Sub Conversations_DoubleClick(ByVal sender As System.Object, ByVal e
➥ As System.EventArgs) Handles Conversations.DoubleClick
    Conversations.TabPages.Remove(Conversations.SelectedTab)
End Sub
```

▲ Fermeture des onglets

Il suffit d'utiliser la méthode `Remove` de la collection `TabPages` pour supprimer l'onglet courant, que l'on retrouve grâce à la propriété `SelectedTab`.

Vous disposez désormais d'une application basique vous permettant de dialoguer avec vos contacts MSN.

10.5 Check-list

Dans ce chapitre, vous avez appris à :

- vous connecter au service de messagerie MSN ;
- utiliser des classes d'une bibliothèque tierce écrite dans un autre langage .NET ;
- mettre à jour des éléments de l'interface graphique à travers différents threads ;
- instancier et ajouter dynamiquement des contrôles utilisateurs à votre interface graphique.

Explorateur de disques

Vous allez créer dans ce chapitre un Explorateur de disques. Cet outil vous permettra de connaître, d'un coup d'œil, les informations concernant vos lecteurs de disque, qu'ils s'agissent de lecteur de disque dur, de lecteur CD, de lecteur DVD ou encore de lecteur réseau.

Ce chapitre sera l'occasion de découvrir tout ce qui concerne la manipulation du système de fichiers avec Visual Basic 2005.

11.1 Classes et espaces de noms utilisés

Vous allez utiliser plusieurs espaces de noms importants.

L'espace de noms `My` permet la manipulation du système de fichiers.

Pour en savoir plus sur My, lisez le chapitre ***Moniteur de performances****.*

`System.IO` propose un grand nombre de classes destinées à la manipulation de dossiers et de fichiers.

Vous allez créer une application Windows et forcément utiliser l'espace de noms `System.Windows.Forms`. Il contient toutes les classes et structures nécessaires à la création d'interfaces Windows grâce à des fenêtres et à l'utilisation de contrôles.

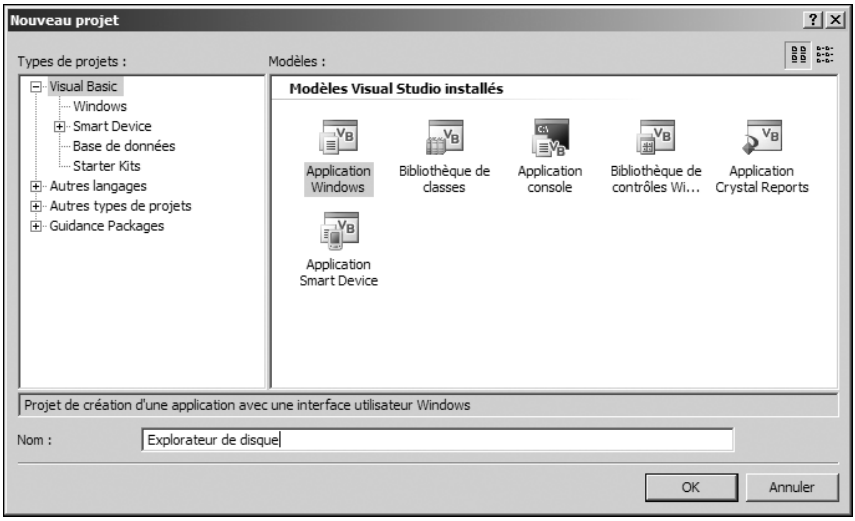
11.2 Lister les lecteurs de l'ordinateur

Commencez par récupérer la liste des lecteurs présents sur votre ordinateur.

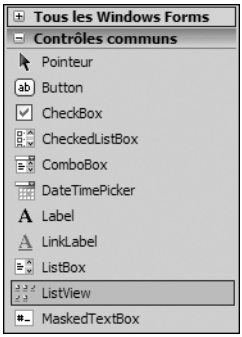
Avant de débuter, créez un nouveau projet dans Visual Basic 2005 Express Edition, et sélectionnez le type de projet intitulé *Application Windows* (voir Figure 11-1).

Vous arrivez alors sur votre surface de travail avec le Concepteur de formulaires Windows ouvert sur le formulaire par défaut.

Listez les lecteurs de votre ordinateur dans un contrôle `ListView`. Ce contrôle est exactement le même que celui utilisé par l'Explorateur de fichiers de Windows pour afficher la liste des dossiers et fichiers présents dans un dossier spécifique. `ListView` permet d'afficher des éléments dans plusieurs vues, entre autres une vue "grandes icônes", une vue "petites icônes", une vue "détail" (voir Figure 11-2).



▲ **Figure 11-1 :** Création d'application Windows



◄ **Figure 11-2 :** Contrôle ListView dans la boîte à outils

Placez un contrôle `ListView` sur votre fenêtre principale et changez la propriété `View` pour définir son style de vue à `SmallIcon`.

Le contrôle `ListView` a la particularité de pouvoir être lié à un contrôle `ImageList`. `ImageList` est un contrôle non visuel, qui permet de stocker une liste d'images, par exemple une liste d'icônes. Utilisé conjointement avec un contrôle `ListView` ou un contrôle `TreeView`, `ImageList` permet de définir l'image à afficher pour un élément précis contenu dans l'un de ces contrôles.

Dans cet exemple, `ImageList` va permettre de stocker une icône de disque dur et une icône de lecteur CD, qui seront utilisées lorsque le lecteur correspondra à l'un de ces types de supports.

Placez donc un contrôle ImageList sur votre formulaire. Il apparaît alors dans la zone des contrôles non visuels. Affichez le smart tag de ce contrôle grâce au petit triangle présent en haut à droite de celui-ci, pour définir les propriétés des différentes images insérées dans ImageList (taille, résolution) et ajouter les icônes correspondant aux lecteurs.

Remarque

Icônes

Si vous ne possédez pas de telles icônes, allez sur Internet : de nombreux sites, comme **www.icomania.com**, en proposent gratuitement.

Une fois les images ajoutées au contrôle ImageList, il ne reste plus qu'à lier celui-ci au contrôle ListView précédemment inséré sur le formulaire. Pour cela, modifiez la propriété SmallImageList du contrôle ListView et sélectionnez le contrôle ImageList.

À présent, listez la liste des lecteurs. Pour cela, utilisez l'espace de noms My, et plus particulièrement la propriété FileSystem de l'objet Computer. En effet, l'objet FileSystem permet de récupérer des informations sur le système de fichiers, et notamment la liste des lecteurs présents sur l'ordinateur. Cette liste inclut les différents disques durs de votre machine, les lecteurs CD/DVD, les lecteurs réseau, ou encore tous les lecteurs connectés occasionnellement, tels que les clés USB ou encore les cartes Compact Flash, SD ou MemoryStick.

La propriété Drives de l'objet FileSystem permet de récupérer la liste des lecteurs du système. Elle renvoie une liste d'objets DriveInfo, qui permettent de récupérer les propriétés des lecteurs, telles que leur lettre, le nom du volume, leur taille, leur type, etc.

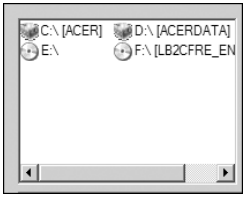
Vous devez écrire une boucle pour lister les lecteurs. Une boucle For… Each est idéale : elle permet d'effectuer une énumération d'objets dans un tableau ou une liste, sans risque de boucle infinie :

```
  For Each drive As DriveInfo In _
My.Computer.FileSystem.Drives
            Dim imageindex As Integer = 2
            Select Case drive.DriveType
                Case DriveType.CDRom
                    imageindex = 1
                Case DriveType.Fixed
                    imageindex = 0
            End Select
If drive.IsReady Then
lstDrives.Items.Add(drive.Name & _
" [" & drive.VolumeLabel & "]", imageindex)
```

```
            Else
                lstDrives.Items.Add(drive.Name, imageindex)
            End If
        Next
```

Dans cette boucle, on sélectionne l'index de l'image présente dans le contrôle `ImageList` en fonction du type de lecteur défini grâce à la propriété `DriveType`, puis on ajoute le lecteur trouvé dans la liste des éléments du contrôle `ListView` en affichant la lettre du lecteur ainsi que le nom du volume entre crochets.

Placez-ce code dans l'événement `Load` du formulaire.



◀ **Figure 11-3 :** Liste des lecteurs

Les différents lecteurs sont à présents lister dans le contrôle `ListView`. Vous allez afficher les informations détaillées concernant le lecteur sélectionné dans `ListView`.

Pour cela, ajoutez un contrôle `GroupBox`, qui permettra de regrouper tous les contrôles d'affichage de données liées aux disques, puis sept contrôles `Label` qui permettront d'afficher la lettre du lecteur, l'espace libre à la disposition de l'utilisateur courant, l'espace libre total, la taille totale du disque, le nom de volumes du disque, le format de disque et l'état du lecteur (prêt ou non). Nommez-les en modifiant leur propriété `Name` et en affectant respectivement les valeurs suivantes : `lblLettreLecteur`, `lblEspaceLibre`, `lblEspaceLibreTotal`, `lblNomLecteur`, `lblFormatLecteur`, `lblLecteurPret`.

Créez sept autres contrôles `Label` permettant d'indiquer à l'utilisateur les données affichées par les champs précédemment créés.

Puis, pour finir, ajoutez un contrôle `ProgressBar`, qui permettra d'afficher le pourcentage d'espace occupé sur le disque. Nommez ce contrôle `prgEspaceOccupe` en changeant sa propriété `Name`.

Pour afficher les informations liées au disque sélectionné, il est nécessaire de récupérer le bon objet `DriveInfo` renvoyé par la propriété `Drives` de l'objet `FileSystem`. Cela est relativement simple : il suffit de récupérer l'indice de l'élément sélectionné dans la `ListView` et de l'utiliser comme index dans la collection retournée par la propriété `Drives` pour récupérer la bonne instance :

```
My.Computer.FileSystem.Drives(lstDrives.SelectedIndices(0))
```

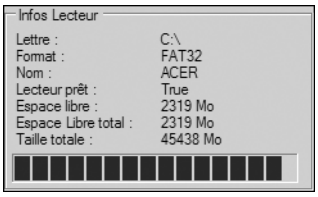
Il ne reste plus qu'à écrire le code d'affichage de données dans l'événement `SelectedIndexChanged` du contrôle `ListView` pour arriver au comportement souhaité :

```
Dim espaceLibre As Integer = 0
Dim espaceLibreTotal As Integer = 0
Dim espaceTotal As Integer = 0

If lstDrives.SelectedIndices.Count = 0 Then Exit Sub
With My.Computer.FileSystem.Drives( _
lstDrives.SelectedIndices(0))
espaceLibre = Math.Round( _
.AvailableFreeSpace / (1024 * 1024), 0)
espaceLibreTotal = Math.Round( _
.TotalFreeSpace / (1024 * 1024), 0)
espaceTotal = Math.Round( _
.TotalSize / (1024 * 1024), 0)
lblEspaceLibre.Text = espaceLibre & " Mo"
lblEspaceLibreTotal.Text = espaceLibreTotal & " Mo"
lblFormatlecteur.Text = .DriveFormat
lblLecteurPret.Text = .IsReady
lblLettreLecteur.Text = .Name
lblTailleTotale.Text = espaceTotal & " Mo"
lblNomLecteur.Text = .VolumeLabel

prgEspaceOccupe.Value = _
100 - (espaceLibreTotal / espaceTotal) * 100
        End With
```

Après s'être assuré qu'un élément est bien sélectionné en vérifiant que la propriété `SelectedIndices.Count` de la `ListView` est différente de 0, on affiche les données en modifiant la propriété `Text` des différents contrôles `Label` précédemment créés.

Les différentes tailles étant définies en octets, il faut les diviser par 1 024 au carré pour avoir un résultat en mégaoctets. Un petit arrondi est nécessaire pour ne pas avoir de décimales. On recourt en ce sens à la méthode `Round` de la classe `Math`.

Enfin, pour afficher le pourcentage de disque occupé, on calcule le pourcentage d'espace libre que l'on retranche à 100.



◄ **Figure 11-4 :** Affichage des informations sur le lecteur sélectionné

11.3 Lister les dossiers

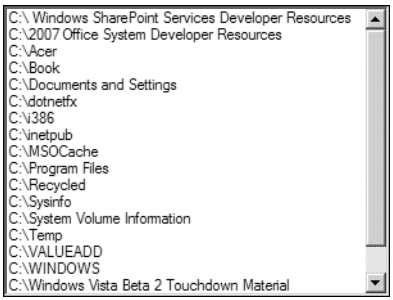
Vous allez à présent lister les dossiers du lecteur sélectionné. Commencez par créer un contrôle `ListBox` qui contiendra la liste des différents dossiers inclus dans la racine du lecteur sélectionné.

Nommez ce contrôle `lstFolders` et placez-le en dessous des contrôles que vous avez auparavant créés pour afficher la liste des lecteurs et les informations afférentes.

Utilisez ensuite l'espace de noms `My` comme précédemment. Mais au lieu d'exploiter la propriété `Drives`, qui permet de récupérer la liste des lecteurs, employez la méthode `GetDirectories`, qui renvoie une collection de `String`. Il faut donc écrire quelques lignes de code dans l'événement `SelectedIndexChanged` du contrôle `ListView`, qui liste les lecteurs, pour modifier la liste des dossiers dès que l'utilisateur sélectionne un autre lecteur :

```
Dim cu_drive As DriveInfo
        cu_drive =
        ➥ My.Computer.FileSystem.Drives(lstDrives.SelectedIndices(0))
   lstFolders.Items.Clear()
        For Each directory As String In
        ➥ My.Computer.FileSystem.GetDirectories(cu_drive.Name)
            Dim dirinfo As New DirectoryInfo(directory)
            lstFolders.Items.Add(dirinfo.FullName)
        Next
```

On énumère la liste des différents dossiers grâce à une boucle `for… each`. Puis, pour chaque dossier trouvé, on ajoute son nom complet dans le contrôle `ListBox` en utilisant la méthode `Add` de la collection `Items`, qui contient la liste des différents éléments affichés, après avoir supprimé tous les éléments de la liste en appelant la méthode `Clear`.



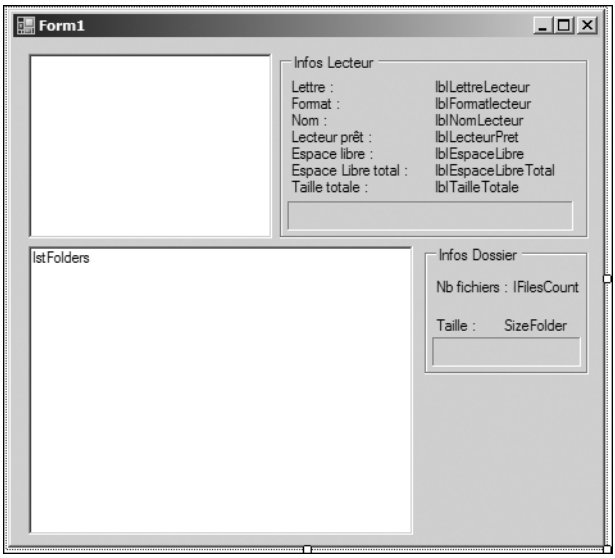
◄ **Figure 11-5 :** Affichage de la liste des dossiers

11.4 Afficher des informations sur les dossiers

Vous allez à présent afficher des informations sur le dossier sélectionné. Vous récupérerez le nombre de fichiers présents dans le dossier (sous-dossiers inclus), la taille du dossier en cumulant la taille de tous les fichiers qu'il contient. Vous afficherez la taille en mégaoctets dans un contrôle `Label` et le pourcentage d'espace occupé par le dossier par rapport à la taille totale du disque sélectionné grâce à un contrôle `ProgressBar`.

Commencez par ajouter un contrôle `GroupBox`, qui permettra de regrouper les informations liées au dossier sélectionné. Ajoutez deux contrôles `Label`, le premier nommé `lblFilesCount`, pour afficher le nombre de fichiers inclus dans le dossier, et le second nommé `lblSizeFolder`, pour afficher la taille du dossier.

Ajoutez ensuite un contrôle `ProgressBar` nommé `prgFolderSize`, pour afficher le taux d'occupation.



▲ **Figure 11-6 :** Interface de l'application

Lister les différents fichiers présents dans un dossier et dans ses sous-dossiers n'est habituellement pas une tâche évidente : il est en effet nécessaire d'écrire une fonction récursive pour parcourir toute l'arborescence. Bien heureusement et encore une fois, Visual Basic 2005 vous vient en aide. Un simple appel de la méthode `GetFiles` permet d'effectuer la recherche :

```
Dim files As ReadOnlyCollection(Of String)
files = My.Computer.FileSystem.GetFiles( _
e.Argument.ToString, _
FileIO.SearchOption.SearchAllSubDirectories, "*.*")
```

Il ne reste plus qu'à parcourir ce tableau pour cumuler les tailles des fichiers et calculer le nombre de ces fichiers :

```
        For Each file As String In files
            m_cuPathSize += _
            My.Computer.FileSystem.GetFileInfo(file).Length
            m_cuPathFilesCount += 1
        Next
```

Pour afficher convenablement la taille totale des fichiers, il est préférable de convertir la taille trouvée, qui est en octets, en mégaoctets, voire en gigaoctets si cela est possible. 1 Ko étant égal à 1 024 octets, un petit calcul suffit pour afficher le volume total :

```
        m_cuPathSize = _
        Math.Round(m_cuPathSize / (1024 * 1024), 0)

Dim driveSize As Double
        driveSize = My.Computer.FileSystem.Drives( _
        ➥ lstDrives.SelectedIndices(0)).TotalSize / (1024 * 1024)
        prgFolderSize.Value = _
        m_cuPathSize / driveSize * 100
        If m_cuPathSize / 1024 > 1 Then
            m_cuPathSize = m_cuPathSize / 1024
            lblSizeFolder.Text = _
            Math.Round(m_cuPathSize, 1) & " Go"
        Else
            lblSizeFolder.Text = m_cuPathSize & " Mo"
        End If
```

Cela donne au final :

```
Private Sub lstFolders_SelectedIndexChanged( _
ByVal sender As System.Object, _
ByVal e As System.EventArgs) _
Handles lstFolders.SelectedIndexChanged
        Me.Cursor = Cursors.WaitCursor
  Dim files As ReadOnlyCollection(Of String)
files = My.Computer.FileSystem.GetFiles( _
e.Argument.ToString, _
FileIO.SearchOption.SearchAllSubDirectories, "*.*")
        For Each file As String In files
m_cuPathSize += _
```

```
My.Computer.FileSystem.GetFileInfo(file).Length
m_cuPathFilesCount += 1
        Next

        m_cuPathSize =
        Math.Round(m_cuPathSize / (1024 * 1024), 0)

Dim driveSize As Double
driveSize = My.Computer.FileSystem.Drives(
➥ lstDrives.SelectedIndices(0)).TotalSize / (1024 * 1024)
prgFolderSize.Value = m_cuPathSize / driveSize * 100
        If m_cuPathSize / 1024 > 1 Then
            m_cuPathSize = m_cuPathSize / 1024
            lblSizeFolder.Text = Math.Round(m_cuPathSize, 1) & " Go"
        Else
            lblSizeFolder.Text = m_cuPathSize & " Mo"
        End If
        lblFilesCount.Text = m_cuPathFilesCount
        lstFolders.Enabled = True
        Me.Cursor = Cursors.Default
        m_cuPathSize = 0
        m_cuPathFilesCount = 0
    End Sub
```

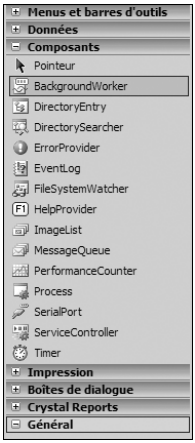
Si vous testez à présent l'application, vous vous rendez compte que tout fonctionne à une exception près : durant le parcours d'un dossier conséquent, l'interface est figée. Vous vous en apercevez si vous tentez de déplacer la fenêtre.

Cela est normal : le traitement est effectué dans le même thread que celui utilisé par l'interface graphique ; durant l'exécution d'une procédure qui prend du temps, le thread est occupé et ne peut donc plus répondre aux actions de l'utilisateur.

11.5 Composant BackgroundWorker

La solution de ce problème est relativement simple : il suffit de créer un autre thread pour lancer le traitement. La mise en œuvre d'un thread supplémentaire est aisée. Mais Visual Basic 2005 propose un composant qui simplifie encore plus la manipulation : `BackgroundWorker` (voir Figure 11-7).

Le composant `BackgroundWorker` est un contrôle non visuel, qui permet de lancer un traitement dans un thread séparé. Le code exécuté dans ce nouveau thread devra être placé dans l'événement `DoWork` du composant, le code de finalisation pouvant être écrit dans l'événement `RunWorkerCompleted`



◄ **Figure 11-7 :** Composant BackgroundWorker dans la boîte à outils

Attention

Pour modifier les propriétés des contrôles de la fenêtre

Le traitement présent dans l'événement `DoWork` étant effectué dans un thread différent de celui utilisé pour le dessin de l'interface graphique, il est impossible d'accéder directement aux différents contrôles de la fenêtre pour en modifier les propriétés. Vous devrez adapter votre code pour faire appel à votre interface graphique depuis l'événement `RunWorkerCompleted`, ou alors vous devrez utiliser les délégués.

En accord avec la précédente remarque, le code réorganisé présent dans l'événement `DoWork` correspond à ceci :

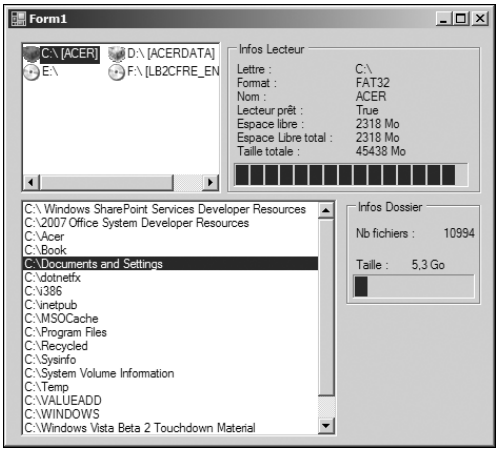
```
  Private Sub BackgroundWorker1_DoWork( _
ByVal sender As System.Object, ByVal e As
➥ System.ComponentModel.DoWorkEventArgs) _
Handles BackgroundWorker1.DoWork

        Dim files As ReadOnlyCollection(Of String)
        files = My.Computer.FileSystem.GetFiles( _
        e.Argument.ToString, _
        FileIO.SearchOption.SearchAllSubDirectories, "*.*")
        For Each file As String In files
            m_cuPathSize += _
            My.Computer.FileSystem.GetFileInfo(file).Length
            m_cuPathFilesCount += 1
        Next
```

```
        m_cuPathSize =
    Math.Round(m_cuPathSize / (1024 * 1024), 0)

    End Sub
```

Voici le code d'accès à l'interface graphique présent dans l'événement RunWorkerCompleted :

```
Private Sub BackgroundWorker1_RunWorkerCompleted( _
ByVal sender As System.Object, ByVal e As
➥ System.ComponentModel.RunWorkerCompletedEventArgs) Handles
➥ BackgroundWorker1.RunWorkerCompleted
        Dim driveSize As Double
        driveSize = My.Computer.FileSystem.Drives(
        ➥ lstDrives.SelectedIndices(0)).TotalSize / (1024 * 1024)
        prgFolderSize.Value = m_cuPathSize / driveSize * 100
        If m_cuPathSize / 1024 > 1 Then
            m_cuPathSize = m_cuPathSize / 1024
            lblSizeFolder.Text = Math.Round(m_cuPathSize, 1) & " Go"
        Else
            lblSizeFolder.Text = m_cuPathSize & " Mo"
        End If
        lblFilesCount.Text = m_cuPathFilesCount
        lstFolders.Enabled = True
        Me.Cursor = Cursors.Default
        m_cuPathSize = 0
        m_cuPathFilesCount = 0
    End Sub
```

L'application est à présent fonctionnelle et l'interface graphique ne se fige plus lors de longs traitements.



▲ **Figure 11-8 :** Résultat final de l'application

11.6 Check-list

Dans ce chapitre vous avez appris à :

- manipuler les lecteurs, dossiers et fichiers ;
- utiliser le composant `BackgroundWorker` pour effectuer des traitements dans un thread séparé ;
- utiliser le contrôle `Listview` et le lier à un contrôle `ImageList`.

Navigateur Internet

Vous allez à présent créer votre propre navigateur Internet. Cela est pratique si vous souhaitez disposer de votre propre outil personnalisé, qui sera plus adapté à votre utilisation ou à celle de votre entourage. Vous allez donc créer les bases d'un navigateur pleinement fonctionnel bien que simplifié, qui aura pour originalité de proposer la navigation par onglet. Vous allez également implémenter quelques fonctionnalités bien utiles si vous souhaitez mettre en place un contrôle parental : le stockage des sites visités dans une base de données, ainsi que le blocage de certaines URL, qui seront donc inaccessibles aux utilisateurs du navigateur.

12.1 Classes et espaces de noms utilisés

Vous allez devoir utiliser principalement trois différents espaces de noms :

- `System.Windows.Forms`, pour la manipulation des formulaires et des contrôles ;
- `System.Data`, pour la gestion de données stockées dans la base de données ;
- `System.IO`, pour la manipulation de fichiers.

12.2 Configuration

En plus d'utiliser Visual Basic 2005 Express Edition, vous allez exploiter SQL Server 2005 Express Edition. Cet outil est un système de gestion de bases de données (SGBD) gratuit proposé par Microsoft. Il s'agit en fait du moteur du SGBD phare du géant de Redmond avec quelques limitations au niveau de la taille des bases de données, de la mémoire utilisée, du nombre de processeurs pris en charge, etc.

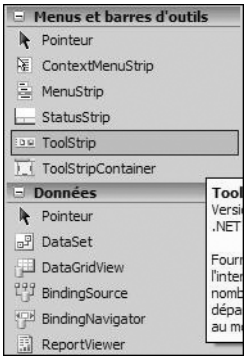
Vous pouvez télécharger cet outil en version française à l'adresse suivante : **www.microsoft.com/france/msdn/vstudio/express/sqlexpress.mspx**.

Il ne s'agit que du moteur de gestion de bases de données. Il n'est donc livré avec aucun outil d'administration permettant de créer et de modifier les différentes bases de données que vous souhaitez gérer.

Vous avez cependant deux solutions pour gérer vos bases de données. La première est de recourir aux fonctionnalités offertes par Visual Basic 2005 Express Edition afin d'effectuer toutes vos opérations courantes, et la seconde est de télécharger un outil additionnel appelé SQL Server Management Studio Express, disponible gratuitement mais uniquement en anglais à l'adresse **http://go.microsoft.com/fwlink/?LinkId=65110**.

12.3 Interface utilisateur

L'interface de votre navigateur Internet est relativement simple. Afin de la réaliser, vous allez vous inspirer de celle proposée par Microsoft Internet Explorer, tout en lui ajoutant les fonctionnalités citées précédemment. Le formulaire se compose principalement de deux barres d'outils (la première pour les boutons de navigation, et la seconde pour la barre d'adresse). Afin de créer ces deux barres d'outils, il faut ajouter deux contrôles `ToolStrip`. Lorsque vous les placez sur votre formulaire, vous pouvez constater qu'elles sont automatiquement disposées en haut de la fenêtre. Il suffit alors de cliquer sur ces barres d'outils afin d'y ajouter différents contrôles tels que des boutons (le plus courant), des séparateurs de groupe, des labels, ou encore des `TextBox`.



◄ **Figure 12-1 :** Contrôles ToolStrip dans la boîte à outils

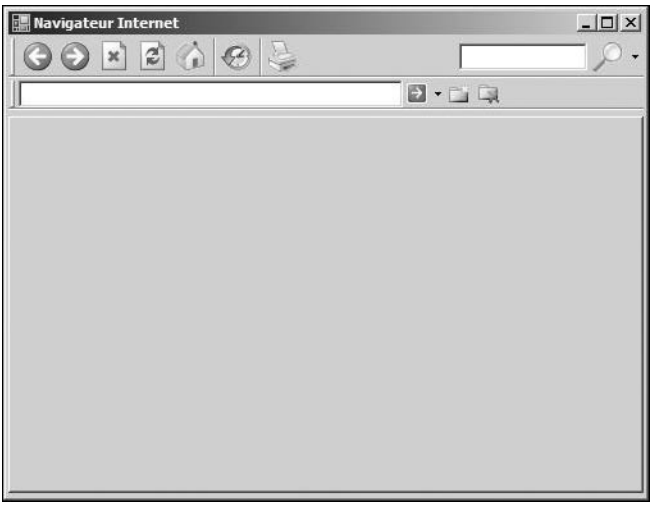
La première barre d'outils propose différentes possibilités de navigation à l'utilisateur. Placez-y donc, pour débuter, cinq boutons correspondant aux actions **Précédent**, **Suivant**, **Arrêter**, **Actualiser** et **Page de démarrage**. Introduisez ensuite un séparateur de groupe suivi d'un bouton pour afficher l'historique, lui-même suivi d'un autre séparateur de groupe. Proposez à présent à l'utilisateur d'imprimer la page courante grâce à un bouton **Imprimer**. Pour compléter cette première barre d'outils, vous allez ajouter une fonctionnalité absente du navigateur de Microsoft : la recherche rapide. Ajoutez une `Textbox` ainsi qu'un `SplitButton` et modifiez la propriété `Alignment` de ces deux contrôles afin de définir sa valeur égale à `Right`. Ajoutez à présent deux éléments de menu au `SplitButton` que vous avez ajouté et définissez leur propriété `Text` à `Google` et `MSN`. Modifiez également la propriété `CheckOnClick` de ces deux éléments : cela permettra de gérer automatiquement l'état coché ou non des éléments sans écrire aucune ligne de code.

▲ **Figure 12-2 :** Barre d'outils du navigateur

Ajoutez à présent les contrôles au second `ToolStrip`. Pour cela, introduisez un contrôle `ToolStripTextBox` qui servira de barre d'adresse, un contrôle `ToolStripSplitButton` qui permettra à l'utilisateur d'afficher le site web correspondant à l'adresse saisie sous l'onglet courant ou sous un nouvel onglet. Terminez ensuite par deux `ToolStripButton`, le premier pour créer un nouvel onglet et le second pour fermer, supprimer l'onglet courant.

Il ne reste plus qu'à ajouter un seul et unique contrôle pour terminer l'interface de l'application : un `TabControl`. Lorsque vous l'insérez sur le formulaire, le designer de Visual Basic 2005 Express Edition crée automatiquement deux onglets, qui sont des contrôles `TabPage`. Supprimez-les en modifiant la propriété `TabPages` du contrôle `TabControl` ou en cliquant sur le lien *Supprimer l'onglet* dans la fenêtre des propriétés. Il est en effet inutile de garder des onglets sur le contrôle `TabControl` puisque vous allez implémenter une gestion dynamique qui va permettre à l'utilisateur de gérer lui-même les onglets comme il le souhaite.

L'interface est à présent terminée.



▲ **Figure 12-3 :** Interface finale de l'application

12.4 Réalisation

Passons à présent au développement de l'application. Vous allez apprendre dans la suite de ce chapitre à gérer des onglets dynamiquement et à accéder à une base de données SQL Server 2005 Express.

Gestion dynamique des onglets

Pour ajouter des onglets contenant un contrôle `WebBrowser` nécessaire à l'affichage d'une page web, vous devez créer dynamiquement des contrôles et les ajouter au contrôle `TabControl` de votre formulaire. Cette gestion dynamique est assez simple à mettre en place même si elle introduit quelques complexités que nous aborderons par la suite. La clé pour créer dynamiquement des contrôles est de les ajouter aux contrôles parents existants (un formulaire, un `TabControl`…) afin de les inclure dans l'arborescence des contrôles de la fenêtre et de pouvoir les gérer correctement. Cet ajout se fait généralement grâce à la propriété `Controls` d'un contrôle. Dans le cas d'un contrôle `TabControl`, il faut utiliser la propriété `TabPages`, qui permet d'accéder directement à la collection des onglets d'un `TabControl`.

La création dynamique d'onglets incluant un contrôle `WebBrowser` peut donc se faire via ce code :

```
Private Sub CreerOnglet(ByVal url As String)
        Dim newTabPage As New TabPage
        Dim newWebBrowser As New WebBrowser
        newWebBrowser.Dock = DockStyle.Fill
        AddHandler newWebBrowser.Navigated, _
            AddressOf Browser_Navigated
        newWebBrowser.Navigate(url)
        newTabPage.Controls.Add(newWebBrowser)
        TabControl1.TabPages.Add(newTabPage)
        TabControl1.SelectTab(newTabPage)
    End Sub
```

La procédure `CreerOnglet` permet de créer un onglet affichant le site web correspondant à l'URL passée en paramètre de cette procédure.

La première ligne de ce code permet d'instancier un contrôle `TabPage` représentant un onglet. On créé ensuite un contrôle `WebBrowser` et on modifie sa propriété `Dock` afin d'adapter la taille du contrôle navigateur dynamiquement en fonction de la taille de la fenêtre. La quatrième ligne de cette procédure permet de gérer dynamiquement l'événement `Navigated` du contrôle `WebBrowser` créé dynamiquement. En effet, comme vous créez vous-même vos contrôles durant l'exécution du programme, il est impossible de modifier leurs propriétés grâce à la fenêtre de propriétés de Visual Basic 2005 Express Edition et de gérer les événements automatiquement comme on peut le faire d'habitude. Il faut donc par programmation refaire tout ce que l'environnement de développement permet de réaliser de manière "visuelle". La gestion des événements peut se faire grâce au mot-clé `AddHandler`. Il permet de lier un événement, dans ce cas l'événement `Navigated` qui est déclenché lorsque le chargement d'une page est terminé, à une fonction, dans cet exemple `Browser_Navigated`. On définit ce lien en créant un pointeur de fonction grâce au mot-clé `AddressOf`.

Cet événement est donc géré grâce à une méthode appelée Browser_Navigated, qui permet de modifier le titre de l'onglet en fonction du titre de la page chargée.

```
Private Sub Browser_Navigated( _
    ByVal sender As System.Object, _
    ByVal e As _
    System.Windows.Forms.WebBrowserNavigatedEventArgs)
        Dim cu_browser As WebBrowser
        cu_browser = sender
        cu_browser.Parent.Text = cu_browser.DocumentTitle
    End Sub
```

On récupère simplement le contrôle Browser qui a déclenché l'événement grâce au paramètre sender de la fonction. On change la propriété Text du contrôle parent du contrôle Browser qui, dans cette application, est forcément un contrôle TabPage correspondant à un onglet, en lui affectant la propriété DocumentTitle du contrôle WebBrowser. Cette propriété retourne le titre du document chargé, c'est-à-dire le titre de la page web chargée par le contrôle.

La suite de la procédure est assez simple à comprendre. On appelle la méthode Navigate du contrôle WebBrowser afin de charger l'URL choisie et l'on ajoute les différents contrôles aux contrôles parents afin que l'arborescence soit correcte et que tout fonctionne parfaitement.

Pour tester ce code, il ne reste plus qu'à l'appeler dans l'événement Load du formulaire :

```
Private Sub Form1_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
        CreerOnglet("about:home")
    End Sub
```

Ces lignes créent un premier onglet qui affiche la page de démarrage de votre navigateur dès le chargement de la fenêtre (et donc de l'application).

Afin de compléter cette gestion d'onglets, vous pouvez ajouter du code dans l'événement click des deux derniers boutons de la deuxième barre d'outils. Ces boutons servent respectivement à ajouter un onglet et à supprimer l'onglet courant. Le code correspondant est relativement simple :

```
Private Sub ToolStripNouvelOnglet_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs)
    Handles ToolStripNouvelOnglet.Click
        CreerOnglet("about:home")
    End Sub
```

```
Private Sub ToolStripButtonSupprOnglet_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs)
    Handles ToolStripButtonSupprOnglet.Click
        If TabControl1.SelectedTab IsNot Nothing And _
        TabControl1.TabPages.Count > 1 Then
            TabControl1.TabPages.Remove( _
                TabControl1.SelectedTab)
        End If
    End Sub
```

Le seul point d'explication nécessaire concerne le code correspondant au bouton de suppression de l'onglet courant. On teste l'existence d'un onglet sélectionné et l'on vérifie que l'on a bien plus d'un onglet présent dans le contrôle `TabControl`. On supprime ensuite simplement l'onglet sélectionné grâce à la méthode `Remove` de la collection de `TabPages` du `TabControl`.

Il faut à présent être capable de manipuler le contrôle `Browser` courant afin de pouvoir fournir les fonctionnalités de navigation. Créer une fonction permettant de récupérer ce contrôle courant est donc une solution pertinente et le code nécessaire pour réaliser cela est assez simple :

```
    Private Function GetActiveBrowser() As WebBrowser
        Return TabControl1.SelectedTab.Controls(0)
    End Function
```

Cette fonction récupère le premier contrôle enfant de l'onglet sélectionné. Comme dans cette application chaque onglet ne contient qu'un contrôle `WebBrowser`, cette fonction renvoie le contrôle `WebBrowser` de l'onglet sélectionné.

Vous pouvez donc à présent manipuler le contrôle `WebBrowser` de manière simple et effectuer toutes les opérations de navigation nécessaires grâce aux méthodes `GoBack`, `GoForward`, `Stop`, `Refresh`, `GoHome`, `ShowPrintDialog`.

```
Private Sub ToolStripButtonBack_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonBack.Click
            GetActiveBrowser.GoBack()
    End Sub

Private Sub ToolStripButtonPrevious_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonForward.Click
            GetActiveBrowser.GoForward()
    End Sub
```

```
Private Sub ToolStripButtonStop_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonStop.Click
        GetActiveBrowser.Stop()
    End Sub

Private Sub ToolStripButtonRefresh_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonRefresh.Click
        GetActiveBrowser.Refresh()
    End Sub

Private Sub ToolStripButtonHome_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonHome.Click
        GetActiveBrowser.GoHome()
    End Sub

Private Sub ToolStripButtonPrint_Click( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripButtonPrint.Click
        GetActiveBrowser.ShowPrintDialog()
    End Sub
```

De plus, afin d'avoir une meilleure ergonomie, il serait pertinent d'activer ou non les boutons **Précédent** et **Suivant** si la réalisation de ces actions est possible. Pour ce faire, vous pouvez ajouter deux lignes de code dans l'événement `SelectedIndexChanged` :

```
Private Sub TabControl1_SelectedIndexChanged(_
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles TabControl1.SelectedIndexChanged
        ToolStripButtonBack.Enabled =  _
            GetActiveBrowser.CanGoBack
        ToolStripButtonForward.Enabled =  _
            GetActiveBrowser.CanGoForward
    End Sub
```

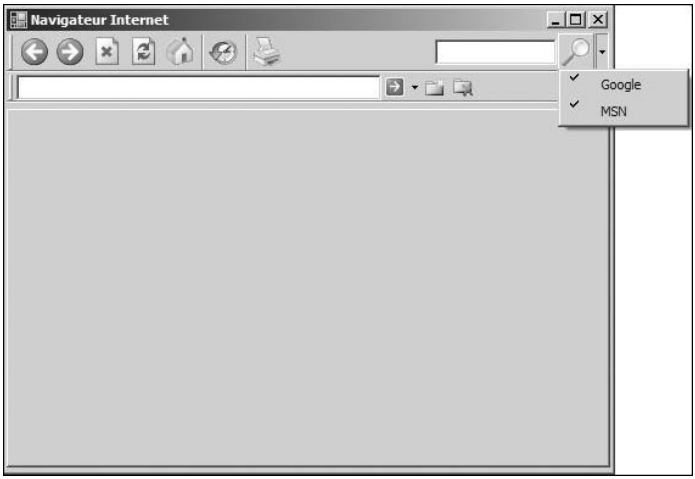
Le fonctionnement de base de votre navigateur est à présent défini.

Recherche

Voyons maintenant comment gérer la fonctionnalité de recherche. Vous allez proposer à l'utilisateur de rechercher simultanément dans plusieurs moteurs de recherche. Cette fonctionnalité n'existe dans aucun autre navigateur. Elle va être mise en place simplement. Vous avez précédemment créé un `ToolStripSplitButton` permettant de sélectionner le ou les moteurs de recherche que souhaite utiliser l'utilisateur.

Il suffit donc à présent de rechercher quels éléments sont sélectionnés et de créer de nouveaux onglets pour chaque recherche afin de permettre à l'utilisateur de rechercher simultanément dans plusieurs moteurs (dans cet exemple Google et MSN).

```
Private Sub ToolStripSplitButtonSearch_ButtonClick(_
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ToolStripSplitButtonSearch.ButtonClick
        If GoogleToolStripMenuItem.Checked Then
            CreerOnglet("http://www.google.com" + _
                "/search?q=" + ToolStripTextBoxSearch.Text)
        End If
        If MSNToolStripMenuItem.Checked Then
            CreerOnglet("http://search.msn.fr" + _
                "/results.aspx?q=" + _
                ToolStripTextBoxSearch.Text)
        End If
    End Sub
```



▲ **Figure 12-4 :** Recherche multimoteur

Gestion des raccourcis clavier

Afin d'améliorer l'ergonomie de l'application, il est préférable de proposer différents raccourcis clavier à l'utilisateur. Le plus simple pour implémenter manuellement des raccourcis clavier simples, comme la validation avec la touche Entrée, est d'écrire du code dans l'événement `KeyDown` des contrôles. Par exemple, la procédure suivante lance la recherche lorsque l'utilisateur appuie sur Entrée dans la zone de texte correspondant à la recherche située dans la première barre d'outils.

```
Private Sub ToolStripTextBoxSearch_KeyDown( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.KeyEventArgs) _
    Handles ToolStripTextBoxSearch.KeyDown
        If e.KeyCode = Keys.Enter Then
            ToolStripSplitButtonSearch_ButtonClick _
                (sender, Nothing)
        End If
    End Sub
```

L'exemple suivant permet de faire de même avec la barre d'adresse et d'ouvrir un nouvel onglet si l'utilisateur appuie sur la combinaison de touches Ctrl+T.

```
Private Sub ToolStripTxtUrl_KeyDown( _
    ByVal sender As System.Object, _
    ByVal e As System.Windows.Forms.KeyEventArgs) _
    Handles ToolStripTxtUrl.KeyDown

        If e.KeyCode = Keys.Enter Then
            ToolStripSplitButtonGo_ButtonClick _
                (sender, Nothing)
        End If

        If e.KeyCode = Keys.T And e.Control Then
            CreerOnglet(ToolStripTxtUrl.Text)
        End If

    End Sub
```

Contrôle parental

Voyons à présent comment ajouter des fonctionnalités de contrôle parental.

Pour commencer, vous allez stocker tout l'historique des visites dans la table d'historique. Cette table comprend trois champs : `id_url`, de type `BigInt` et auto-incrémenté, `url`, de type `VarChar` et qui a une longueur de 255 caractères, et `date_visite`, de type `DateTime`.

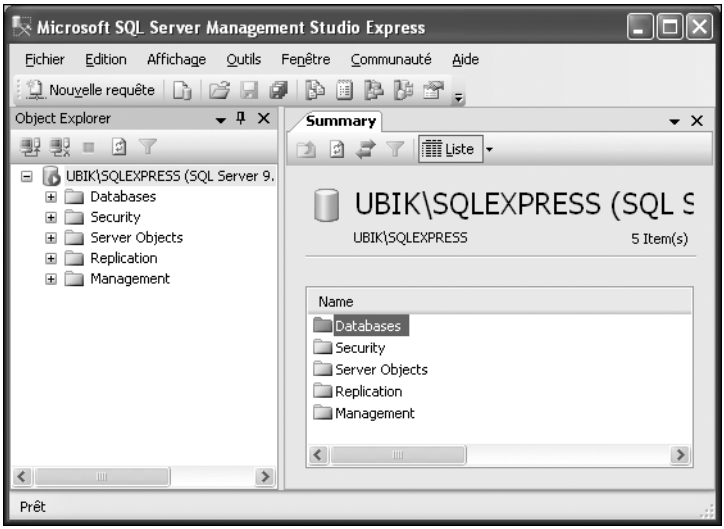
Voici le script SQL qui crée la base de données ainsi que la table :

```
USE [master]
GO
/****** Object:  Database [Navigateur]
        Script Date: 06/26/2006 06:12:11 ******/
CREATE DATABASE [Navigateur] ON  PRIMARY
( NAME = N'Navigateur',
     FILENAME = N'C:\Program Files\Microsoft SQL
         Server\MSSQL.1\MSSQL\DATA\Navigateur.mdf' ,
     SIZE = 3072KB , MAXSIZE = UNLIMITED,
     FILEGROWTH = 1024KB )
LOG ON
( NAME = N'Navigateur_log',
     FILENAME = N'C:\Program Files\Microsoft SQL
     Server\MSSQL.1\MSSQL\DATA\Navigateur_log.ldf' ,
     SIZE = 1024KB , MAXSIZE = 2048GB ,
     FILEGROWTH = 10%)
 COLLATE French_CI_AS
GO
EXEC dbo.sp_dbcmptlevel @dbname=N'Navigateur',
     @new_cmptlevel=90
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Navigateur].[dbo].[sp_fulltext_database]
     @action = 'enable'
end
GO
ALTER DATABASE [Navigateur] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Navigateur] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Navigateur] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Navigateur] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [Navigateur] SET ARITHABORT OFF
GO
ALTER DATABASE [Navigateur] SET AUTO_CLOSE OFF
GO
ALTER DATABASE [Navigateur] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [Navigateur] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [Navigateur] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [Navigateur] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
ALTER DATABASE [Navigateur] SET CURSOR_DEFAULT GLOBAL
GO
```

```
ALTER DATABASE [Navigateur] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [Navigateur] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [Navigateur] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [Navigateur] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [Navigateur] SET  ENABLE_BROKER
GO
ALTER DATABASE [Navigateur] SET
    AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [Navigateur] SET
    DATE_CORRELATION_OPTIMIZATION OFF
GO
ALTER DATABASE [Navigateur] SET TRUSTWORTHY OFF
GO
ALTER DATABASE [Navigateur] SET
    ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [Navigateur] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [Navigateur] SET  READ_WRITE
GO
ALTER DATABASE [Navigateur] SET RECOVERY SIMPLE
GO
ALTER DATABASE [Navigateur] SET  MULTI_USER
GO
ALTER DATABASE [Navigateur] SET PAGE_VERIFY CHECKSUM
GO
ALTER DATABASE [Navigateur] SET DB_CHAINING OFF

USE [Navigateur]
GO
/****** Object:  Table [dbo].[Historique]
        Script Date: 06/26/2006 06:12:43 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Historique](
  [id_url] [bigint] IDENTITY(1,1) NOT NULL,
  [url] [nvarchar](255) COLLATE French_CI_AS NOT NULL,
  [date_visite] [datetime] NOT NULL,
 CONSTRAINT [PK_Historique] PRIMARY KEY CLUSTERED
(
  [id_url] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

Afin d'exécuter ce script, démarrez SQL Server Management Express, connectez-vous en cliquant sur le bouton **Connect** dans la boîte de dialogue qui s'ouvre.

Cliquez ensuite sur le bouton **New Query** et recopiez le script SQL précédent. Validez en cliquant sur le bouton **Execute** pour que la base de données ainsi que la table d'historique soient créées.



▲ **Figure 12-5 :** SQL Server Management Studio Express

Il est maintenant nécessaire d'écrire le code qui permet de se connecter à la base de données et d'insérer les URL saisies dans la barre d'adresse.

Afin de créer la couche d'accès aux données, vous aller définir une classe intitulée `HistoriqueDAO`. Elle contiendra une méthode statique nommée `InsertIntoHistorique`, qui prendra en paramètre une chaîne contenant l'URL qui sera insérée dans l'historique.

La connexion à une base de données se fait à l'aide d'une brique du Framework .NET nommée ADO .NET. ADO .NET propose différentes classes permettant de manipuler des données. Plusieurs d'entre elles sont spécifiques au système de gestion de bases de données. Dans ce cas, vous allez utiliser une base SQL Server. Vous devez donc exploiter les classes présentes dans le namespace `System.Data.SqlClient`.

La manipulation de bases de données peut paraître compliquée au premier abord, mais nous allons procéder par étapes.

Pour manipuler des données présentes dans une base de données, il faut commencer par créer une connexion à cette base. Pour ce faire, ADO .NET propose la classe `SqlConnection`. Elle permet d'établir des connexions à des serveurs et à des bases SQL Server. Cette classe `SqlConnection` a deux propriétés importantes : la propriété `ConnectionString`, qui permet de définir la chaîne de connexion à utiliser afin de se connecter à une base de données, et la propriété `State`, qui permet de connaître l'état de la connexion (à savoir ouverte, fermée…).

La chaîne de connexion est primordiale. Toute erreur dans la création de celle-ci empêchera le code de fonctionner. Les chaînes de connexion permettant de se connecter à une base de données SQL Server 2005 Express peuvent s'écrire de différentes manières. Dans ce cas, on définit trois paramètres :

- `Data Source` : permet de définir le serveur SQL Server 2005 Express qui héberge la base de données. Dans ce cas, on spécifie la machine locale grâce à l'opérateur `.`, et l'instance de la base qui, par défaut, se nomme `SQLEXPRESS`.
- `Initial Catalog` : permet de définir la base de données qui sera utilisée par la connexion.
- `Integrated Security` : permet de définir le fait que l'on souhaite utiliser une authentification Windows, et non SQL.

La classe `SqlConnection` a également deux méthodes importantes : `Open`, qui permet d'ouvrir la connexion, et `Close`, qui permet de la fermer.

Ainsi, l'ouverture d'une connexion à une base SQL Server 2005 Express peut se faire ainsi :

```
Dim myconnection As New SqlConnection
myconnection.ConnectionString = _
    "Data Source=.\SQLEXPRESS;Initial Catalog=" _
    "Navigateur;Integrated Security=True;"
myconnection.Open()
```

Bien évidemment, ce code est susceptible de lever une exception puisque l'ouverture de la connexion peut échouer. Vous devez donc utiliser un bloc `Try…Catch` afin de créer vos connexions.

Maintenant que vous savez créer une connexion à une base, vous devez être capable d'exécuter des requêtes SQL. En ce sens, ADO .NET propose une classe nommée `SqlCommand`. Elle permet d'exécuter des requêtes SQL pour insérer des données dans une table, les modifier, les supprimer et les récupérer.

La première manipulation à faire lorsque l'on souhaite utiliser un objet `SqlCommand` est d'attacher cet objet à une connexion existante. Il faut en effet que la requête SQL soit exécutée sur une connexion, et non "dans le vide". Vous

devez donc modifier sa propriété `Connection` afin de sélectionner une connexion valide et ouverte.

Vous devez en outre modifier la propriété `CommandText` afin de définir la requête SQL à exécuter : `Select`, `Insert`, `Update` ou `Delete`. Il s'agit simplement d'écrire une requête SQL valide.

Si vous souhaitez passer des paramètres dans votre requête, comme dans cet exemple, vous devez ajouter le préfixe `@` à vos paramètres. L'utilisation de la méthode `AddWithValue` de la propriété `Parameters` permet ensuite d'ajouter les valeurs de ces paramètres assez facilement : il suffit d'indiquer le nom du paramètre dont il faut modifier la valeur et ladite valeur.

Après avoir initialisé toutes ces données importantes, il ne reste plus qu'à exécuter la requête à l'aide de la méthode `ExecuteNonQuery`.

Le code d'insertion d'URL dans l'historique est le suivant :

```
Imports System.Data
Imports System.Data.SqlClient

Public Class HistoriqueDAO
    Public Shared Sub InsertIntoHistorique(_
            ByVal url As String)
        Dim myconnection As New SqlConnection
        myconnection.ConnectionString = _
            "Data Source=.\SQLEXPRESS;Initial Catalog=" _
            "Navigateur;Integrated Security=True;"
        Try
            myconnection.Open()
            Dim cmdInsert As New SqlCommand
            cmdInsert.Connection = myconnection
            cmdInsert.CommandText = "insert into " _
                "Historique (url, date_visite) " _
                "values (@url,@date_visite)"
            cmdInsert.Parameters.AddWithValue _
                ("@url", url)
            cmdInsert.Parameters.AddWithValue _
                ("@date_visite", Date.Now)
            cmdInsert.ExecuteNonQuery()
        Catch sqlex As SqlException
            MsgBox("Impossible de stocker l'url " _
                "dans la base de données", _
                MsgBoxStyle.Exclamation, "Erreur")
        Catch ex As Exception
            MsgBox("Une erreur non gérée est survenue " + _
                ex.Message, MsgBoxStyle.Exclamation, _
                "Erreur")
        Finally
            If myconnection.State = ConnectionState.Open _
```

```
                Then
                myconnection.Close()
            End If
        End Try
    End Sub
End Class
```

Votre application est à présent terminée et fonctionnelle.

12.5 Check list

Voici les concepts et les fonctionnalités présentés dans ce chapitre :

- créer et gérer dynamiquement des contrôles ;
- se connecter et récupérer des informations d'une base de données SQL Server ;
- utiliser le contrôle WebBrowser ;
- créer des barres d'outils dans des applications Windows.

Aggrégateur RSS

Vous allez découvrir dans ce chapitre ce qu'est RSS et comment créer des pages web permettant de consulter des flux RSS de manière simple.

Définition

Flux RSS

Les flux RSS (Really Simple Syndication) sont une nouvelle manière de consulter les sites web de façon rapide. Ils existent depuis quelques années et sont incontournables.

Plus besoin d'utiliser votre navigateur préféré et de charger les sites web que vous aimez un par un : vous avez à présent la possibilité de télécharger le contenu de ces sites web (news, articles…) et de les consulter en local s'ils proposent des flux RSS.

◄ **Figure 13-1 :** Icône RSS affichée sur les pages web proposant des flux

Un flux RSS est un document XML qui regroupent les informations importantes proposées par un site web pour lister des news, des articles, etc.

Voici un exemple de flux RSS raccourci afin que vous compreniez la logique.

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml-stylesheet type="text/xsl"
href="http://blogs.developpeur.org/rss.xsl"
media="screen"?>
<rss version="2.0"
xmlns:dc="http://purl.org/dc/elements/1.1/"
➥ xmlns:slash="http://purl.org/rss/1.0/modules/slash/"
➥ xmlns:wfw="http://wellformedweb.org/CommentAPI/">
  <channel>
    <title>Patrice Lamarche</title>
<link>http://blogs.developpeur.org/patrice/default.aspx
</link>
<description>Des infos sur WinFX, .net 2, etc.
</description>
<dc:language>fr-FR</dc:language>
    <generator>CommunityServer 2.0 (Build: 60217.2664)
</generator>
    <item>
      <title>Et un de plus !</title>
      <link>http://blogs.developpeur.org/patrice/archive/
      2006/07/02/nomination_mvp.aspx</link>
      <pubDate>Sun, 02 Jul 2006 17:15:00 GMT</pubDate>
      <guid
      ➥ isPermaLink="false">7d6e5d3c-61cc-4264-bc3a-a336023fcec4:22073</guid>
      <dc:creator>patrice</dc:creator>
```

```
<slash:comments>13</slash:comments>
<comments>
 http://blogs.developpeur.org/patrice/comments/22073.aspx
</comments>
      <wfw:commentRss>
http://blogs.developpeur.org/patrice/
commentrss.aspx?PostID=22073</wfw:commentRss>
      <description>
        L'été commence bien !
      </description>
<category domain="http://blogs.developpeur.org/patrice/
archive/category/1235.aspx">Perso</category>
    </item>
    <item>
      <title>[Session] D&#233;veloppement
 avec Windows Presentation Foundation</title>
      <link>http://blogs.developpeur.org/patrice/archive/
2006/06/21/wpf_windows_vista_evenement_asp_php
_codes_sources.aspx</link>
      <pubDate>Wed, 21 Jun 2006 11:50:00 GMT</pubDate>
      <guid
      ➥ isPermaLink="false">7d6e5d3c-61cc-4264-bc3a-a336023fcec4:21766</guid>
      <dc:creator>patrice</dc:creator>
      <slash:comments>0</slash:comments>
<comments>
http://blogs.developpeur.org/patrice/comments/21766.aspx
</comments>
<wfw:commentRss>http://blogs.developpeur.org/
patrice/commentrss.aspx?PostID=21766</wfw:commentRss>
<description>
        Je vous en ai parlé lors de mon précédent post,
        j'ai eu l'occasion de donner une présentation du
        développement sur Windows Vista avec Windows
        Presentation Foundation lors de l'événement
        CodeS-SourceS/ASP-PHP.net.
        Je viens d'uploader les slides en ve…<img src=
"http://blogs.developpeur.org/aggbug.aspx?PostID=21766"
 width="1" height="1">
</description>
<category domain="http://blogs.developpeur.org/
patrice/archive/category/1189.aspx">Mes Sessions
</category>
</item>
</channel>
</rss>
```

Un flux RSS est donc bien un document XML regroupant des informations dans des balises `item`.

Chaque élément du flux est représenté par une balise `item`, elle-même incluse dans une balise `channel` qui représente l'ensemble du flux. Chaque balise `item` contient un ensemble de balises définissant l'information grâce à plusieurs caractéristiques.

Balises les plus importantes	
Nom de balise	**Description**
`title`	Contient le titre de l'information. Il s'agit en général d'une chaîne de caractères ne contenant pas d'information de formatage.
`link`	Définit le lien de la page contenant l'information. Il s'agit donc du lien qu'il faudra suivre si l'on souhaite avoir plus de détails dans le cas où l'auteur ne syndique que le résumé.
`pubDate`	Définit la date de publication de l'information.
`guid`	Définit de manière unique l'information.
`comments`	Lien pointant vers la page contenant les propriétés.
`description`	Balise contenant l'information elle-même. Contient souvent des balises HTML encodées afin de formater l'information, d'y placer des liens, etc.
`category`	Définit la ou les catégories auxquelles est rattachée l'information.

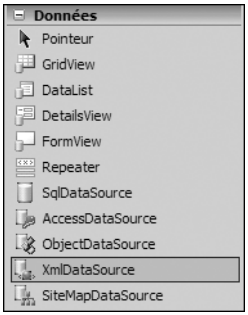
13.1 Réalisation

Afin de créer une page web capable de proposer ce genre de flux et donc d'afficher de manière intuitive un document XML, vous devez commencer par créer une source de données, puis utiliser un contrôle d'affichage pour afficher les données fournies. Cette opération est appelée "databinding".

ASP .NET 2 propose plusieurs types de sources de données :

- `SqlDataSource` permet de fournir des données en provenance de bases de données SQL relationnelles comme des bases de données SQL Server, Oracle, DB2, etc.
- `AccessDataSource` permet de récupérer des données issues de bases de données Microsoft Access.
- `ObjectDataSource` permet de fournir des données disponibles sous forme d'objets.
- `XmlDataSource` permet de fournir des données disponibles sous forme de document XML.

- `SiteMapDataSource` permet de fournir des données représentant l'arborescence du site web.

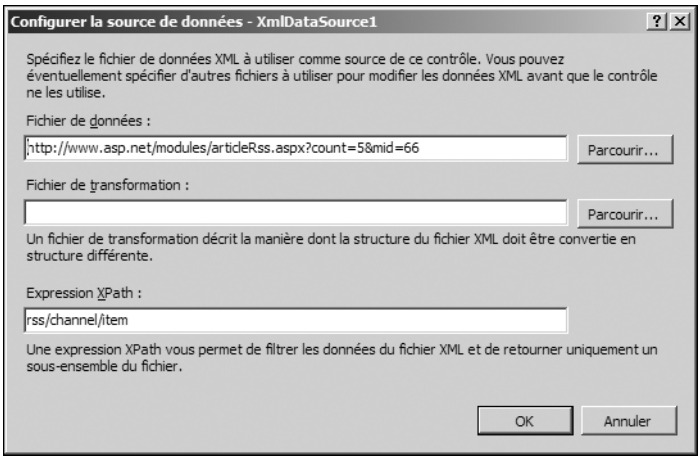


◄ **Figure 13-2 :** Contrôles Sources de données dans la boîte à outils

Vous allez devoir utiliser un contrôle `XmlDataSource` afin de récupérer des données issues d'un flux RSS, puisqu'un flux RSS est un document XML.

Pour cela, rendez-vous dans la boîte à outils, affichez la catégorie *Données* et insérez un contrôle `XmlDataSource` sur votre surface de travail.

Lorsque vous l'insérez sur le formulaire web, cliquez sur le smart tag du contrôle (le petit triangle en haut à droit du cadre gris) pour ouvrir une boîte de dialogue permettant de définir ses différentes propriétés.



▲ **Figure 13-3 :** Saisie des propriétés du contrôle XmlDataSource

Cette boîte de dialogue permet de définir le fichier de données à utiliser, c'est-à-dire le document XML qui va servir de source de données. Ce document peut être un fichier présent sur votre disque dur, ou disponible sur Internet et accessible via une simple URL. Dans ce dernier cas, ASP .NET va automati-

quement télécharger le document pour que vous puissiez le manipuler grâce au contrôle `XmlDataSource`. C'est donc cette option que vous allez choisir.

Pour débuter et à des fins de test, vous pouvez utiliser un flux RSS proposé par la version française du site **www.asp.net développé par Microsoft, et qui énumère les différents articles techniques en français traitant d'ASP .NET : www.asp.net/modules/articleRss.aspx?count=5&mid=66.**

Vous pouvez également définir un fichier de transformation. XML est en effet un standard qui est accompagné d'autres standards indissociables :

- XSD permet de définir des schémas pour définir la structure d'un document XML
- XSL permet de transformer un document XML en un autre document XML respectant un autre schéma.
- XPath est un langage de requête qui permet de naviguer au sein d'un document XML.

Le fichier de transformation attendu est donc un fichier XSL. Il peut servir à transformer un document XML en un document XHTML, lisible par des navigateurs. Ce paramètre étant facultatif, vous allez vous en passer et définir la mise en forme et la mise en page directement au sein du contrôle d'affichage des données.

Vous pouvez également définir une expression XPath. Cette requête va être exécutée par le contrôle `XmlDataSource` pour récupérer le document XML.

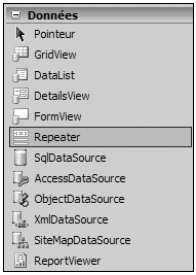
Ici, vous souhaitez uniquement afficher les différents posts du flux RSS et vous allez donc récupérer uniquement le contenu des différentes balises `item`. Pour ce faire, vous devez définir une requête XPath de la forme `rss/channel/item`.

Le code généré par le designer de Visual Basic 2005 Express Edition doit donc ressembler à cela :

```
       <asp:XmlDataSource ID="XmlDataSource1"
runat="server" DataFile=
"http://www.asp.net/modules/articleRss.aspx?count=5&mid=66"
            XPath="rss/channel/item"></asp:XmlDataSource>
```

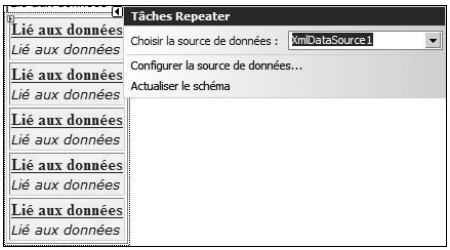
Vous devez à présent utiliser un contrôle d'affichage de données afin d'afficher les données fournies par ce contrôle `XmlDataSource`.

Vous pouvez par exemple utiliser un contrôle `RepeaterXE` Repeater. Ce contrôle permet d'afficher une liste d'éléments et de définir la manière de l'afficher.



◄ **Figure 13-4 :** Contrôle Repeater dans la boîte à outils

Placez-donc un contrôle `Repeater` sur votre formulaire et définissez la source de données utilisée par ce contrôle à l'aide du smart tag du contrôle.



◄ **Figure 13-5 :** Définition de la source de données du contrôle Repeater

Pour définir la manière d'afficher les données, vous devez vous rendre dans la source HTML de la page et personnaliser manuellement le contrôle `Repeater`.

Le `Repeater` offre la possibilité de définir des templates. Ces templates peuvent contenir du code HTML ainsi que du code ASP .NET "inline".

L'exemple suivant utilise trois types différents de templates :

- Un `HeaderTemplate`XE HeaderTemplate permet de définir ce qui va être affiché en en-tête de liste.
- Un `ItemTemplate`XE ItemTemplate permet de définir ce qui va être affiché pour chaque élément.
- Un `FooterTemplate`XE FooterTemplate permet de définir ce qui va être affiché en pied de liste.

Il est donc possible de personnaliser à loisir l'affichage des données, en créant, comme dans cet exemple, un tableau contenant les différents posts du flux RSS.

```
<asp:Repeater ID="Repeater1" runat="server"
 DataSourceID="XmlDataSource1">
<HeaderTemplate><table border="1">
</HeaderTemplate>
<ItemTemplate>
```

```
<tr><td style="background-color:#ffff99">
<strong>
<a href="<%#XPath("link")%>"><%#XPath("title")%></a>
</strong><br />
<font face="Verdana" size="2"><i><%#XPath("description") %></i>
</font>
</td></tr>
</ItemTemplate>
<FooterTemplate>
</table>
</FooterTemplate>
</asp:Repeater>
```

◄ **Figure 13-6 :** Rendu du contrôle Repeater en mode Design

Lorsque vous exécutez votre page web, elle doit afficher le flux RSS de la version française du site ASP .NET.

Les différentes stratégies de gestion de la session HTTP en ASP.NET
ASP.NET met à votre disposition tout un panel d'options vous permettant de gérer la session de manière la plus efficace possible en fonction de vos besoins. Il vous appartient ensuite de choisir celle qui correspondra le mieux à vos attentes en terme de performances mais aussi en terme de simplicité d'administration.

Formulaire et contrôle utilisateur, un pas vers la réutilisation
Lors d'un développement de projet web nous nous sommes tous retrouvé au moins une fois dans la situation où nous avions une page d'adjonction de données ainsi qu'une page de modification. Les problèmes commencent à survenir lors de la maintenance de l'application. Que va t-il se passer lorsque nous devrons éditer le formulaire d'adjonction ? Il y a de fortes chances pour que nous soyons également contraint de modifier celui de modification. N'est-ce pas un travail inefficace que d'avoir deux formulaires quasi identiques à maintenir séparément ? Ne vaudrait-il pas mieux disposer d'un seul et même formulaire chargé de réaliser ces deux opérations ? Nous allons essayer de répondre à ces questions en fondant notre solution sur la puissance du Framework .NET ainsi que sur les concepts objets.

Les contrôles serveur ASP.NET
ASP.NET révolutionne les formulaires de nos pages Web avec les WebForms pouvant être utilisés par le serveur pour générer des pages de manière dynamique. Vous allez devoir vous séparer des éléments input, pour ce que l'on appelle les contrôles serveur. Ceux-ci présentent une simplicité presque exemplaire d'utilisation donc pas de panique :)

Le ViewState en ASP.NET
Le ViewState est un système de maintien de la persistance des données ajouté dans le FrameWork .NET pour les pages ASP.NET. Ainsi dans chaque page ASP.NET où l'on a un formulaire, il existe un objet particulier que je vais essayer de présenter maintenant.

S'équiper pour ASP.NET
Afin de réaliser nos projets, ASP.NET nécessite l'installation de plusieurs outils sur votre poste comme le .NET Framework, un moteur de base de données et bien sur un environnement de développement. Voyons comment les installer. Microsoft mettant à notre disposition certains outils gratuitement, nous pouvons nous lancer dans l'ASP.NET sans aucun frais :)

▲ **Figure 13-7 :** Rendu de la page web

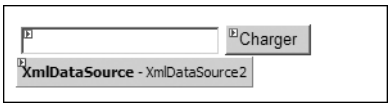
Charger un flux RSS manuellement

Le flux RSS est pour le moment défini de manière statique puisque vous avez indiqué en mode Design l'URL du flux RSS à charger. Il est bien évidemment préférable de laisser l'utilisateur sélectionner le flux qu'il souhaite consulter. En ce sens, vous avez le choix entre deux manières de procéder.

Pour débuter, vous allez proposer à l'utilisateur de saisir dans une zone de texte l'URL du flux RSS qu'il souhaite visualiser. Pour cela, ajoutez une `TextBox`XE TextBox nommé `txtUrl` ainsi qu'un bouton permettant de charger le flux saisi :

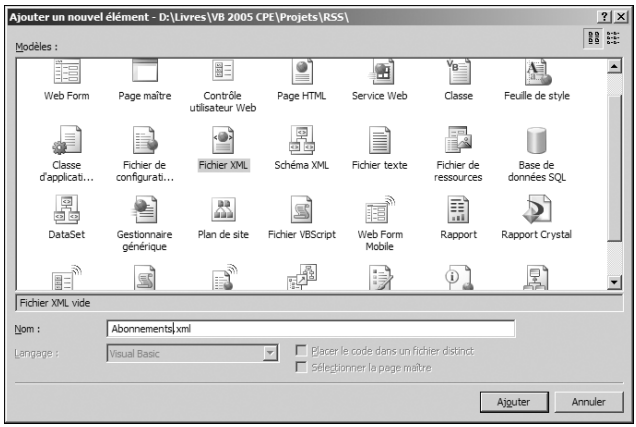
```
Protected Sub btnLoadUrl_Click(ByVal sender As Object, _
 ByVal e As System.EventArgs) Handles btnLoadUrl.Click
        XmlDataSource1.DataFile = txtUrl.Text
End Sub
```

Vous allez à présent permettre à l'utilisateur de stocker ses flux RSS favoris dans un fichier XML. Ce fichier XML sera ensuite chargé dans un contrôle `DropDownList`XE DropDownList grâce à un contrôle `XmlDataSource` de la même manière que précédemment.



◄ **Figure 13-8 :** Interface de chargement manuel

Le fichier XML est créé de la manière suivante. Libre à l'utilisateur d'ajouter autant de flux RSS qu'il souhaite. Pour créer un nouveau fichier XML, cliquez du bouton droit sur votre projet web et sélectionnez la commande **Ajouter un nouvel élément**. Dans la boîte de dialogue qui s'affiche, sélectionnez l'icône *Fichier XMLXE Fichier XML* et nommez votre fichier *Abonnements.xml*.



▲ **Figure 13-9 :** Création du fichier XML

```
<?xml version="1.0" encoding="utf-8" ?>
<abonnements>
  <flux url=http://blogs.developpeur.org/patrice/rss.aspx
 titre="Blog Patrice Lamarche Member of Wygteam">
  </flux>
  <flux url=http://blogs.developpeur.org/tonio/rss.aspx
 titre="Blog Antoine Griffard Member of Wygteam">
  </flux>
  <flux url=http://blog.developpez.com/xmlsrv/rss2.php?blog=26
 titre="Blog Jean Marc Rabillou">
  </flux>
<flux url=http://blog.madd0.com/feed/
 titre="Blog Mauricio Diaz Orlich" ></flux>
</abonnements>
```

Pour charger ce fichier XML dans le contrôle `DropDownList`, il suffit de procéder comme précédemment, en utilisant un contrôle `XmlDataSource`.

```
<asp:XmlDataSource ID="XmlDataSource2" runat="server"
➥ DataFile="~/Abonnements.xml" XPath="abonnements/flux">
</asp:XmlDataSource>
<asp:DropDownList ID="ddlAbonnements"
runat="server" AutoPostBack="True"
DataSourceID="XmlDataSource2" DataTextField="titre"
DataValueField="url"></asp:DropDownList><br />
```

Il ne reste plus qu'à charger le flux RSS sélectionné :

```
Protected Sub ddlAbonnements_SelectedIndexChanged(
ByVal sender As Object, ByVal e As System.EventArgs) _
Handles ddlAbonnements.SelectedIndexChanged
XmlDataSource1.DataFile = ddlAbonnements.SelectedValue
End Sub
```

Votre aggrégateur de flux RSS est terminé.

▲ **Figure 13-10** : Rendu final

13.2 Check-list

Dans ce chapitre, vous avez appris :

- ce qu'est le format RSS, à le lire manuellement et à comprendre son intérêt ;
- à intégrer des informations provenant d'autres sites web directement dans vos pages ;
- à utiliser des flux XML comme source de données grâce au contrôle `XmlDataSource` ;
- à utiliser les contrôles `Repeater` et `DropDownList`.

Création d'un gadget Live.com

Internet a pris récemment une nouvelle tournure avec l'avènement du Web 2.0. Le Web 2.0 est un terme marketing désignant une nouvelle vague de sites web. Ces derniers possèdent plus de dynamisme que les sites traditionnels grâce à la technologie Ajax, mise à la mode par Google, et s'adaptent mieux à l'utilisateur en lui permettant de personnaliser l'interface, d'incorporer les services qu'ils offrent dans d'autres sites web, etc.

Microsoft a bien entendu pris la vague du Web 2.0 en proposant une gamme de services Live disponibles à partir de n'importe quel type de périphérique, que l'on soit connecté à Internet ou non.

Un des premiers services Live lancés par Microsoft est le portail Live.com, qui permet à l'utilisateur de créer son propre portail en personnalisant tous les éléments affichés sur la page.

On peut en effet, ajouter, déplacer, supprimer tous les éléments affichés de manière intuitive grâce à la souris. Ainsi, ajouter un élément affichant la météo, les dernières sorties au cinéma, le contenu d'une boîte d'e-mails, ou encore les dernières informations d'un site récupérées via un flux RSS est un jeu d'enfant.

En plus de tous les éléments proposés par défaut dans Live.com, il est possible de développer ses propres éléments appelés "gadgets", qui, une fois publiés sur Internet, pourront être exploités par les utilisateurs de Live.com du monde entier.

Vous allez apprendre, dans ce chapitre, à créer un gadget simple, qui permet d'afficher le statut d'un contact Skype grâce à Visual Web Developer Express et Visual Basic 2005.

14.1 Configuration du système

Vous devez appliquer quelques changements à votre système pour développer des gadgets qui pourront être publiés sur le site Live.com.

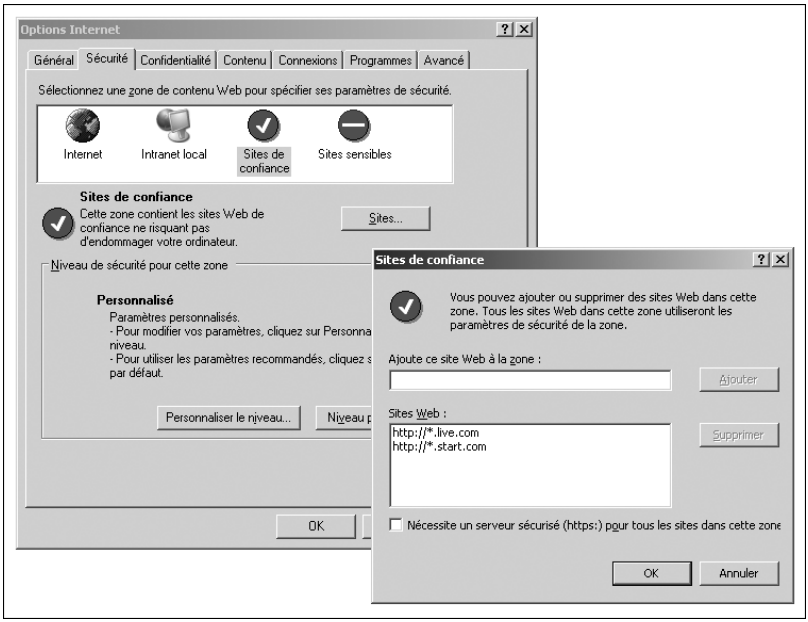
Vous devez en effet configurer Internet Explorer pour être capable de tester votre gadget développé localement sur votre machine, sur le portail Live.com. Il faut pour cela ajouter le site **http://*.live.com** et le site **http://*.start.com** dans les sites de confiance d'Internet Explorer.

Pour ce faire, lancez Internet Explorer, cliquez sur le menu **Outils** puis sur **Options Internet**.

Dans la boîte de dialogue qui s'affiche, cliquez sur l'onglet **Sécurité** et sélectionnez l'icône *Sites de confiance*. Cliquez ensuite sur le bouton **Sites**.

Dans la nouvelle fenêtre, vérifiez que la case *Nécessite un serveur sécurisé (https:) pour tous les sites dans cette zone* est bien décochée. Puis ajoutez les sites **http://*.live.com** et **http://*.start.com** dans les sites de confiance. L'étoile dans

l'URL indique que tous les sous-domaines de ces sites seront également considérés comme des sites de confiance.



▲ **Figure 14-1 :** Ajout dans les sites de confiance

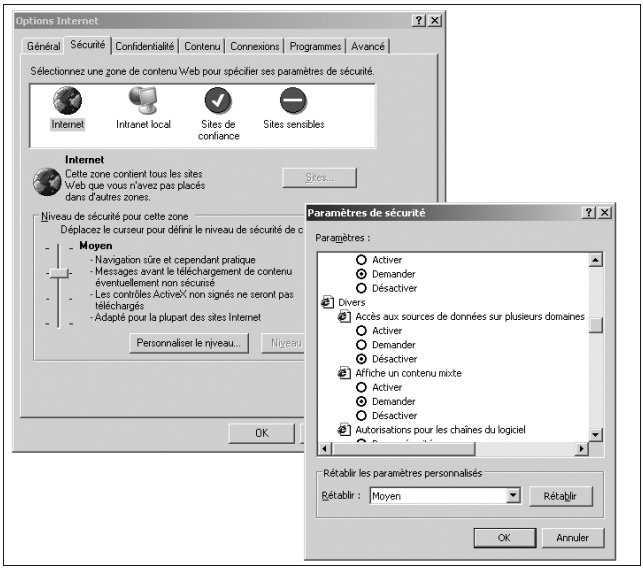
Ensuite, activez l'accès aux sources de données sur plusieurs domaines puisque vous souhaitez développer l'application localement et la tester sur le portail Live.com.

Pour cela, toujours sous l'onglet **Sécurité** de la fenêtre **Options Internet**, sélectionnez l'icône *Internet*, puis cliquez sur le bouton **Personnaliser le niveau**.

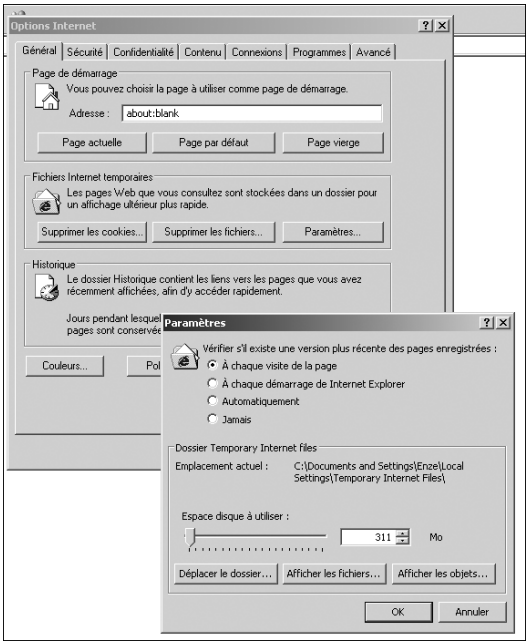
Dans la fenêtre qui s'affiche, recherchez la catégorie *Divers* et sélectionnez l'action *Demander* pour l'élément *Accès aux sources de données depuis plusieurs domaines* (voir Figure 14-2).

Enfin, désactivez le cache de votre navigateur pour être certain que le gadget qui s'affiche dans le site Live.com est bien celui que vous souhaitez tester, et non une ancienne version mise en cache.

Pour cela, sous l'onglet **Général** de la fenêtre **Options Internet**, cliquez sur le bouton **Paramètres** présent dans la zone intitulée *Fichiers Internet temporaires*. Dans la boîte de dialogue qui s'affiche, sélectionnez l'option *À chaque visite de la page*. Ainsi, Internet Explorer chargera la page web depuis le serveur qui la délivre à chaque visite (voir Figure 14-3).



▲ **Figure 14-2 :** Modification de la sécurité



▲ **Figure 14-3 :** Désactivation du cache

14.2 Composition d'un gadget

Un gadget se compose de trois éléments :

- Le premier et le plus important est le manifeste du gadget. Il s'agit d'un fichier XML qui permet de définir les informations sur le gadget : son titre, sa description et les liens vers ses autres composants.

Liste des balises les plus courantes dans un manifeste

Élément	Description
`title`	Titre du gadget. Vous pouvez définir n'importe quelle valeur de type texte. Ce titre sera affiché par Live.com en haut de votre gadget.
`Description`	Description du gadget.
`language`	Code de langue utilisé par le gadget.
`Binding:type`	Type utilisé par le gadget et qui doit être défini dans le fichier *.js* (code JavaScript) associé.
`item`	Un composant du gadget. Contient généralement un lien pointant vers un fichier JavaScript ou CSS.
`link`	Balise généralement présente dans une balise `item`. Permet de définir un lien vers un fichier JavaScript.
`link binding :type=css`	Balise incluse dans une balise `item`. Permet de définir un lien vers un fichier CSS.
`icons`	Balise contenant une ou plusieurs balises `icon` définissant les icônes à afficher, et en particulier l'icône principale du gadget.

Exemple de fichier manifeste :

```
<?xml version="1.0"?>
    <rss version="2.0" xmlns:binding="http://www.live.com">
        <channel>
<title>Mon premier Gadget</title>
<description>Description de mon premier gadget
</description>
<language>fr-fr</language>
<binding:type>Wygwam.monPremierGadget</binding:type>
<item>
<link>http://www.wygwam.com/Gadgets/
MonPremierGadget/MonPremierGadget.js</link>
</item>
<item>
<link binding:type="css"> http://www.wygwam.com/Gadgets/
MonPremierGadget/MonPremierGadget.css
</link>
```

```
</item>
<icons>
<icon height="32" width="32">
➥ http://www.mydomain.com/Gadgets/MonPremierGadget/MonPremierGadget.gif</icon>
</icons>
</channel>
</rss>
```

- Le deuxième élément important est le fichier JavaScript (**.js*). Il contient la logique du gadget et du code JavaScript, exécuté du côté client.
- Le dernier important est le fichier *.css*. Il permet de définir la mise en page et la mise en forme du contenu du gadget.

14.3 Créer un gadget de manière rapide et simple

Créer un gadget consiste donc à créer le fichier manifeste, un fichier JavaScript qui générera lui-même le rendu du gadget, et un style CSS pour mettre en forme et mettre en page le gadget.

Le problème est que toute l'interface doit être générée via du code JavaScript. Ainsi, si vous souhaitez ajouter des images ou des boutons, vous devrez générer le code HTML correspondant grâce à du code JavaScript.

Le sujet de cet ouvrage n'étant pas le langage JavaScript, et la génération d'un gadget réel se révélant vite complexe si l'on utilise uniquement JavaScript, vous allez mettre en œuvre une autre solution que celle promue par Microsoft, mais qui facilite grandement la tâche.

L'astuce consiste à utiliser une iframe, ce qui permettra d'afficher la page ASP .NET souhaitée à la place du code HTML généré à l'aide de JavaScript.

Le code JavaScript se résumera donc à quelques lignes dédiées à l'intégration de l'iframe dans le gadget.

Commencez par créer le manifeste du gadget :

```
<?xml version="1.0"?>
<rss version="2.0" xmlns:binding="http://www.live.com">
    <channel>
        <title>Skype Statut</title>
        <link>http://www.wygwam.com/</link>
        <description>Permet de visualiser le statut
        d'un utilisateur du logiciel Skype.</description>
 <language>en-us</language>
 <pubDate>Tue, 15 Aug 2006 12:30:00 GMT</pubDate>
```

```
        <binding:type>Gadget.Skype</binding:type>
        <item>
<link>http://www.wygwam.com/Gadgets/Patrice/Skype.js</link>
        </item>
        <item>
<link binding:type="css">
http://www.wygwam.com/Gadgets/Patrice/Skype.css</link>
        </item>
    </channel>
</rss>
```

Vous devez appeler l'URL du manifeste pour ajouter le gadget à votre page Live.com.

Pour cela, cliquez sur le bouton **Ajouter du contenu**. Dès lors un panneau s'affiche et propose les différentes catégories disponibles (*Gadgets*, *Actualités*, *Loisirs*, *Tech*…). Cliquez sur *Options avancées* pour accéder au panneau d'ajout avancé. Via ce panneau, vous pouvez rechercher des flux de données avec l'aide de Live Search, ajouter un gadget via une URL, ou bien importer un fichier OPML (fichier de listing de flux RSS). Vous allez à présent ajouter un gadget. L'URL est **www.wygwam.com/gadgets/patrice/skype.xml**. Elle redirige sur le manifeste du gadget. Celui-ci va permettre à la plateforme Live de savoir quels fichiers doivent être téléchargés pour l'affichage du gadget. Une fois ajouté, ce dernier vient se placer automatiquement sur la page en cours de lecture. Vous pourrez cependant retrouver la liste de vos gadgets et flux RSS dans la rubrique *Ma Sélection*.

14.4 Intégration d'une iframe

L'intégration de l'iframe se fait via le fichier JavaScript indiqué dans le manifeste.

```
////Création et référencement de l'espace de noms du gadget
registerNamespace("Gadget");

//Classe à implémenter

Gadget.Skype = function(p_elSource, p_args, p_namespace)
{
Gadget.Skype.initializeBase(this, arguments);
this.initialize = function(p_objScope)
     {
Gadget.Skype.getBaseMethod(this, "initialize",
 "Web.Bindings.Base").call(this, p_objScope);

var url = "http://localhost:4655/GadgetLive/Default.aspx";
```

```
        m_iframe = document.createElement("iframe");
        m_iframe.scrolling = "yes";
        m_iframe.frameBorder = "0";
        m_iframe.src = url;
        m_iframe.width="95%";
        m_iframe.height="285px";
        p_elSource.appendChild(m_iframe);
    }

//Référencement de la classe
Gadget.Skype.registerClass("Gadget.Skype",
 "Web.Bindings.Base");
}
```

Remarque

URL de la page incluse

Vous devez changer l'URL contenue dans la variable `url` par celle de votre page. Une fois l'application déployée, vous devrez changer cette URL pour afficher la page hébergée sur Internet, et non votre page locale.

14.5 Création de la page ASP .NET

Maintenant que vous êtes capable d'afficher votre page au sein du gadget grâce à l'utilisation d'une iframe, vous devez développer cette page pour afficher le statut des contacts Skype et gérer lesdits contacts.

Gestion des contacts

L'utilisateur du gadget doit être capable de gérer ses contacts. Vous allez lui permettre d'ajouter un contact ou de supprimer directement toute sa liste de contacts.

Pour cela, vous allez utiliser la gestion des profils proposée par ASP .NET 2. La nouvelle version d'ASP .NET permet de gérer facilement des profils utilisateurs stockés dans une base SQL Server. Ces profils étant également applicables aux utilisateurs anonymes non authentifiés sur le serveur, vous allez les sauvegarder pour tous les visiteurs sans leur demander de s'authentifier.

Pour cela, lancez l'utilitaire *aspnet_regsql.exe* présent dans le dossier du Framework .NET 2.

Remarque

Dossier du Framework .NET 2

Le Framework .NET se situe dans le dossier *Windows/Microsoft.NET/Framework*. Vous y trouverez un dossier par version du Framework .NET installé sur votre machine. Rendez-vous dans le dossier de la version 2. Exemple : *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

Indiquez les différents renseignements demandés par l'Assistant et validez.

Cet Assistant permet de générer une base de données SQL Server, qui peut contenir toutes les informations nécessaires à la gestion des membres, avec entre autres, tout ce qui concerne la gestion des profils.

Pour activer les profils, vous devez modifier le fichier *Web.config* de votre application web. Ajoutez la section suivante à votre *Web.config* pour configurer correctement votre application :

```
<anonymousIdentification enabled="true"/>
    <trust level="Medium"/>
    <profile defaultProvider="MyProfileProvider">
      <properties>
        <add name="Contacts" allowAnonymous="true"
type="ContactsManager" />
      </properties>
      <providers>
        <add name="MyProfileProvider"
             connectionStringName="SqlServer"
                applicationName="/"
type="System.Web.Profile.SqlProfileProvider" />
      </providers>
    </profile>
```

En ajoutant cette section, vous spécifiez que vous activez la gestion des profils pour les utilisateurs anonymes, qu'un profil est constitué d'une propriété `Contact` qui est de type `ContactsManager`. Vous spécifiez également que vous souhaitez utiliser le fournisseur de profils `SqlProfileProvider` (livré avec ASP .NET 2). Ce fournisseur utilise une chaîne de connexion nommée `SqlServer`.

Vous devez donc la définir dans la section `ConnectionStrings` du *Web.config* :

```
<connectionStrings>
<add
connectionString="server=.\SQLEXPRESS;database=aspnetdb;
integrated security=SSPI;" name="SqlServer"/>
</connectionStrings>
```

La base par défaut générée par l'Assistant est nommée `aspnetdb`. Utilisez donc ce nom dans la chaîne de connexion.

Il ne reste plus qu'à créer la classe `ContactsManager` pour gérer les contacts :

```
Imports Microsoft.VisualBasic
Imports System.Collections.Generic

Public Class ContactsManager
    Private m_contacts As List(Of String)

    Public Sub New()

    End Sub

    Public Property Contacts() As List(Of String)
        Get
            If m_contacts Is Nothing Then
                m_contacts = New List(Of String)
            End If
            Return m_contacts
        End Get
        Set(ByVal value As List(Of String))
            m_contacts = value
        End Set
    End Property

    Public Sub Ajouter(ByVal contact As String)
        If m_contacts Is Nothing Then
            m_contacts = New List(Of String)

        End If
        m_contacts.Add(contact)
    End Sub

End Class
```

Cette classe permet d'accéder à la liste des contacts grâce à la propriété du même nom et d'ajouter des contacts grâce à la méthode `Ajouter`.

14.6 Interface de gestion des contacts

Il faut proposer à l'utilisateur d'ajouter des contacts ou de réinitialiser sa liste grâce à quelques contrôles web. Pour cela, ajoutez un contrôle `TextBox` que vous nommerez `txtContact` en modifiant sa propriété `Name`, et deux boutons, le premier pour ajouter le contact saisi, et le second pour réinitialiser la liste.

◄ **Figure 14-4 :** Interface de gestion des contacts

Dans l'événement `Click` du bouton d'ajout, placez le texte saisi dans la collection de contacts :

```
  Protected Sub btnAjout_Click(ByVal sender As Object,
  ByVal e As System.EventArgs) Handles btnAjout.Click
        Profile.Contacts.Ajouter(txtContact.Text)
    End Sub
```

Puis dans l'événement `Click` du bouton de réinitialisation, modifiez la propriété `Contacts` de l'objet `Profile` en lui attribuant une nouvelle instance de `ContactsManager` :

```
Protected Sub BtnInitialiser_Click(ByVal sender As Object,
 ByVal e As System.EventArgs) Handles Button1.Click
        Profile.Contacts = New ContactsManager
    End Sub
```

14.7 Affichage des contacts

Pour proposer une interface facile d'emploi à l'utilisateur, placez les contrôles de gestion des utilisateurs dans une `div` nommée `ajout`. Cette `div` sera affichée ou masquée par l'utilisateur grâce à un clic sur un lien.

Pour afficher tous les contacts présents dans le profil de l'utilisateur, on effectue une boucle `for… each`, qui permet d'énumérer tous les contacts et d'afficher l'image de statut Skype correspondante.

Remarque

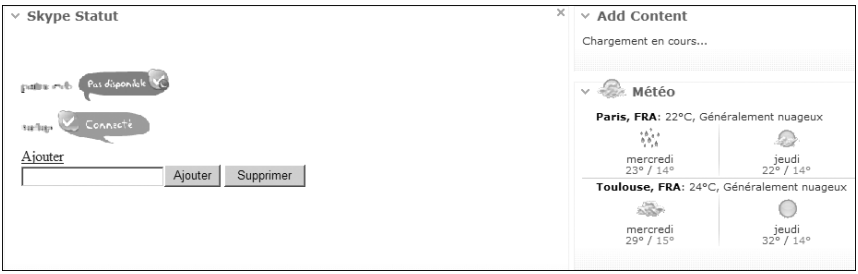
Récupérer le statut d'un contact Skype

Skype permet de connaître l'état d'un utilisateur connecté à son service grâce à une image disponible à l'adresse **http://mystatus.skype.com/balloon/pseudo**, où **pseudo** est le pseudo Skype de l'utilisateur.

Cela donne :

```
<body>
<script>
function masquerAjout()
{
if (document.getElementById('ajout').style.display=='block')
document.getElementById('ajout').style.display='none';
```

```
else
document.getElementById('ajout').style.display='block';
}
</script>
<form id="form1" runat="server">
    <div>
        <br />
<%For Each s As String In Profile.Contacts.Contacts%>
<%=s %><a href="skype:<%=s %>?call">
<img align="absmiddle" width="112,5" height="45" border="0"
src="http://mystatus.skype.com/balloon/<%=s %>"/></a><br />
        <%Next %>
<a href="#" onclick="javascript:masquerAjout()">Ajouter</a>
        <div id="ajout" style="display:block">
            <asp:TextBox ID="txtContact"
            ➥ runat="server"></asp:TextBox><asp:Button
ID="btnAjout" runat="server" Text="Ajouter" />
            <asp:Button ID="Button1" runat="server"
 Text="Supprimer" />
        </div>
    </div>
    </form>
</body>
```



▲ **Figure 14-5 :** Interface du gadget dans Live.com

14.8 Check-list

Dans ce chapitre, vous avez appris à :

- créer un gadget Live.com grâce à l'utilisation d'une iframe ;
- attribuer des profils à des utilisateurs anonymes.

Sélecteur de papier peint

Dans ce chapitre, vous allez créer un utilitaire de changement de papier peint (wallpaper), l'image d'arrière-plan du Bureau de Windows. Le système d'exploitation de Microsoft permet de sélectionner une image et non d'afficher de nouvelles images aléatoirement. Vous allez donc combler ce manque grâce à un petit utilitaire développé en Visual Basic 2005.

Cette application sera le prétexte pour développer une application Windows qui s'intègre totalement au système d'exploitation en utilisant des mécanismes courants tels que le démarrage automatique lors du lancement de Windows, l'utilisation d'icône dans la barre système de la barre des tâches (trayIcon), l'accès à la Base de registre, la gestion par glisser-déposer, etc.

Vous verrez également comment utiliser les fonctions système API Win32 offertes par Windows.

15.1 Classes et espaces de noms utilisés

Bien que l'application que vous allez réaliser soit assez simple, vous allez devoir utiliser des espaces de noms assez variés puisque l'utilitaire nécessite des fonctionnalités dans de multiples domaines :

- `System.Windows.Forms`, pour tout ce qui concerne la création d'interface Windows ;
- `System.IO`, pour la manipulation de fichiers ;
- `System.Drawing`, pour la manipulation d'images ;
- `Microsoft.Win32`, pour l'interopérabilité avec le système d'exploitation et plus particulièrement l'accès à la Base de registre ;
- `System.XML`, pour la manipulation de documents XML.

15.2 Accès aux données

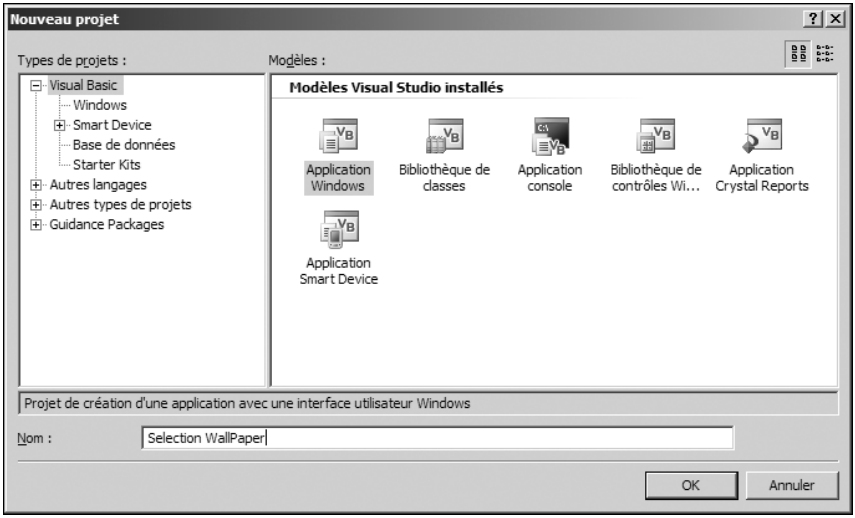
L'objectif de l'application que vous allez développer est de changer de papier peint automatiquement à chaque démarrage de l'application, ou manuellement suite à une action de l'utilisateur.

Vous devez donc créer et stocker une liste d'images qui sera utilisée par l'utilitaire.

Stocker une liste de chemins de fichiers ne nécessite pas la création d'une base de données car elle ne contiendrait qu'une table et qu'un seul champ. Un fichier XML convient mieux au stockage d'une simple liste d'informations.

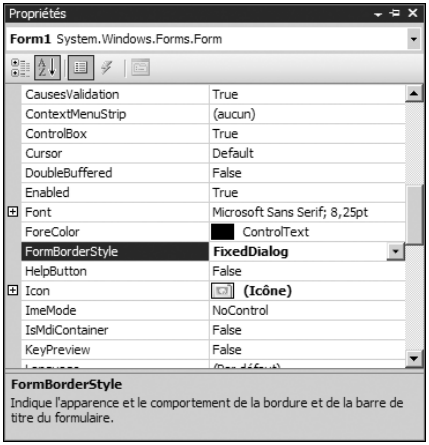
15.3 Interface utilisateur

L'application ne compte qu'un seul formulaire. Vous allez donc modifier celui généré par défaut lors de la création du projet par Visual Basic 2005 Express.



▲ **Figure 15-1 :** Création du projet Application Windows

L'interface graphique du formulaire est assez simple. Pour éviter de gérer le redimensionnement des fenêtres (inutile dans ce cas), vous allez modifier deux propriétés du formulaire. Pour commencer, modifiez la propriété `FormBorderStyle`. Elle permet de définir le type de bordure du formulaire. Vous pouvez donc indiquer que votre formulaire ne possède pas de bordure, ou possède une bordure redimensionnable, etc. Dans ce cas, vous allez affecter la valeur `FixedDialog` pour indiquer que la fenêtre n'est pas redimensionnable car il s'agit d'une boîte de dialogue simple. Modifier cette propriété ne suffit cependant pas à empêcher le redimensionnement de la fenêtre par un utilisateur. Le bouton d'agrandissement de la fenêtre est toujours présent et accessible. L'utilisateur peut donc agrandir la fenêtre grâce à ce bouton présent dans la barre de titre de la plupart des applications. Vous pouvez bien évidemment désactiver ce bouton grâce à la propriété `MaximizeBox` en lui affectant la valeur `False`.



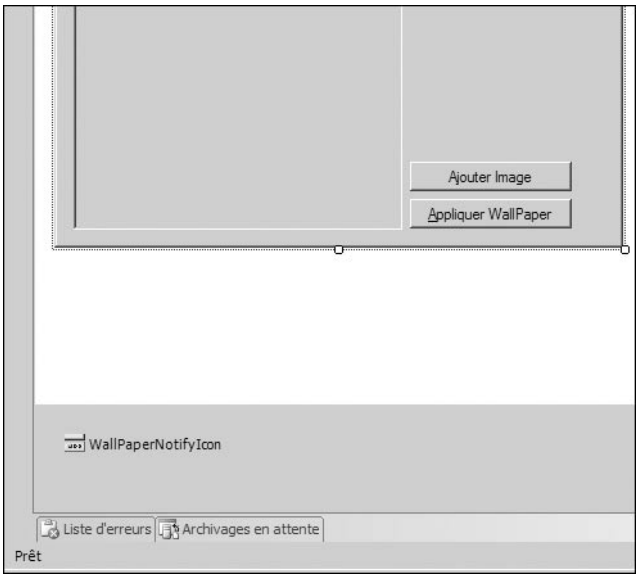
◄ **Figure 15-2 :** Modification des propriétés du formulaire

Vous allez utiliser un contrôle `ListBox`, qui va permettre d'afficher la liste des images qui seront affichées aléatoirement. Placez ce contrôle dans la partie supérieure de votre formulaire de telle sorte qu'il prenne toute la largeur de la fenêtre. Placez également un contrôle `PictureBox` pour afficher la miniature de l'image sélectionnée dans la zone de liste que vous venez de créer. Pour permettre à l'utilisateur de délimiter la `PictureBox`, modifiez sa propriété `BorderStyle` et définissez sa valeur à `Fixed3D` pour afficher une bordure avec un effet 3D délimitant l'emplacement et la taille de la miniature. Placez deux boutons, le premier permettant d'ajouter une image à la liste, et le second permettant de changer manuellement le fond d'écran du Bureau de Windows.

◄ **Figure 15-3 :** Interface de l'application

Afficher une icône dans la zone de notification

Pour rendre accessible l'application et pour qu'elle n'occupe pas de place dans la barre des tâches, vous allez créer une icône dans la zone de notification, sur le côté droit de la barre des tâches, près de l'heure du système, de l'icône permettant de modifier le volume ou de l'icône MSN Messenger si vous l'avez installé. Pour ce faire, il suffit d'utiliser le composant `NotifyIcon` en double-cliquant dessus dans la boîte à outils. Le contrôle n'est pas ajouté sur le formulaire, mais en dessous, dans la zone réservée aux composants non visuels. Changez la propriété `Text` du contrôle pour afficher un message dans l'info-bulle de l'icône. Vous pouvez par exemple définir cette valeur à `"Sélection WallPaper"`.



▲ **Figure 15-4 :** Contrôle non visuel NotifyIcon

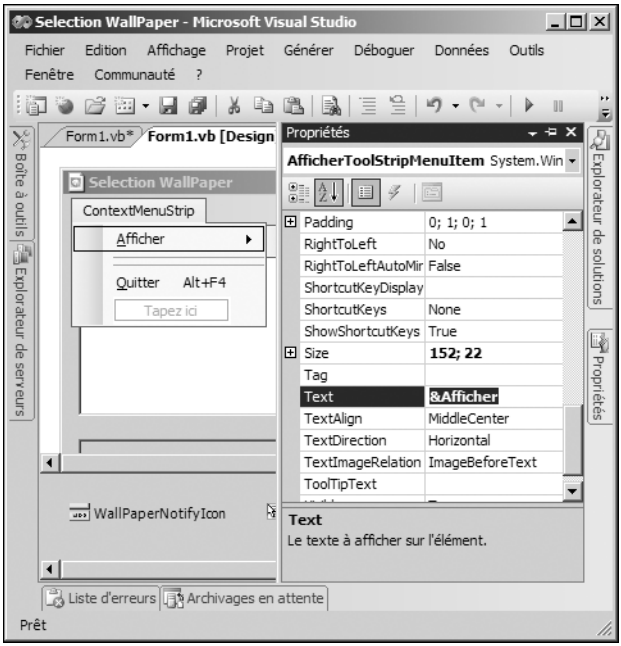
Maintenant que l'icône est créée grâce au contrôle `NotifyIcon`, vous allez permettre à l'utilisateur d'accéder rapidement aux fonctions de l'application grâce à un menu contextuel. Ce menu permettra d'afficher le formulaire principal et de quitter l'application. Pour le créer, il suffit d'utiliser le contrôle `ContextMenuStrip`, disponible dans la catégorie *Menus et barres d'outils*. Double-cliquez sur le contrôle pour l'attacher au formulaire et ajoutez une entrée `&Afficher`, une entrée ayant comme texte un simple tiret (`-`) et une dernière entrée ayant pour texte `&Quitter`.

Remarque

Astuces pour les menus

L'esperluette (&) permet d'indiquer la lettre qui pourra servir à l'utilisateur pour accéder à l'élément du menu via un raccourci clavier Alt+la lettre suivant l'esperluette.

L'utilisation d'un simple tire (-) permet de créer un séparateur dans le menu.



▲ **Figure 15-5 :** Utilisation du caractère & dans les menus

Pour lier l'icône dans la zone de notification et le menu contextuel, il faut modifier la propriété `ContextMenuStrip` du contrôle `NotifyIcon` en sélectionnant le menu contextuel que vous venez de créer.

Dans l'événement `Click` de l'élément `&Afficher` du menu contextuel, placez le code suivant pour afficher le formulaire :

```
Private Sub AfficherToolStripMenuItem_Click(ByVal sender As System.Object,
➥ ByVal e As System.EventArgs) Handles AfficherToolStripMenuItem.Click
        Me.WindowState = FormWindowState.Normal
    End Sub
```

Pour gérer la présence de l'icône dans la barre des tâches, placez le code suivant dans l'événement SizeChanged du formulaire :

```
Private Sub Form1_SizeChanged(ByVal sender As System.Object, ByVal e As
➥ System.EventArgs) Handles MyBase.SizeChanged
        If Me.WindowState = FormWindowState.Minimized Then
            Me.ShowInTaskbar = False
        Else
            Me.ShowInTaskbar = True
        End If

    End Sub
```

Ce code permet de masquer l'icône de la barre des tâches lorsque la fenêtre est réduite, et de l'afficher lorsque la fenêtre est visible grâce à la propriété ShowInTaskBar.

Pour gérer la fermeture de l'application, ajoutez le code suivant dans l'événement Click de l'élément &Quitter :

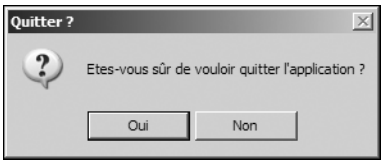
```
  Private Sub QuitterToolStripMenuItem_Click(ByVal sender As System.Object,
  ➥ ByVal e As System.EventArgs) Handles QuitterToolStripMenuItem.Click
        Application.Exit()
    End Sub
```

La méthode Exit de la classe Application permet de quitter l'application.

Pour demander une confirmation à l'utilisateur, il faut saisir quelques lignes de code dans l'événement FormClosing du formulaire :

```
Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As
➥ System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        If MsgBox("Êtes-vous sûr de vouloir quitter l'application ?",
        ➥ MsgBoxStyle.Question + MsgBoxStyle.YesNo, "Quitter ?") <>
        ➥ MsgBoxResult.Yes Then
            e.Cancel = True
        End If
    End Sub
```

Dans ce code, vous affichez une boîte de dialogue à l'aide de la fonction MsgBox et vous annulez la fermeture de la fenêtre en modifiant le paramètre FormClosingEventArgs, nommé e, de l'événement FormClosing.



◀ **Figure 15-6 :** Boîte de dialogue de confirmation

15.4 Réalisation

Passons à présent au "gros œuvre", c'est-à-dire à l'essentiel de l'application. Vous allez apprendre à modifier le papier peint courant, à afficher une miniature d'image, et à interagir avec Windows en implémentant des fonctionnalités de glisser-déposer et de manipulation de Base de registre.

Définition du papier peint courant

L'objectif principal de l'utilitaire que vous développez est de changer le papier peint courant grâce à la liste d'images que l'utilisateur a définie.

Il n'existe malheureusement pas de fonction dans le Framework .NET qui permette d'effectuer ce changement de papier peint. Vous allez donc devoir utiliser les fonctions propres au système d'exploitation : l'API (Application Programming Interface) de Windows. L'interface de programmation d'applications proposée par Windows permet d'accéder à une importante partie des fonctionnalités utilisées par le système d'exploitation lui-même. Ainsi, la gestion des fenêtres, la gestion du Presse-papiers, le formatage de disques peuvent être effectués par l'appel des fonctions utilisées par Windows et proposées par l'API, nommées couramment "fonctions API".

La fonction que vous devez utiliser est `SystemParametersInfo`, présente dans la bibliothèque *user32.dll*, qui est l'une des librairies les plus importantes de Windows. Comme souvent, l'appel à une fonction API requiert l'utilisation de certaines constantes. Vous allez en utiliser trois différentes. Les valeurs de ces constantes ainsi que la syntaxe de la déclaration de la fonction API sont indiquées dans la documentation de Windows destinée aux développeurs et fournie par Microsoft. Malheureusement, cette documentation est assez ancienne ; elle date de la création de la version de Windows que vous utilisez, et les exemples fournis sont rarement proposés en Visual Basic .NET ou en C#. Il est donc préférable de rechercher de la documentation sur un site dédié disponible sur **www.pinvoke.net**.

Voici la déclaration des constantes et de la fonction API à utiliser pour le changement du papier peint :

```
Private Const SPI_SETDESKWALLPAPER As Integer = &H14
    Private Const SPIF_UPDATEINIFILE As Integer = &H1
    Private Const SPIF_SENDWININICHANGE As Integer = &H2

    Private Declare Auto Function SystemParametersInfo Lib "user32.dll" ( _
        ByVal uAction As Integer, ByVal uParam As Integer, _
        ByVal lpvParam As String, ByVal fuWinIni As Integer) As Integer
```

Vous allez maintenant créer une méthode `SetCurrentWallPaper` qui va permettre de définir le papier peint courant, à savoir le chemin de l'image qui sera

affichée, et le mode d'affichage (centré ou étiré). Le changement d'image est effectué grâce à un appel à la fonction API que vous avez déclarée précédemment et le mode d'affichage est défini grâce à des modifications dans la Base de registre de Windows.

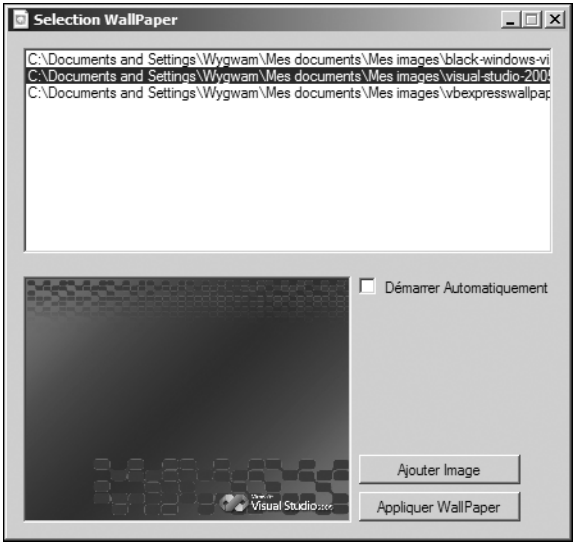
```
Public Shared Sub SetCurrentWallPaper(ByVal imagePath As String)
        Dim imgBitmap As Image = Image.FromFile(imagePath)
        Dim desktopKey As RegistryKey =
        ➥ My.Computer.Registry.CurrentUser.OpenSubKey("Control
        ➥ Panel\Desktop", True)

        desktopKey.SetValue("TileWallpaper", "0")
        If imgBitmap.Width > My.Computer.Screen.Bounds.Width Or
        ➥ imgBitmap.Height > My.Computer.Screen.Bounds.Height Then
            desktopKey.SetValue("WallpaperStyle", "2")
        Else
            desktopKey.SetValue("WallpaperStyle", "0")
        End If
        desktopKey.Flush()
        desktopKey.Close()

        If System.IO.Path.GetExtension(imagePath) <> "bmp" Then
            imagePath = Path.ChangeExtension(imagePath, "bmp")
            imgBitmap.Save(imagePath, Imaging.ImageFormat.Bmp)
        End If

imgBitmap.Dispose()
        SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, imagePath, _
                    SPIF_UPDATEINIFILE Or SPIF_SENDWININICHANGE)
    End Sub
```

Affichage de la miniature

Pour améliorer l'ergonomie de l'utilitaire, un aperçu de l'image sélectionnée dans le contrôle `ListBox` présent sur le formulaire va être affiché. Pour cela, il faut utiliser l'événement `SelectedIndexChanged` et vérifier qu'un élément est sélectionné en testant si la propriété `SelectedIndex` du contrôle est supérieure à –1.

Vous ne pouvez malheureusement pas afficher l'image dans un contrôle en spécifiant directement le nom et le chemin du fichier souhaité car la propriété `Image` d'une `PictureBox` attend un objet de type `Image`. Pour des raisons de simplicité, vous pouvez utiliser un objet de type `Bitmap` (le type `Bitmap` hérite du type `Image` et pourra donc être utilisé pour définir la propriété `Image` d'une `PictureBox`) pour charger un fichier image et ensuite l'afficher. Pour bénéficier d'un meilleur affichage de l'image, écrivez le code suivant, qui centre l'image si sa taille est inférieure à la taille de la `PictureBox` et la redimensionne si elle est plus importante.

```
Private Sub lstWallPapers_SelectedIndexChanged(ByVal sender As
➥ System.Object, ByVal e As System.EventArgs) Handles
➥ lstWallPapers.SelectedIndexChanged
      If lstWallPapers.SelectedIndex > -1 Then
          Dim apercu As Bitmap = Image.FromFile(lstWallPapers.Text)
          If apercu.Width > 192 Or apercu.Height > 256 Then
              pctApercu.SizeMode = PictureBoxSizeMode.StretchImage
          Else
              pctApercu.SizeMode = PictureBoxSizeMode.CenterImage
          End If
          pctApercu.Image = apercu
      End If
  End Sub
```



◄ **Figure 15-7 :** Affichage de l'aperçu

Gestion du glisser-lâcher

Une des fonctionnalités intéressantes qui est de plus en plus implémentée dans les applications Windows et qui offre une souplesse d'utilisation à l'utilisateur est de permettre le "glisser-lâcher" de fichiers depuis l'Explorateur de fichiers de Windows vers la zone de liste de l'application.

Pour activer le glisser-lâcher vers la zone de liste, vous devez commencer par modifier la propriété `AllowDrop` et la définir à `True`.

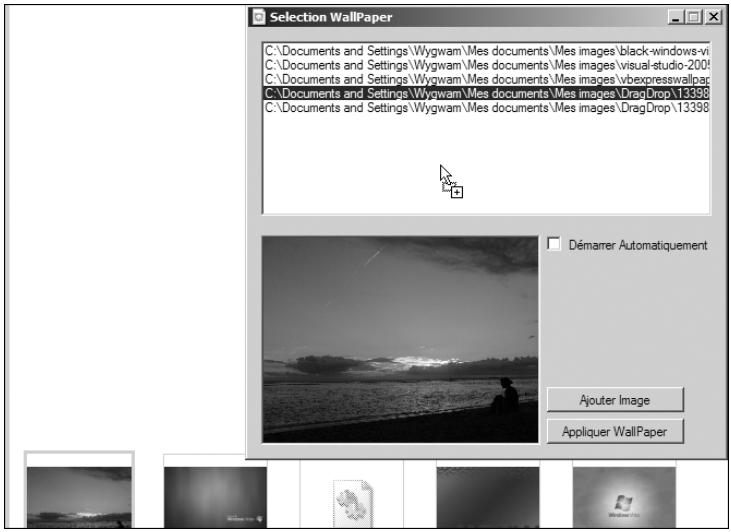
Maintenant que le contrôle est capable de recevoir des éléments par glisser-lâcher, il est nécessaire de changer le curseur de la souris lorsqu'un élément glisse au-dessus du contrôle pour indiquer à l'utilisateur qu'il peut déposer

l'élément sur ledit contrôle. Il faut en ce sens utiliser l'événement `DragOver` de la `ListBox` et modifier le paramètre `e` de type `DragEventArgs` pour spécifier le curseur affiché.

```
Private Sub lstWallPapers_DragOver(ByVal sender As System.Object, ByVal e As
➥ System.Windows.Forms.DragEventArgs) Handles lstWallPapers.DragOver
        If (e.Data.GetDataPresent(DataFormats.FileDrop)) Then
            e.Effect = DragDropEffects.Copy
        End If
    End Sub
```

Lorsqu'un élément est déposé, l'événement `DragDrop` est déclenché. Il faut donc ajouter le code nécessaire à l'ajout des éléments dans la `ListBox` dans cet événement.

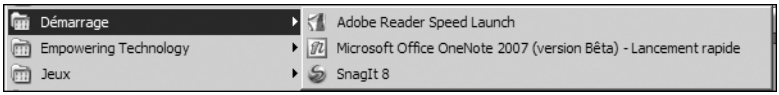
```
Private Sub lstWallPapers_DragDrop(ByVal sender As System.Object, ByVal e As
➥ System.Windows.Forms.DragEventArgs) Handles lstWallPapers.DragDrop
        If (e.Data.GetDataPresent(DataFormats.FileDrop)) Then
            Dim files() As String =
            ➥ CType(e.Data.GetData(DataFormats.FileDrop), String())
            For Each file As String In files
                lstWallPapers.Items.Add(file)
            Next
        End If

    End Sub
```



▲ **Figure 15-8 :** Gestion du glisser-déposer

Démarrage automatique

Le démarrage automatique d'une application lors du lancement de Windows peut se faire de deux manières. La première est de créer un raccourci dans le groupe de démarrage du menu **Démarrer**. La seconde est la création d'une valeur dans la Base de registre de Windows.



▲ **Figure 15-9 :** Groupe de démarrage dans le menu Démarrer

Pour gérer toute cette partie liée au démarrage automatique, il est préférable de créer une classe `StartManager`, responsable de la création et de la suppression de cette valeur dans la Base de registre. Il suffit ensuite de créer trois méthodes, `EnableAutoStart`, `DisableAutoStart` et `IsAutoStart`, pour respectivement activer le démarrage automatique, le désactiver, et tester si l'application a démarré automatiquement.

```
Imports Microsoft.Win32

Public Class StartManager
    Public Shared Sub EnableAutoStart()
        If Not IsAutoStart() Then
            Dim runKey As RegistryKey
            runKey =
            My.Computer.Registry.CurrentUser.OpenSubKey("Software\Microsoft\
            ➥ Windows\CurrentVersion\Run", True)
            runKey.SetValue("SelecteurWallPaper", Application.ExecutablePath)
            runKey.Flush()
            runKey.Close()
        End If
    End Sub

    Public Shared Sub DisableAutoStart()
        Dim runKey As RegistryKey
        runKey =
        ➥ My.Computer.Registry.CurrentUser.OpenSubKey("Software\Microsoft\
        ➥ Windows\CurrentVersion\Run", True)
        runKey.DeleteValue("SelecteurWallPaper")
        runKey.Flush()
        runKey.Close()

    End Sub

    Public Shared Function IsAutoStart() As Boolean
        Dim runKey As RegistryKey
        Dim returnValue As Boolean
```

```
        runKey =
        ➥ My.Computer.Registry.CurrentUser.OpenSubKey("Software\Microsoft\
        ➥ Windows\CurrentVersion\Run", False)
        returnValue = (Not runKey.GetValue("SelecteurWallPaper") Is Nothing)
        runKey.Close()
        Return returnValue
    End Function
End Class
```

Il suffit à présent d'appeler ces méthodes lors de l'événement `CheckedChange` du contrôle `CheckBox` pour gérer le démarrage automatique de l'application.

Chargement et sauvegarde de la liste

Il est possible de stocker des informations dans des fichiers XML grâce au mécanisme de sérialisation

Reportez-vous au chapitre ***Gestion d'un concours*** *pour en savoir plus à ce sujet.*

Vous allez ici utiliser une autre technique, qui consiste à créer manuellement et ex nihilo un document XML grâce aux objets proposés par le Framework .NET dans l'espace de noms `System.XML`. Vous allez donc utiliser un objet `XmlTextWriter`, qui va vous permettre d'écrire un fichier XML en définissant ses balises une par une, ainsi qu'un objet `XmlDocument`, qui vous permettra de charger un fichier XML et de naviguer au sein de celui-ci.

```
Private Sub SaveList()
        Dim ListWriter As New XmlTextWriter("images.xml",
        ➥ System.Text.Encoding.UTF8)
        ListWriter.WriteStartDocument()
        ListWriter.WriteStartElement("Images")
        For Each item As String In lstWallPapers.Items
            ListWriter.WriteElementString("Image", item)
        Next
        ListWriter.WriteEndElement()
        ListWriter.WriteEndDocument()
        ListWriter.Flush()
        ListWriter.Close()
    End Sub

    Private Sub LoadList()
        Dim listDocument As New XmlDocument
        Try
            listDocument.Load("images.xml")
            For Each image As XmlNode In
            ➥ listDocument.GetElementsByTagName("Image")
                lstWallPapers.Items.Add(image.InnerText)
```

```
            Next
        Catch filex As System.IO.FileNotFoundException
            'on ne fait rien, erreur générée si le fichier est absent (c'est
            ➥ le cas lors du premier démarrage
        Catch ex As Exception
            MsgBox("Une erreur s'est produite : " + ex.Message,
            ➥ MsgBoxStyle.Exclamation)
        End Try
```

Il faut à présent appeler ces méthodes depuis le formulaire. Vous devez donc charger le fichier XML lors du démarrage du formulaire grâce à son événement `Load`.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
➥ System.EventArgs) Handles MyBase.Load
        LoadList()
    End Sub
```

Ensuite sauvegardez la liste courante des images vers le fichier lors de la fermeture de l'application et donc du formulaire en modifiant son événement `FormClosing`.

```
Private Sub Form1_FormClosing(ByVal sender As Object, ByVal e As
➥ System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        If MsgBox("Êtes-vous sûr de vouloir quitter l'application ?",
        ➥ MsgBoxStyle.Question + MsgBoxStyle.YesNo, "Quitter ?") <>
        ➥ MsgBoxResult.Yes Then
            e.Cancel = True
        Else
            SaveList()
        End If

    End Sub
```

15.5 Check-list

Dans ce chapitre, vous avez appris à :

- utiliser le contrôle `NotifyIcon` pour afficher une icône dans la barre système de la barre des tâches ;
- lire et écrire un fichier XML grâce à un objet `XmlDocument` ;
- utiliser les fonctions API Win32 ;
- implémenter les opérations de glisser-lâcher entre l'Explorateur et votre application ;
- manipuler des images.

WebParts

Laisser à l'utilisateur la possibilité de personnaliser lui-même une application selon ses propres besoins est une fonctionnalité intéressante. Elle serait néanmoins compliquée à mettre en place si vous deviez partir de zéro pour réaliser vos propres mécanismes de stockage des éléments personnalisés par les utilisateurs et leur fournir des moyens de les modifier et de les organiser selon leur convenance. Heureusement, le Framework ASP .NET 2.0 met désormais à disposition ce que l'on appelle les WebParts.

Il s'agit d'un ensemble de composants qui interagissent et permettent de créer des pages web dans lesquelles les utilisateurs peuvent modifier l'apparence, la disposition, le comportement et les propriétés de chaque élément directement à partir du navigateur.

La fonctionnalité de base des WebParts est la personnalisation. Elle permet aux utilisateurs de modifier ou de personnaliser la disposition, l'apparence et le comportement des contrôles `WebPart` sur une page. Ces paramètres personnalisés sont rendus persistants non seulement durant la session de navigation en cours, mais aussi à long terme, afin que les paramètres d'un utilisateur ne soient pas perdus entre deux consultations du site.

16.1 Classes et espaces de noms utilisés

Pas moins de cent cinquante classes sont associées aux WebParts, soit presque autant que dans `System.Web.UI.WebControls`, qui contient les contrôles serveurs web classiques. C'est pourquoi elles nécessitent un nouvel espace de noms : `System.Web.UI.WebControls.WebParts`.

Toutes ces classes ne sont en réalité qu'un portage de ce qui constitue la base du portail de travail collaboratif Windows Sharepoint Services, le premier Framework qui a utilisé cette notion de WebPart et mis à disposition un ensemble de WebParts déjà réalisés qui permettent de gérer des documents, de créer des listes d'événements, d'annonces ou des tâches.

16.2 Configuration

Les WebParts sont intégrés au modèle de fournisseur ASP .NET dont nous avons parlé dans plusieurs chapitres de ce livre. Cela implique notamment que vous pouvez configurer certaines fonctionnalités à partir du fichier *web.config* et de la section `WebParts` appropriée.

Section WebParts

Voici à quoi ressemblent les sections de configuration qui permettent de personnaliser le fournisseur de personnalisation des WebParts :

```
<webParts enableExport="true|false">
   <personalization>…</personalization>
   <transformers>…</transformers>
</webParts>
<personalization defaultProvider="">
    <authorization>…</authorization>
    <providers>…</providers>
</personalization>
```

Section providers

La section `providers` permet d'ajouter des fournisseurs de personnalisation de WebParts et de spécifier leurs attributs :

```
<providers>
    <add name="String"
     type="String"
     connectionStringName="String"
     applicationName="String"
     commandTimeout="Integer"/>
</providers>
```

Attributs de l'élément add d'un SqlPersonalizationProvider

Attribut	Description
`applicationName`	Attribut `String` facultatif. Spécifie le nom de l'application pour laquelle on veut stocker et récupérer des informations de personnalisation.
`connectionStringName`	Attribut `String` requis. Spécifie la chaîne spécifique au fournisseur SQL utilisé pour établir la connexion à la base de données.
`commandTimeout`	Attribut `Int32` facultatif. Spécifie le nombre de secondes avant l'expiration du délai imparti à une commande émise dans la source de données de personnalisation WebParts.
`name`	Attribut `String` requis. Nom convivial du fournisseur.
`type`	Attribut `String` requis. Spécifie une référence d'assembly qualifiée complète à une classe qui implémente la classe `PersonalizationProvider` de base.

Par défaut, le fichier *web.config* utilise un fournisseur appelé `AspNetSqlPersonalizationProvider`, dont le nom de chaîne de connexion est `LocalSqlServer`

et dont le type est System.Web.UI.WebControls.WebParts.SqlPersonalization Provider :

```
<providers>
    <add connectionStringName="LocalSqlServer"
         name="AspNetSqlPersonalizationProvider"
         type="System.Web.UI.WebControls.WebParts.
               SqlPersonalizationProvider"
    />
</providers>
```

Si vous voulez l'utiliser, vous devez configurer votre base de données SQL ou SQL Express avec l'outil *aspnet_regsql.exe* situé dans le dossier *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

Section authorization

La syntaxe de la section authorization est similaire à celle utilisée pour spécifier les rôles et les utilisateurs autorisés ou refusés dans une application :

```
<authorization>
    <allow…/>
    <deny…/>
</authorization>
```

Elle dispose en plus d'un attribut verbs que vous pouvez utiliser pour déclarer les autorisations de modification de page et de basculement entre la portée utilisateur et partagée.

Les valeurs possibles sont donc les suivantes :

- enterSharedScope : indique si un utilisateur ou un rôle peut entrer dans la portée partagée.
- modifyState : indique si un utilisateur ou un rôle peut modifier des données de personnalisation pour la portée active.

Voici à quoi peut ressembler la section webParts complète de votre fichier *web.config* :

```
<webParts>
      <personalization
        defaultProvider="AspNetSqlPersonalizationProvider">
        <providers>
          <remove name="AspNetSqlPersonalizationProvider">
          </remove>
          <add name="AspNetSqlPersonalizationProvider"
  type="System.Web.UI.WebControls.WebParts.
        SqlPersonalizationProvider"
```

```
          connectionStringName="LocalSqlServer"
          applicationName="/WebPart" />
        </providers>
      </personalization>
<authorization>
     <deny users="*" verbs="enterSharedScope" />
     <allow users="*" verbs="modifyState" />
   </authorization>
   </webParts>
```

16.3 Construction de l'application

Maintenant que votre fournisseur est configuré, vous pouvez commencer à construire votre page de WebParts. Afin d'avoir une meilleure compréhension des éléments qui entrent en jeu, vous allez construire votre application en plusieurs étapes en découvrant à chaque fois quel contrôle `WebPart` vous devez utiliser et son utilité.

Vous allez également séparer chaque fonctionnalité dans des contrôles utilisateurs différents, ce qui vous permettra de mieux comprendre le rôle de chacun.

Ajout du WebPartManager

Le contrôle `WebPartManager` est l'élément principal qui gère les autres contrôles `WebPart` d'une page. Un seul contrôle `WebPartManager` est nécessaire dans une page qui utilise des WebParts. Voici sa syntaxe déclarative :

```
<asp:WebPartManager ID="wpManager" runat="server" />
```

Ce contrôle n'a aucun rendu, c'est-à-dire qu'il n'est pas visible au moment de l'affichage de la page. Son rôle est de gérer tous les contrôles `WebPart` de la page. Il gère des zones (régions qui contiennent des contrôles `WebPart` sur une page) et les contrôles qui se trouvent dans celles-ci. Il suit également et contrôle les différents modes d'affichage d'une page (navigation, connexion, modification ou catalogue), et vérifie si les modifications de personnalisation s'appliquent à tous les utilisateurs ou aux utilisateurs individuels. Enfin, il initialise et suit les connexions et la communication entre des contrôles `WebPart`.

Ajout des contrôles WebPartZone

Le contrôle `WebPartZone` se charge de la disposition complète pour les contrôles `WebPart` qui composent l'interface utilisateur principale d'une page. Vous pouvez décomposer votre page en plusieurs zones dans lesquelles vous organiserez les différents WebParts que vous voulez utiliser.

La prochaine étape est donc de créer une page contenant deux contrôles `WebPartZone` dans lesquels vous pourrez déposer vos WebParts, les organiser et les personnaliser à votre guise. La manière la plus simple est de les placer dans un tableau à deux colonnes :

```
<table border="0" width="100%" cellpadding="2"
       cellspacing="2">
    <tr>
        <td valign="top" height="100%">
            <asp:WebPartZone
                ID="LeftWebPartZone"
                HeaderText="Gauche" runat="server">
                <ZoneTemplate>
                </ZoneTemplate>
            </asp:WebPartZone>
        </td>
        <td valign="top" height="100%">
            <asp:WebPartZone ID="RightWebPartZone"
                HeaderText="Droite" runat="server">
                <ZoneTemplate>
                </ZoneTemplate>
            </asp:WebPartZone>
        </td>
    </tr>
</table>
```

▲ **Figure 16-1 :** Affichage d'une page avec une zone gauche et une zone droite

Les zones servent de gestionnaires de présentation sur une page WebPart. Les fonctionnalités disponibles pour ces zones et l'affichage dépendent du mode d'affichage en cours.

Intéressons-nous à présent aux différents modes d'affichage qui existent et à la façon de basculer de l'un à l'autre.

Modes d'affichage

Voici les modes d'affichage par défaut vers lesquels l'utilisateur peut basculer s'il a les droits appropriés. Ces modes héritent tous de la classe abstraite `WebPartDisplayMode`.

Valeurs possibles de l'énumération WebPartDisplayMode	
Mode	**Description**
`BrowseDisplayMode`	Affichage normal de consultation des contrôles `WebPart`.
`DesignDisplayMode`	Permet à l'utilisateur de faire glisser les contrôles `WebPart` pour modifier la disposition d'une page.
`EditDisplayMode`	Permet à l'utilisateur de modifier les contrôles d'une page.
`CatalogDisplayMode`	Permet aux utilisateurs d'ajouter et de supprimer des contrôles sur une page à l'aide des catalogues qui listent les WebParts disponibles.
`ConnectDisplayMode`	Permet aux utilisateurs de connecter des contrôles `WebPart` entre eux.

Pour basculer facilement entre les modes d'affichage, vous allez créer un contrôle utilisateur qui va contenir des boutons dont les actions auront pour effet de modifier le mode d'affichage en cours.

1. Dans le contrôle utilisateur, insérez pour chaque mode d'affichage disponible un contrôle `LinkButton` qui servira à faire basculer la page dans ce mode.

```
<asp:LinkButton ID="lnkBtnBrowse" runat="server">
    Normal
</asp:LinkButton>
```

▲ **Figure 16-2 :** Boutons de basculement du mode d'affichage

2. Au chargement de la page, vérifiez pour chaque mode s'il est pris en charge, et masquez ou affichez le lien en conséquence. La collection `SupportedDisplayModes` du contrôle `WebPartManager` vous aidera à faire ce test.

```
Protected Sub PageLoad(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
        lnkBtnBrowse.Visible = wpManager. _
            SupportedDisplayModes.Contains _
            (WebPartManager.BrowseDisplayMode)
    End Sub
```

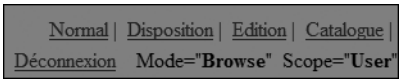
3. Ajoutez également des gestionnaires d'événements sur le clic des boutons afin de faire basculer la page dans le mode sélectionné.

```
Protected Sub lnkBtnBrowse_Click(ByVal sender As Object, _
  ByVal e As System.EventArgs) Handles lnkBtnBrowse.Click
   wpManager.DisplayMode = WebPartManager.BrowseDisplayMode
  End Sub
```

4 Vous pouvez également ajouter des contrôles `Label` indiquant la portée de la page ainsi que le mode en cours obtenu.

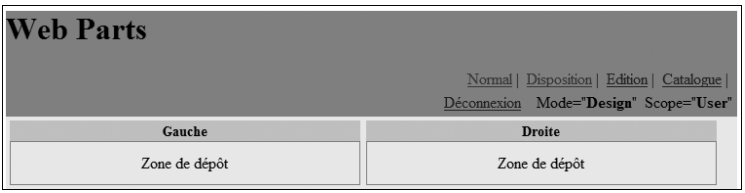
```
Mode="<%= wpManager.DisplayMode.Name %>"
Scope="<%= wpManager.Personalization.Scope.ToString() %>"
```

Au chargement de la page web, vous obtiendrez cet affichage. Vous pouvez constater que le mode d'affichage de connexions entre WebParts n'est pas disponible et que la portée est au niveau de l'utilisateur. En outre, le mode d'affichage en cours est `Browse`, ce qui correspond au mode normal.



◄ **Figure 16-3 :** Boutons de basculement dans une page web

Un clic sur le bouton **Disposition** permet de passer en mode Design, qui fait apparaître les deux zones gauche et droite et qui permettra plus tard de déplacer les WebParts.



▲ **Figure 16-4 :** Page en mode Design

Maintenant que le cadre est mis en place, il faut un contrôle `WebPart` à disposition. Vous pouvez en créer un vous-même, comme s'il s'agissait d'un contrôle serveur personnalisé, mais cela s'avère assez compliqué puisque vous devez tout créer par programmation. Heureusement, il existe une façon simple pour transformer un contrôle utilisateur en WebPart.

Transformer un contrôle utilisateur en WebPart

Cette méthode a pour avantage de vous laisser la possibilité de créer l'interface de votre WebPart à l'aide du designer de contrôle utilisateur. Elle ne nécessite ensuite que quelques étapes de configuration. Il s'agit d'ajouter les propriétés qui vont faire de votre contrôle utilisateur, un contrôle `WebPart` à part entière. Il suffit pour cela d'implémenter l'interface `IWebPart`.

Interface IWebPart

L'interface `IWebPart` fournit les méthodes qui doivent être implémentées si vous voulez créer un WebPart. Voici les propriétés concernées et leurs rôles :

Propriétés à implémenter de l'interface IWebPart	
Nom	**Description**
`CatalogIconImageUrl`	Obtient ou définit l'URL d'une image qui représente un contrôle `WebPart` dans un catalogue de contrôles.
`Description`	Obtient ou définit une brève expression qui résume la fonction d'un contrôle, en vue d'une utilisation dans les info-bulles et les catalogues de contrôles `WebPart`.
`Subtitle`	Obtient une chaîne concaténée avec la valeur de la propriété `Title` pour constituer le titre complet d'un contrôle `WebPart`, ce qui permet de différencier deux WebParts qui auraient le même titre.
`Title`	Obtient ou définit le titre d'un contrôle `WebPart`.
`TitleIconImageUrl`	Obtient ou définit l'URL d'une image utilisée pour représenter un contrôle `WebPart` dans la propre barre de titre du contrôle.
`TitleUrl`	Obtient ou définit une URL vers des informations supplémentaires relatives à un contrôle `WebPart`.

Afin d'avoir à disposition une classe dont vous pouvez vous servir pour créer facilement des WebParts à partir de contrôles utilisateurs, vous pouvez avoir recours à une petite astuce qui consiste à déclarer une classe abstraite qui hérite de `System.Web.UI.UserControl` et qui implémente l'interface `IWebPart`.

Le code fournit une implémentation de base pour les propriétés que vous venez de voir :

```
Public MustInherit Class ucWebPart
    Inherits System.Web.UI.UserControl
    Implements IWebPart

    Private _catalogImageUrl As String = String.Empty
    Private _description As String = String.Empty
    Private _subTitle As String = "[0]"
    Private _title As String = "Titre"
    Private _titleUrl As String = String.Empty
    Private _titleIconImageUrl As String = String.Empty

    Public Overridable Property CatalogIconImageUrl() _
    As String Implements IWebPart.CatalogIconImageUrl
        Get
```

```
            Return _catalogImageUrl
        End Get
        Set(ByVal value As String)
            _catalogImageUrl = value
        End Set
    End Property
    Public Overridable Property Description() As String _
    Implements IWebPart.Description
        Get
            Return _description
        End Get
        Set(ByVal value As String)
            _description = value
        End Set
    End Property
    Public Overridable ReadOnly Property Subtitle() _
    As String Implements IWebPart.Subtitle
        Get
            Return _subTitle
        End Get
    End Property
    Public Property Title() As String _
    Implements IWebPart.Title
        Get
            Return _title
        End Get
        Set(ByVal value As String)
            _title = value
        End Set
    End Property
    Public Property TitleIconImageUrl() As String _
    Implements IWebPart.TitleIconImageUrl
        Get
            Return _titleIconImageUrl
        End Get
        Set(ByVal value As String)
            _titleIconImageUrl = value
        End Set
    End Property
    Public Property TitleUrl() As String _
    Implements IWebPart.TitleUrl
        Get
            Return _titleUrl
        End Get
        Set(ByVal value As String)
            _titleUrl = value
        End Set
    End Property
End Class
```

Vous n'avez ainsi à déclarer ce code qu'une seule fois et vous pouvez en hériter à chaque fois que vous voulez transformer un contrôle utilisateur en WebPart. Pour cela, vous devez créer un contrôle utilisateur comme si vous développiez une page puis hériter de la classe que vous venez de créer en personnalisant les propriétés de base selon vos besoins :

```
Partial Class UCWebPart
    Inherits ucWebPart
    Public Sub New()
        MyBase.Title = "Titre personnalisé"
    End Sub
End Class
```

L'exemple de WebPart que vous allez créer est basique puisqu'il ne contiendra qu'une propriété `Text` que vous allez rendre "personnalisable" afin de pouvoir modifier le texte à afficher.

16.4 Ajout d'une propriété "personnalisable" à un WebPart

Pour qu'une propriété puisse être utilisée par un WebPart, vous devez lui ajouter des attributs, dont certains seront utiles au moment de l'édition du WebPart :

- `WebBrowsable` indique si la propriété sera affichée dans l'éditeur.
- `WebDisplayName` définit le nom convivial de la propriété qui apparaîtra comme intitulé dans l'éditeur.
- `WebDescription` définit la description à utiliser comme info-bulle.
- `Personalizable` détermine si la propriété est "personnalisable" et dans quelle portée.

Voici le code correspondant à la propriété `Text` :

```
    <WebBrowsable(), _
    WebDisplayName("Texte"), _
    WebDescription("Texte à afficher"), _
    Personalizable()> _
    Public Property Text() As String
        Get
            Return m_text
        End Get
        Set(ByVal value As String)
            m_text = value
        End Set
    End Property
```

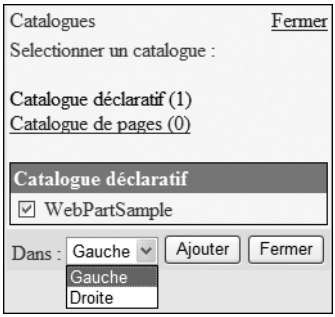
Maintenant que vous avez un contrôle WebPart à disposition, vous pouvez ajouter une troisième colonne aux tableaux contenant les zones, qui va servir à sélectionner les WebParts que vous voulez déposer dans celles-ci et à les éditer ensuite.

Ajout de WebParts à partir d'un catalogue

CatalogZone Le contrôle CatalogZone contient des contrôles de type CatalogPart.

Vous pouvez utiliser cette zone pour créer un catalogue des contrôles WebPart que les utilisateurs pourront sélectionner et ajouter à une page.

Lorsque vous cliquez sur le lien *Catalogue*, le catalogue des WebParts disponibles s'affiche. Vous pouvez alors cocher les WebParts que vous voulez insérer et la zone cible.



◄ **Figure 16-5 :** Catalogue des WebParts disponibles

Voici une présentation des différents catalogues disponibles.

Contrôle DeclarativeCatalogPart

DeclarativeCatalogPart Le contrôle DeclarativeCatalogPart permet de proposer un catalogue des contrôles WebPart. Vous pouvez y ajouter de façon déclarative les contrôles que vous avez implémentés.

```
<%@ Register Src="WebPartSample.ascx"
    TagName="WebPartSample" TagPrefix="uc1" %>
<asp:DeclarativeCatalogPart ID="DeclarativeCatalogPart"
    runat="server">
      <WebPartsTemplate>
        <uc1:WebPartSample ID="WebPartSample1"
            runat="server" />
      </WebPartsTemplate>
</asp:DeclarativeCatalogPart>
```

Contrôle ImportCatalogPart

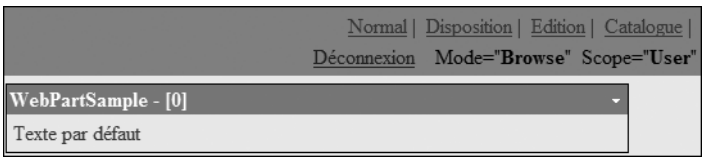
Le contrôle `ImportCatalogPart` permet d'importer un fichier de description pour un contrôle `WebPart`, afin que les utilisateurs puissent ajouter le contrôle à une page web avec des paramètres prédéfinis. Ces fichiers de description possèdent une extension *.webpart*.

Contrôle PageCatalogPart

Le contrôle `PageCatalogPart` est un catalogue qui conserve les références à tous les contrôles `WebPart` qu'un utilisateur a fermés sur la page `WebPart` en cours afin de lui permettre de restaurer les contrôles qu'il a supprimés.

Manipulation d'un contrôle WebPart

Vous pouvez enfin découvrir les différentes fonctionnalités d'un WebPart une fois qu'il a été ajouté dans une zone. En mode normal, le contrôle prend quasiment toute la place en largeur puisque la seconde zone est vide.



▲ **Figure 16-6 :** Page en mode normal

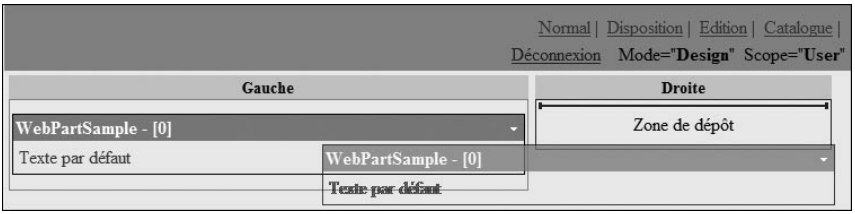
Le WebPart dispose de plusieurs actions possibles accessibles notamment à partir d'un menu et qui varient selon le mode d'affichage en cours. Une icône à droite du WebPart permet d'afficher ce menu. Il contient toujours un lien *Fermer* qui supprime le WebPart de la zone même si vous pouvez le récupérer à partir du catalogue de pages.

◄ **Figure 16-7 :** Menu par défaut

Le lien *Réduire* permet d'afficher uniquement la barre de titre. Le texte de ce lien devient alors *Restaurer* et permet de réafficher la totalité du contenu du WebPart.

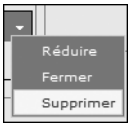
◄ **Figure 16-8 :** Lien Restaurer

La fonctionnalité la plus visuelle et la plus simple d'utilisation est le déplacement des WebParts entre les zones ou à l'intérieur d'une même zone. Pour cela, vous devez être en mode Design. En cliquant sur la barre de titre sans relâcher le bouton, vous pouvez faire glisser une copie transparente du WebPart ; une barre qui apparaît au survol d'une zone accessible indique à quel endroit vous pouvez le déposer.



▲ **Figure 16-9 :** Déplacement d'un WebPart

Dans ce mode, un lien *Supprimer* est présent dans le menu et permet de supprimer définitivement le contrôle de la page.

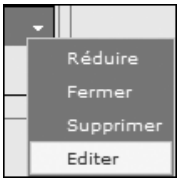


◄ **Figure 16-10 :** Lien Supprimer

Édition d'un contrôle WebPart

Le contrôle `EditorZone` correspond à la zone chargée de l'édition d'un WebPart. Vous pouvez utiliser cette zone pour permettre aux utilisateurs de modifier et de personnaliser les contrôles `WebPart` d'une page.

En mode Edit, le lien *Éditer* du menu sert à afficher les éditeurs déclarés à l'intérieur de la balise `<ZoneTemplate>` du contrôle `EditorZone`.



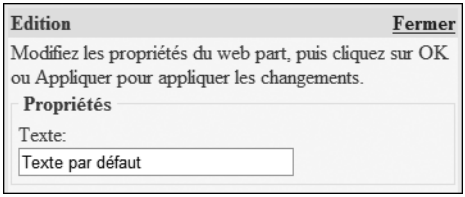
◄ **Figure 16-11 :** Lien Éditer

Il contient des contrôles de type `EditorPart` dont voici un aperçu.

PropertyGridEditorPart

Le contrôle `PropertyGridEditorPart` permet de modifier des propriétés personnalisées d'un contrôle `WebPart`.

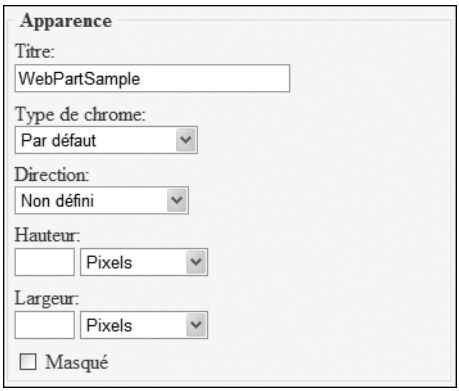
Le contrôle de saisie associé dépend du type de la propriété. Par exemple, une chaîne de caractères sera modifiable à l'aide d'une `TextBox` tandis qu'un booléen sera représenté par une `CheckBox` et une énumération pourra avoir une `DropDownList`.



◄ **Figure 16-12 :** Éditeur des propriétés personnalisées

AppearanceEditorPart

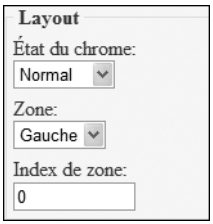
AppearanceEditorPart Le contrôle `AppearanceEditorPart` permet de modifier plusieurs propriétés d'interface utilisateur relatives à l'apparence d'un contrôle `WebPart`, comme sa taille ou son titre.



◄ **Figure 16-13 :** Éditeur des propriétés d'apparence

LayoutEditorPart

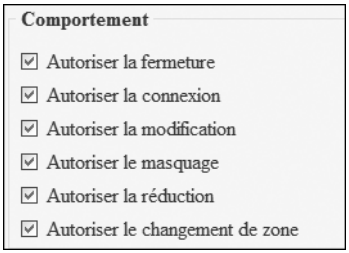
LayoutEditorPart Le contrôle `LayoutEditorPart` permet de modifier plusieurs propriétés d'interface utilisateur concernant la disposition d'un contrôle `WebPart` dans une zone.



◄ **Figure 16-14 :** Éditeur des propriétés de disposition

BehaviorEditorPart

Le contrôle `BehaviorEditorPart` permet de modifier plusieurs propriétés d'interface utilisateur concernant le comportement et les autorisations d'un contrôle `WebPart`.



◄ **Figure 16-15 :** Éditeur des propriétés de comportement

16.5 Check-list

Votre page de WebParts est maintenant terminée. Les WebParts fournissent des fonctionnalités extrêmement puissantes permettant de créer des applications que l'utilisateur peut personnaliser.

Voici les concepts et les fonctionnalités présentés dans ce chapitre :

- créer une page de WebParts ;
- ajouter des contrôles `WebPartZone` à une page ;
- basculer entre les différents modes d'affichage ;
- transformer un contrôle utilisateur en WebPart ;
- ajouter une propriété "personnalisable" à un WebPart ;
- manipuler les WebParts (actions, disposition…) ;
- ajouter un catalogue de WebParts (`CatalogZone`) ;
- ajouter des éditeurs de WebParts (`WebPartZone`).

Stockage d'informations dans un profil

Dans de nombreuses applications qui gèrent des utilisateurs, il faut stocker des informations différentes sur chacun d'entre eux. La personnalisation d'une application nécessite certains développements. Vous devez stocker les informations à l'aide d'un identifiant d'utilisateur unique, reconnaître des utilisateurs lorsqu'ils reviennent sur le site, puis extraire les informations nécessaires à leur sujet. Pour faciliter vos développements, vous pouvez utiliser la fonctionnalité de profil ASP .NET, qui permet de gérer la totalité de ces tâches en toute transparence.

La fonctionnalité de profil ASP .NET associe des informations à un utilisateur donné et les stocke sous un format persistant. Elle permet de les gérer facilement sans qu'il soit nécessaire de créer ou de gérer une base de données personnelle. Elle les met à votre disposition à l'aide d'une classe qui contient des propriétés fortement typées, auxquelles vous pouvez accéder facilement depuis n'importe quel endroit de votre application. Elle permet en outre de stocker des objets de tout type en sérialisant ces objets en binaire ou en XML. Il s'agit d'une fonction de stockage générique.

17.1 Classes et espaces de noms utilisés

La classe `System.Web.Profile.ProfileBase` héritée de `SettingsBase` fournit le modèle à respecter si vous voulez développer votre propre fournisseur de profils. Voici les propriétés héritées et ajoutées :

Propriétés de la classe System.Web.Profile.ProfileBase	
Propriété	**Description**
`Context`	Obtient le contexte de paramètres associé.
`IsAnonymous`	Obtient une valeur qui indique si le profil utilisateur est destiné à un utilisateur anonyme.
`IsDirty`	Obtient une valeur qui indique si l'une des propriétés de profil a été modifiée.
`Item`	Substitué (hérité de la classe de base). Obtient ou définit une valeur de propriété de profil indexée par le nom de propriété.
`LastActivityDate`	Obtient la date et l'heure de dernière lecture ou de modification du profil.
`LastUpdatedDate`	Obtient la date et l'heure de dernière modification du profil.
`Properties`	Obtient une collection d'objets `SettingsProperty` pour chaque propriété dans le profil.

Propriétés de la classe System.Web.Profile.ProfileBase	
Propriété	**Description**
`PropertyValues`	Obtient une collection de valeurs de propriété de paramètre.
`Providers`	Obtient une collection de fournisseurs de paramètres.
`UserName`	Obtient le nom d'utilisateur du profil.

17.2 Configuration

La section `profile` du fichier *Web.config* se compose d'une collection de propriétés dont il faut conserver les valeurs et d'une collection de fournisseurs personnalisés. Son prototype se présente comme suit :

```
<profile defaultProvider="SqlProvider">
      <providers>
      …
      </providers>
      <properties>
      …
      </properties>
</profile>
```

Les tableaux suivants présentent les syntaxes et les attributs que vous pouvez employer pour configurer vos propriétés et vos fournisseurs de profils :

Attributs de l'élément profile	
Attribut	**Description**
`enabled`	Attribut `Boolean` facultatif. Valeur par défaut : `true`. Spécifie si les profils utilisateurs ASP .NET sont activés. Ils le sont si la valeur est `true`.
`defaultProvider`	Attribut `String` facultatif. Valeur par défaut : `AspNetSqlProfileProvider`. Spécifie le nom du fournisseur de profils par défaut.
`inherits`	Attribut `String` facultatif. Contient une référence de type pour un type personnalisé qui dérive de la classe abstraite `ProfileBase`. Une classe `ProfileCommon` qui hérite de ce type est générée dynamiquement et enregistrée dans la propriété `Profile` du `HttpContext` en cours.

Attributs de l'élément profile	
Attribut	**Description**
`automaticSaveEnabled`	Attribut `Boolean` facultatif. Valeur par défaut : `true`. Indique si le profil utilisateur est automatiquement enregistré à la fin de l'exécution d'une page ASP .NET. Il l'est si la valeur est `true`. L'objet `ProfileModule` enregistre un profil utilisateur uniquement si le module détecte que le profil a été modifié, en d'autres termes si la propriété `IsDirty` est `true`.

Déclaration d'une propriété de profil

La déclaration d'une propriété de profil se fait dans le fichier de configuration de l'application, *Web.config*.

Dans la section `profile`, créez une section `properties` qui va contenir la liste des propriétés à attribuer au profil des utilisateurs. L'ajout d'une propriété se fait de manière simple : insérez au minimum une balise `add`, en spécifiant le nom de la propriété à ajouter grâce à l'attribut `name`. Le type par défaut d'une propriété est la chaîne de caractères (`string`). Si vous avez besoin d'un autre type, pour spécifier l'âge de l'utilisateur par exemple, servez-vous de l'attribut `type` et spécifiez le type désiré, par exemple `System.Int32`. Exemple :

```
<properties>
    <add name="Age" type="System.Int32" />
    <add name="Adresse"/>
</properties>
```

Voici la syntaxe générale de déclaration de propriétés de profil :

```
<properties>
    <add... />
    <clear />
    <remove... />
    <group name="group name">...</group>
</properties>

<add
    name="Nom de la propriété"
    type="Type de la propriété"
    provider="Provider"
    serializeAs="String|Xml|Binary|Personnalisé…"
    allowAnonymous="true|false"
    defaultValue="Valeur par défaut"
    readOnly="true|false"/>
/>
```

Attributs de l'élément <add> de la collection properties de l'élément profile	
Attribut	**Description**
`name`	Attribut `String` requis. Spécifie le nom de la propriété. Cette valeur est utilisée comme nom de la propriété pour la classe de profil générée automatiquement et comme valeur d'index pour la propriété dans la collection `Properties`. Le nom de la propriété ne peut pas contenir de point.
`type`	Attribut `String` facultatif. Valeur par défaut : `String`. Spécifie le type de la propriété.
`provider`	Attribut `String` facultatif. Spécifie le fournisseur de profils utilisé pour stocker et récupérer des valeurs de la propriété. La valeur de l'attribut `provider` est le nom de l'un des fournisseurs de profils spécifiés dans l'élément `providers`. Si aucun nom de fournisseur n'est spécifié, le fournisseur par défaut spécifié dans l'élément `profile` est utilisé.
`serializeAs`	Attribut facultatif. Spécifie le format de sérialisation de la valeur de propriété dans le magasin de données. Le format de sérialisation utilisé par défaut est spécifique au fournisseur. C'est le fournisseur qui détermine la sérialisation utilisée, à savoir la sérialisation `String` ou `Binary` dans le cas du fournisseur SQL.
`allowAnonymous`	Attribut `Boolean` facultatif. Valeur par défaut : `false`. Spécifie si la propriété peut être obtenue ou définie, si l'utilisateur de l'application est anonyme.
`defaultValue`	Attribut `String` facultatif. Spécifie la valeur par défaut, en l'absence d'une valeur de la propriété `Profile`.
`readOnly`	Attribut `Boolean` facultatif. Valeur par défaut : `false`. Spécifie si la propriété est en lecture seule.

Déclaration d'un fournisseur de profils personnalisé

Si vous voulez utiliser une autre chaîne de connexion ou personnaliser votre fournisseur de profils, vous pouvez en ajouter un. Spécifiez ensuite dans l'élément `profile` parent qu'il s'agit du fournisseur par défaut en donnant son nom à l'attribut `defaultProvider`.

Remarque

Ajouter un fournisseur ou redéfinir celui par défaut

Il est fortement conseillé de toujours ajouter un fournisseur ou de redéfinir celui par défaut, ne serait-ce que pour spécifier l'attribut `applicationName` : donnez-lui le nom correspondant à l'application que vous développez. Cela est notamment indispensable si vous utilisez une seule base de données pour gérer plusieurs applications.

```
<add name="SqlProfileProvider"
    type="System.Web.Profile.SqlProfileProvider"
    connectionStringName="ConnectionString"
    applicationName="Application"
    description="Description" />
```

Attributs de l'élément <add> de la collection providers de l'élément profile

Attribut	Description
`name`	Attribut `String` requis. Spécifie le nom de l'instance de fournisseur. Il s'agit de la valeur utilisée par l'attribut `defaultProvider` de l'élément `<profile>` pour désigner l'instance de fournisseur comme fournisseur de profils par défaut. `name` est également utilisé pour indexer le fournisseur dans la collection `Providers`.
`type`	Attribut `String` requis. Spécifie le type qui implémente la classe de base abstraite `ProfileProvider`.
`connectionStringName`	Attribut `String` requis. Spécifie le nom d'une chaîne de connexion définie dans l'élément `<connectionStrings>`. La chaîne de connexion spécifiée sera utilisée par le fournisseur qui est ajouté.
`applicationName`	Attribut `String` facultatif. Spécifie le nom de l'application sous laquelle les données de profil sont stockées dans la source de données. Le nom d'application permet à plusieurs applications ASP. NET d'utiliser la même base de données sans rencontrer de données de profil en double. Plusieurs applications ASP .NET peuvent également utiliser les mêmes informations de profil si l'on spécifie le même nom d'application. Les fournisseurs de profils inclus dans le Framework .NET utilisent la valeur `ApplicationPath` pour la propriété `ApplicationName` si cet attribut n'est pas spécifié.

Attributs de l'élément <add> de la collection providers de l'élément profile	
Attribut	**Description**
`commandTimeout`	Attribut `Int32` facultatif. Valeur par défaut (pour ADO .NET) : 30. Spécifie le nombre de secondes avant l'expiration du délai imparti à une commande envoyée à la source de données d'appartenance.
`description`	Attribut `String` facultatif. Spécifie une description de l'instance de fournisseur de profils.

17.3 Persistance des données

Maintenant que vous savez déclarer des profils au sein de votre application, vous devez mettre en place la persistance des informations qu'ils contiennent, les sauvegarder afin qu'elles puissent être récupérées ultérieurement par ASP .NET lorsque l'utilisateur se connectera à votre site ou lorsque vous les afficherez.

Cette persistance s'effectue grâce à des fournisseurs de données spécialisés dans la sauvegarde de données liées aux profils. Ces classes héritent toutes de `System.Configuration.SettingsProvider`.

Membres de la classe System.Configuration.SettingsProvider	
Méthode	**Description**
`GetPropertyValues`	Prend en paramètre des objets `SettingsContext` et `SettingsPropertyCollection`. `SettingsContext` fournit des informations sur l'utilisateur. Vous pouvez les utiliser comme clé primaire pour récupérer des informations de propriété de profil sur l'utilisateur. Servez-vous de l'objet `SettingsContext` pour obtenir le nom de l'utilisateur et pour savoir s'il est authentifié ou anonyme. `SettingsPropertyCollection` contient une collection d'objets `SettingsProperty`. Chaque objet `SettingsProperty` fournit le nom et le type de la propriété, ainsi que des informations supplémentaires, comme la valeur par défaut de la propriété et si elle est ou non en lecture seule. La méthode `GetPropertyValues` remplit `SettingsPropertyValueCollection` avec les objets `SettingsPropertyValue` établis d'après les objets `SettingsProperty` utilisés comme entrée.

Membres de la classe System.Configuration.SettingsProvider	
Méthode	**Description**
	Les valeurs de la source de données sur l'utilisateur spécifié sont affectées aux propriétés `PropertyValue` de chaque objet `SettingsPropertyValue` et la collection entière est retournée. L'appel de cette méthode met également à jour la valeur `LastActivityDate` du profil utilisateur spécifié, avec la date et l'heure courantes.
`SetPropertyValues`	Prend en paramètre des objets `SettingsContext` et `SettingsPropertyValueCollection`. `SettingsContext` fournit des informations sur l'utilisateur. Vous pouvez les utiliser comme clé primaire pour récupérer des informations de propriété de profil sur l'utilisateur. Servez-vous de l'objet `SettingsContext` pour obtenir le nom de l'utilisateur et pour savoir s'il est authentifié ou anonyme. `SettingsPropertyValueCollection` contient une collection d'objets `SettingsPropertyValue`. Chaque objet `SettingsPropertyValue` fournit le nom, le type et la valeur de la propriété, ainsi que des informations supplémentaires, comme la valeur par défaut de la propriété et si elle est ou non en lecture seule. La méthode `SetPropertyValues` met à jour les valeurs des propriétés de profil dans la source de données sur l'utilisateur spécifié. L'appel de cette méthode met également à jour les valeurs `LastActivityDate` et `LastUpdatedDate` du profil utilisateur spécifié, avec la date et l'heure courantes.

`SettingsProvider` est une classe d'assez bas niveau. Créer une classe héritant de celle-ci demanderait pas mal de travail. C'est pourquoi Microsoft propose une classe abstraite plus spécialisée, nommée `System.Web.Profile.ProfileProvider`. Ainsi si vous souhaitez créer votre propre fournisseur de données, vous n'avez qu'à créer une classe héritant de `ProfileProvider` et à implémenter la logique technique correspondant à votre besoin (stockage dans Oracle, dans un fichier plat…).

Bien évidemment, Microsoft ne vous oblige pas à créer votre propre fournisseur de données à chaque fois que vous souhaitez implémenter une gestion de profils, et propose un fournisseur de données par défaut nommé `System.Web.Profile.SqlProfileProvider`. Celui-ci permet d'enregistrer les données liées aux profils directement dans une base de données SQL Server, qui peut être créée à l'aide d'un Assistant.

La fonctionnalité de profil ASP .NET utilise la même structure à base de fournisseur que celle utilisée par l'appartenance (la gestion des membres), la gestion des rôles ou d'autres fonctionnalités ASP .NET. Elle s'appuie sur des fournisseurs de profils pour effectuer les tâches principales (stocker et récupérer les valeurs des propriétés de profil entre autres).

Fournisseur de profils

ASP .NET comprend un fournisseur de profils qui stocke des données à l'aide d'une base SQL Server. La configuration ASP .NET par défaut de l'ordinateur contient et utilise une instance de `SqlProfileProvider` nommée `AspNetSqlProfileProvider`. Vous pouvez spécifier un fournisseur par défaut différent dans le fichier *Web.config* de votre application.

Pour utiliser le `SqlProfileProvider`, vous devez créer la base de données SQL Server dont il a besoin. En ce sens, exécutez le programme *Aspnet_regsql.exe* qui se trouve dans le répertoire *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*.

Lorsque vous exécutez cet outil, spécifiez l'option `–Ap` pour activer la fonctionnalité de profil sur une base de données et ainsi stocker des profils ASP .NET à l'aide du `SqlProfileProvider`.

Utilisateurs anonymes

Les profils peuvent également fonctionner avec des utilisateurs anonymes. La prise en charge des profils anonymes n'étant pas activée par défaut, vous devez l'activer explicitement. De plus, lorsque vous définissez des propriétés de profil dans le fichier *Web.config*, vous devez les rendre explicitement accessibles pour les utilisateurs anonymes individuellement. Les propriétés de profil ne prennent pas en charge l'accès anonyme par défaut.

Si l'identification anonyme est activée, ASP .NET crée une identification unique pour les utilisateurs la première fois qu'ils visitent votre site. L'identification utilisateur unique est stockée dans un cookie sur l'ordinateur du visiteur afin que ce dernier puisse être identifié à chaque demande de page.

```
<anonymousIdentification
   enabled="[true | false]"
   cookieless=
"[UseUri | UseCookies | AutoDetect | UseDeviceProfile]"
   cookieName=""
   cookiePath=""
cookieProtection="[None | Validation | Encryption | All]"
   cookieRequireSSL="[true | false]"
   cookieSlidingExpiration="[true | false]"
   cookieTimeout="[DD.HH:MM:SS]"
```

```
   domain="domaine.ext"
/>
```

Attributs de l'élément anonymousIdentification	
Attribut	**Description**
cookieless	Spécifie s'il faut utiliser des cookies dans une application web. L'énumération HttpCookieMode permet de spécifier la valeur de cet attribut dans la section de configuration. Elle est utilisée par toutes les fonctionnalités qui prennent en charge l'authentification sans cookie. Lorsque la valeur AutoDetect est spécifiée, ASP .NET interroge le navigateur ou le périphérique pour déterminer s'il prend en charge les cookies. Si c'est le cas, ces derniers sont utilisés pour rendre les données utilisateurs persistantes ; sinon, un identificateur est saisi dans la chaîne de requête. Cet attribut peut avoir l'une des valeurs suivantes : Valeur, Description. AutoDetect : indique que ASP .NET doit déterminer si le navigateur ou le périphérique à l'origine de la demande prend en charge des cookies. Si c'est le cas, AutoDetect utilise des cookies pour rendre les données utilisateurs persistantes ; sinon, un identificateur est saisi dans la chaîne de requête. Si le navigateur ou le périphérique prend en charge les cookies alors qu'ils sont actuellement désactivés, ils sont néanmoins utilisés par la fonctionnalité qui effectue la demande. UseCookies : spécifie que les cookies doivent être utilisés pour rendre les données utilisateurs persistantes, que le navigateur ou le périphérique prenne en charge ou non les cookies. Il s'agit de l'option par défaut. UseDeviceProfile : indique que ASP .NET doit déterminer s'il faut utiliser des cookies sur la base du paramètre HttpBrowserCapabilities. Si le paramètre indique que le navigateur ou le périphérique prend en charge les cookies, ces derniers sont employés ; sinon, un identificateur est utilisé dans la chaîne de requête. UseUri : indique que la fonctionnalité appelante doit utiliser la chaîne de requête pour stocker un identificateur, que le navigateur ou le périphérique prenne en charge ou non les cookies. Valeur par défaut : UseCookies.
cookieName	Valeur par défaut : .ASPXANONYMOUS. Spécifie le nom du cookie.

Attributs de l'élément anonymousIdentification	
Attribut	**Description**
`cookiePath`	Valeur par défaut : le répertoire racine spécifié par `"/"`. Spécifie le chemin d'accès au répertoire de stockage du cookie. Le chemin d'accès respecte la casse.
`cookieProtection`	Spécifie la méthode de protection contre les cookies.
`cookieRequireSSL`	Valeur par défaut : `false`. Spécifie si le cookie exige une connexion SSL lorsqu'il est transmis au client. Comme ASP .NET définit la propriété de cookie d'authentification, `Secure`, le client ne retourne pas le cookie à moins qu'une connexion SSL soit utilisée.
`cookieSlidingExpiration`	Attribut `Boolean` requis. Valeur par défaut : `true`. Spécifie si le délai d'expiration d'un cookie est réinitialisé à chaque demande ou selon un intervalle de temps fixe prédéfini. Si la valeur est `true`, le cookie expire lorsqu'il reste moins de 50 % de durée de vie (TTL, Time-To-Live). Si la valeur est `false`, le cookie expire une fois la durée `cookieTimeout` écoulée.
`cookieTimeout`	Attribut `TimeSpan` requis. Valeur par défaut : 10 000 minutes. Spécifie le délai d'expiration d'un cookie en minutes.
`domain`	Valeur par défaut : une chaîne vide (`""`). Spécifie le domaine du cookie. Cet attribut autorise le partage du cookie d'identification anonyme entre les domaines qui disposent d'un espace de noms DNS commun. Les sites doivent alors partager des clés de déchiffrement et de validation. D'autres attributs de configuration de l'identification anonyme, tels que le chemin d'accès et le nom du cookie, doivent être identiques pour tous les sites.
`enabled`	Attribut `Boolean` facultatif. Valeur par défaut : `false`. Spécifie si l'identification anonyme est activée. Si la valeur est `true`, un cookie (ou une valeur sans cookie) est utilisé pour gérer l'identificateur anonyme de l'utilisateur.

Migration des informations de profil anonyme

Dans certains cas, une propriété de profil doit être accessible pour les utilisateurs anonymes. Il peut être utile de stocker des informations de profils pour les visiteurs d'un site web, notamment lorsque l'on souhaite implémenter une gestion de paniers. Par exemple, les sites d'e-commerce proposent à tous

leurs utilisateurs, de rajouter des produits à un panier, ces utilisateurs devant s'authentifier avant de valider leur commande.

La difficulté consiste à faire migrer les informations d'un profil anonyme vers un profil utilisateur au moment où celui-ci se connecte.

Vous pouvez gérer l'événement `MigrateAnonymous` dans le fichier *Global.asax* pour effectuer une migration des informations de l'identité anonyme de l'utilisateur vers la nouvelle identité authentifiée.

L'exemple de code suivant montre comment effectuer une migration des informations au moment où l'utilisateur s'authentifie :

```
Public Sub Profile_OnMigrateAnonymous( _
sender As Object, args As ProfileMigrateEventArgs)
  ' Migration vers le profil authentifié.
  Dim anonymousProfile As ProfileCommon = _
  ➥ Profile.GetProfile(args.AnonymousID)
  Profile.PropertyName = anonymousProfile.PropertyName

  ' Suppression du profil anonyme.
  ProfileManager.DeleteProfile(args.AnonymousID)
  AnonymousIdentificationModule.ClearAnonymousIdentifier()
End Sub
```

17.4 Check-list

Les différentes utilisations de la fonctionnalité de profil ASP .NET présentées ici montrent à quel point il est simple de stocker des informations sur un utilisateur et de les récupérer.

Dans ce chapitre, vous avez appris à :

- stocker des informations sur un client ;
- faire migrer des informations d'un profil anonyme.

Site web avec sélecteur de thèmes

Le graphisme des pages est un élément important d'un site et doit faire appel à d'autres compétences que celles d'un développeur, c'est-à-dire à celles d'un designer ou d'un intégrateur. Ces métiers sont de plus en plus amenés à cohabiter dans le cas de la construction d'une application web. Pour que leurs rôles soient clairement définis, il est donc indispensable de séparer la présentation et le code à proprement parler.

La fonctionnalité de thèmes introduite avec le Framework 2.0 va vous aider à regrouper les fichiers spécifiques au graphisme dans un répertoire séparé et à faire hériter vos pages et vos contrôles d'un thème visuel.

Dans ce chapitre, vous apprendrez à connaître les notions associées à la fonctionnalité de thèmes ASP .NET, à créer vos propres thèmes et à donner une apparence par défaut à vos pages et à vos contrôles.

18.1 Mise en forme des contrôles serveurs

Avant d'expliquer les thèmes, il est nécessaire de présenter les différentes façons de mettre en forme un contrôle serveur. Certains moyens sont plus simples que d'autres a priori, mais au final, ils peuvent alourdir le code inutilement et devenir difficilement configurables. Nous allons vous présenter les différentes techniques, ce qui vous permettra de mieux comprendre comment cela fonctionne et de faire votre choix parmi elles.

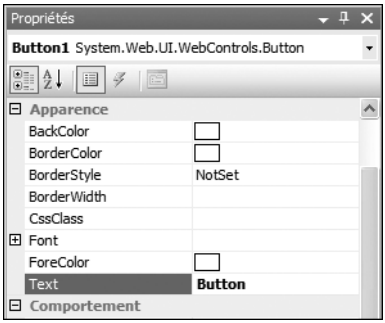
Propriétés de style

La méthode la plus simple pour personnaliser rapidement l'apparence de vos contrôles est de configurer les propriétés de style. Celles-ci sont un équivalent des attributs des balises HTML qui servent à représenter le contrôle.

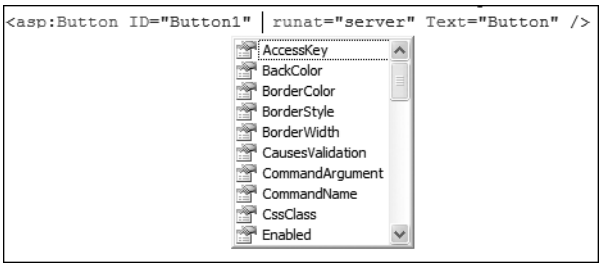
Édition de propriétés de contrôle en mode Source

En mode Source, la fonctionnalité d'Intellisense est accessible lorsque vous commencez à modifier la balise asp.net correspondant au contrôle serveur et propose toutes les propriétés disponibles par ordre alphabétique (voir Figure 18-1).

Lorsqu'il s'agit de propriétés booléennes ou d'énumération, l'éditeur propose en plus l'ensemble des valeurs possibles une fois saisi le caractère =. Vous n'aurez qu'à cliquer sur la valeur désirée pour l'affecter à la propriété, ce qui vous évitera notamment de faire une erreur de saisie (voir Figure 18-2).



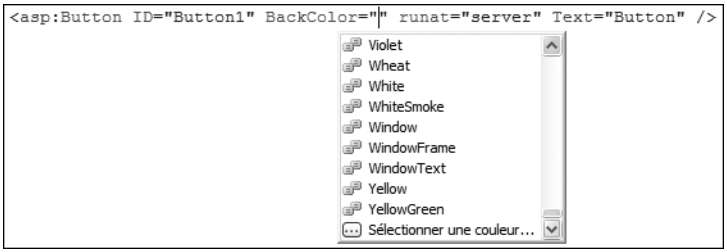
◄ **Figure 18-1 :** Liste des propriétés d'un contrôle d'une page aspx en mode Source grâce à l'Intellisense



▲ **Figure 18-2 :** Liste des valeurs possibles d'une propriété d'un contrôle

Édition de propriétés de contrôle en mode Design

En mode Design, l'éditeur de propriétés est encore plus simple. Il permet de trouver la propriété à personnaliser et d'en modifier la valeur.

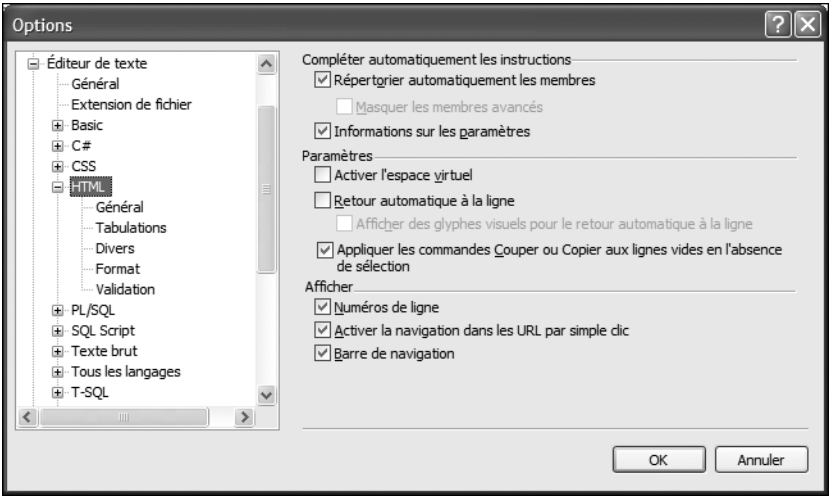


▲ **Figure 18-3 :** Fenêtre Propriétés d'un contrôle d'une page aspx en mode Design

Options de Visual Studio

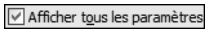
Visual Studio possède des options intéressantes relatives à l'éditeur HTML en mode Source, c'est-à-dire celui que vous utilisez pour les pages ou les contrôles utilisateurs aspx. Pour configurer ces options, ouvrez la boîte de dialogue

correspondante en sélectionnant la commande **Options** du menu **Outils**. Les options relatives aux différents éditeurs se trouvent dans le menu **Éditeur de texte**.



▲ **Figure 18-4 :** Options de Visual Studio relatives à l'éditeur HTML

Options de formatage



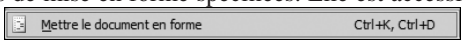
Si toutes les options ne sont pas affichées, cochez la case *Afficher toutes les options* en bas de la fenêtre. Elle permet d'afficher et de masquer les options les plus fréquemment utilisées.

La commande *Insérer des guillemets de valeur d'attributs lors de la saisie* du menu **Éditeur de texte/HTML/Format** active une fonctionnalité qui ajoute automatiquement des guillemets une fois saisi le caractère = à la suite d'une propriété de contrôle.

Cela peut paraître anodin, mais ce genre d'option d'aide à la saisie permet d'améliorer considérablement la productivité. Il est parfois plus rapide d'écrire une propriété que d'affecter une valeur à partir du mode Design.

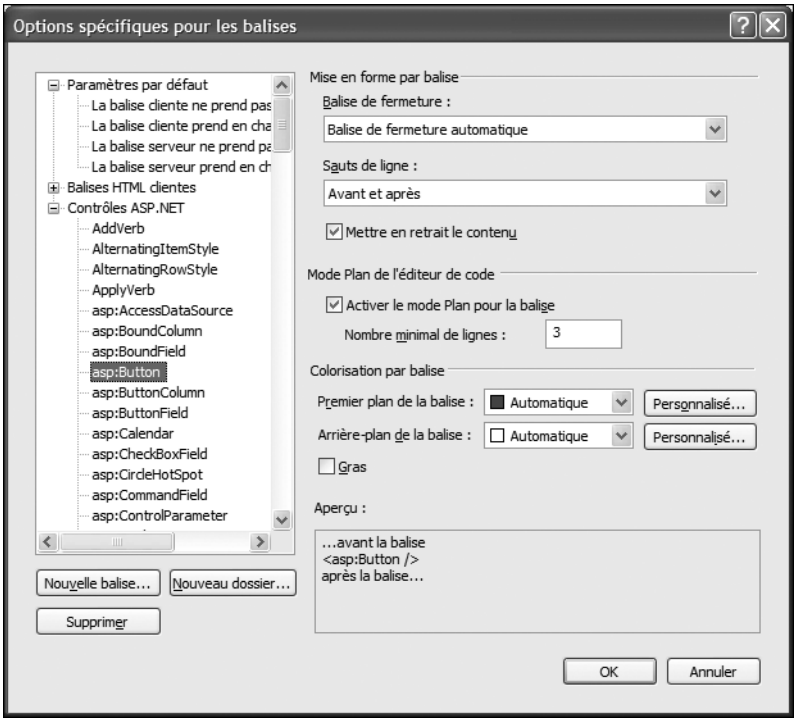
Options communes à tous les éditeurs

Le menu **Éditeur de texte/Basic** permet de spécifier des options communes à tous les éditeurs, comme l'affichage des numéros de ligne ou le retour automatique à la ligne.

La plupart des options de cette fenêtre servent notamment lors de la mise en forme du document. Cette fonctionnalité extrêmement pratique permet de réorganiser les contrôles de la page selon les règles de mise en forme spécifiées. Elle est accessible à partir du menu **Edition/Avacé** : 

Vous pouvez également y accéder à partir d'un bouton de la barre d'outils *Modification de la source HTML* : 

Si vous voulez aller plus loin dans la personnalisation des options de formatage ou comprendre quels choix de formatage ont été faits pour chaque contrôle ou balise HTML, cliquez sur le bouton **Options spécifiques pour les balises**. Il ouvre une fenêtre à partir de laquelle vous pourrez visualiser et configurer tous ces choix. Une arborescence permet de naviguer facilement dans la hiérarchie des contrôles et de découvrir pour chacun d'entre eux les options qui lui sont affectées. Un aperçu qui montre le rendu généré par les options sélectionnées est également affiché.



▲ **Figure 18-5 :** Fenêtre Options spécifiques pour les balises

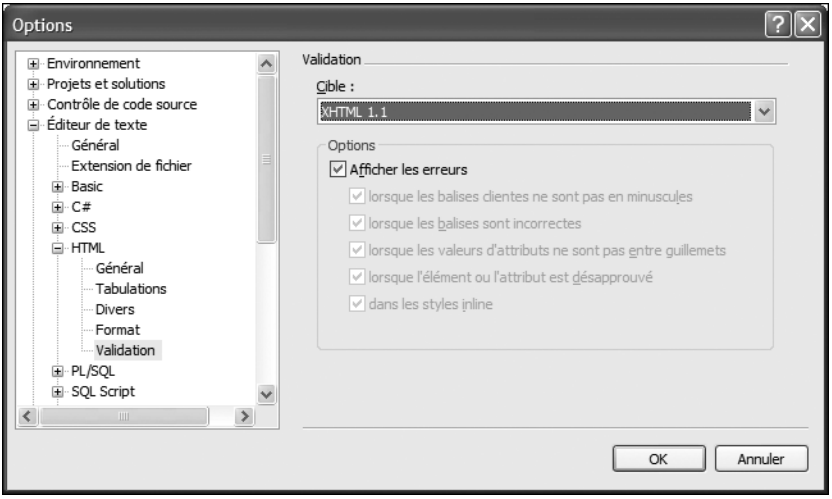
Selon les paramètres par défaut, les balises qui ne contiennent aucun contrôle enfant sont fermées automatiquement alors que les balises avec contenu ont une balise de fermeture distincte. Par ailleurs, des sauts de ligne sont effectués dans certains cas. Enfin, le mode Plan (bouton de développement et de réduction) est activé pour des contrôles qui sont déclarés sur trois lignes ou plus.

```
4    <asp:Button ID="Button1"
5    BackColor=""
6    runat="server" Text="Button" />
```

◄ **Figure 18-6 :** Activation du mode Plan pour les balises de plus de trois lignes

Options de validation

Le menu **Éditeur de texte/HTML/Validation** permet de sélectionner le schéma de validation que les pages doivent vérifier et respecter. Des erreurs seront affichées si votre code ne respecte pas ce schéma et les options de validation associées.



▲ **Figure 18-7 :** Fenêtre de sélection du schéma de validation

Astuce

Personnaliser un environnement de développement

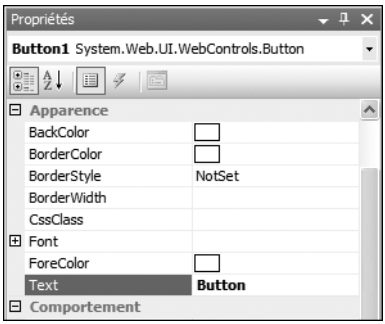
N'hésitez pas à parcourir l'ensemble des options de Visual Studio afin de configurer votre environnement de développement selon vos souhaits.

Mise en forme automatique

La plupart des contrôles serveurs évolués disposent d'un Assistant de mise en forme automatique. En mode Design, un smart tag permet d'ouvrir cet Assistant.

◄ **Figure 18-8 :** Smart tag de mise en forme automatique d'un contrôle

Une boîte de dialogue s'ouvre alors et propose plusieurs styles préconfigurés. La sélection d'un style permet de visualiser le rendu que vous obtiendrez.



◄ **Figure 18-9 :** Boîte de dialogue Mise en forme automatique

Les propriétés de style du contrôle associé sont affectées en accord avec l'apparence qui a été sélectionnée.

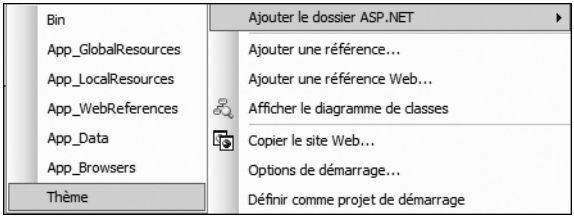
Un contrôle `Login` par exemple ressemblera à ce qui suit une fois sélectionnée l'apparence :

```
<asp:Login ID="Login1" runat="server" BorderColor="#E6E2D8"
    BorderPadding="4" BorderStyle="Solid" BorderWidth="1px"
    Font-Names="Verdana" Font-Size="0.8em"
    ForeColor="#333333">
    <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True"
        Font-Size="0.9em" ForeColor="White" />
    <InstructionTextStyle Font-Italic="True"
        ForeColor="Black" />
    <TextBoxStyle Font-Size="0.8em" />
    <LoginButtonStyle BackColor="#FFFBFF"
        BorderColor="#CCCCCC" BorderStyle="Solid"
        BorderWidth="1px" Font-Names="Verdana"
        Font-Size="0.8em" ForeColor="#284775" />
</asp:Login>
```

Cette technique n'est pas conseillée puisqu'elle est en désaccord avec notre volonté de séparer la présentation et le code et de diminuer le poids des pages grâce à l'utilisation de feuilles de styles CSS.

18.2 Thèmes

Un thème permet de définir l'apparence des pages et des contrôles puis de l'appliquer de manière cohérente sur les pages d'une application web. Les thèmes sont stockés dans un répertoire particulier du site nommé *App_Themes*. Chaque dossier situé à la racine constitue un thème. Pour ajouter ce répertoire, cliquez du bouton droit sur le site dans l'Explorateur de solutions puis sélectionnez **Thème** dans le menu **Ajouter le dossier ASP.NET**.



◄ **Figure 18-10 :** Ajout du dossier App_Themes

Un thème sert à paramétrer de manière globale, et dans des fichiers séparés, les propriétés des contrôles qui apparaissent dans les pages. Il peut contenir des feuilles de style CSS, des images et des fichiers d'apparence.

Fichiers d'apparence

Un fichier d'apparence a l'extension *.skin* et contient les paramètres de propriété des contrôles serveurs. Les paramètres d'apparence de contrôle sont semblables au balisage du contrôle lui-même, mais contiennent uniquement les propriétés que vous souhaitez définir dans le cadre du thème. Par exemple, l'apparence suivante représente l'apparence de contrôle du contrôle `Button` :

```
<asp:button runat="server" BackColor="lightblue"
 ForeColor="black" />
```

Vous créez des fichiers *.skin* dans le répertoire de votre thème. Un fichier *.skin* peut contenir une ou plusieurs skins de contrôle pour un ou plusieurs types de contrôles. Vous pouvez ainsi définir les skins dans un fichier séparé pour chaque contrôle ou définir toutes les skins d'un thème dans un seul fichier.

Vous êtes libre de définir soit une apparence par défaut, soit une apparence nommée pour un contrôle grâce à l'attribut `SkinId`.

Apparence par défaut et apparence nommée

Une apparence par défaut ne possède pas d'attribut `SkinID` et s'applique automatiquement à tous les contrôles du même type lorsqu'un thème est appliqué à une page.

Remarque

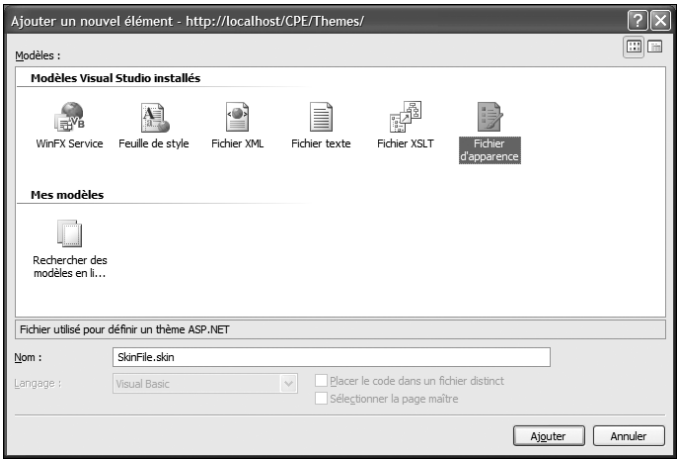
Aucun héritage d'apparence entre un contrôle parent et enfant

Les apparences par défaut sont mises en correspondance exacte par type de contrôle. Cela signifie qu'il n'y a aucun héritage d'apparence pour un contrôle qui dériverait d'un type de contrôle auquel on aurait affecté un thème. Cela évite par exemple qu'un contrôle `LinkButton` hérite de l'apparence d'un contrôle `Button`.

Une apparence nommée est une apparence de contrôle dotée d'un ensemble de propriétés `SkinID`. Les apparences nommées ne s'appliquent pas automatiquement aux contrôles. Vous devez affecter explicitement une apparence nommée à un contrôle en définissant la propriété `SkinID` du contrôle. La création d'apparences nommées permet ainsi de définir différentes apparences pour plusieurs instances d'un même type de contrôle.

Création d'un fichier d'apparence

La boîte de dialogue d'ajout de fichier dans un dossier de thèmes permet d'ajouter des fichiers CSS mais aussi des fichiers d'apparence *.skin*. Pour afficher cette boîte de dialogue, cliquez du bouton droit sur le dossier du thème et sélectionnez la commande **Ajouter un nouvel élément**.

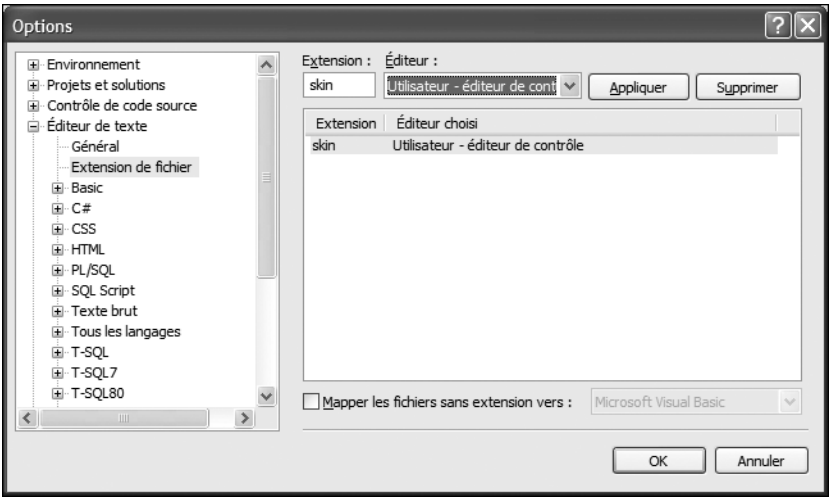


▲ **Figure 18-11 :** Ajout d'un fichier d'apparence (.skin) à un thème

Copiez ensuite le code précédent en enlevant toutes les propriétés non Themeable – les propriétés non liées à la mise en forme - , comme l'identifiant du contrôle, mais en gardant `runat="server"` qui est indispensable. Vous pouvez ensuite supprimer la mise en forme du contrôle grâce au premier style de l'Assistant de mise en forme nommé **Supprimer la mise en forme**. Il supprimera proprement l'ensemble des propriétés qu'il avait éventuellement ajoutées auparavant. Tous les contrôles du même type contenus dans une page avec ce thème ressembleront ainsi au modèle que vous venez de spécifier.

Éditeur de fichiers d'apparence

Par défaut, aucun éditeur n'est associé aux fichiers d'apparence. Cela a notamment pour conséquence que l'Intellisense n'est pas disponible et que les erreurs d'orthographe sur les noms de propriété ou les valeurs ne sont pas relevées. Heureusement, il est possible d'associer n'importe quelle extension à un éditeur spécifique. Vous pouvez ainsi utiliser l'éditeur de contrôle utilisateur qui vous apportera toutes les fonctionnalités qui vous manquaient. Pour cela, ouvrez la fenêtre des options puis sélectionnez le menu **Éditeur de texte/ Extension de fichier**.



▲ **Figure 18-12 :** Ajout d'un éditeur aux fichiers d'extension .skin

Vous pouvez alors ajouter l'extension *.skin* et l'associer à l'éditeur de contrôle utilisateur.

Feuilles de style en cascade

Un thème peut également contenir des feuilles de style en cascade (fichier *.css*). Lorsque vous placez un fichier *.css* dans le répertoire des thèmes, la feuille de style est ajoutée automatiquement à la balise `Head` d'une page associée à ce thème.

Différences entre les thèmes et les feuilles de style

Les thèmes sont semblables aux feuilles de style CSS car tous deux servent à configurer l'apparence des éléments des pages. Cependant, les thèmes ont la particularité de pouvoir définir plusieurs propriétés d'un contrôle ou d'une page, et pas seulement les propriétés de style.

Astuce

Utilisation de la propriété CssClass

En affectant la propriété `CssClass` des contrôles, vous pouvez facilement définir une apparence globale tout en gardant des pages conformes aux standards CSS.

Association d'images aux propriétés

Les thèmes peuvent également contenir des images. Certaines propriétés de contrôle sont en effet reliées au chemin d'accès vers une image, comme les boutons d'un TreeView, le séparateur d'un SiteMapPath ou les boutons d'un Login.

De manière générale, les fichiers de ressources du thème se trouvent dans le même dossier que les fichiers d'apparence de ce même thème, mais elles peuvent aussi être stockées à un autre emplacement de l'application web, comme un sous-dossier du répertoire des thèmes. Pour faire référence à un fichier de ressources d'un sous-dossier du répertoire des thèmes, utilisez un chemin d'accès similaire à celui représenté pour la définition de l'apparence de ce contrôle `Image` :

```
<asp:Image runat="server" ImageUrl="Theme/img.gif" />
```

Vous pouvez aussi stocker vos fichiers de ressources en dehors du répertoire des thèmes. Si vous utilisez le caractère tilde (~) pour faire référence aux fichiers de ressources, vous pouvez spécifier un chemin d'accès relatif par rapport à la racine :

```
<asp:Image runat="server" ImageUrl="~/Images/img.gif" />
```

Propriétés définies à l'aide de thèmes

Vous ne pouvez définir que les propriétés ayant un attribut `ThemeableAttribute` doté de la valeur `true` dans la classe du contrôle.

Les propriétés qui spécifient explicitement le comportement du contrôle plutôt que l'apparence n'accepte pas de valeurs de thème. Par exemple, vous ne pouvez pas utiliser un thème pour définir la propriété `CommandName` d'un contrôle `Button` ou la propriété `AllowPaging` d'un contrôle `GridView`.

Si vous le faites par curiosité ou par inadvertance, un message d'erreur explicite vous le rappellera.

18.3 Appliquer un thème

Priorité des paramètres de thème

Si vous définissez la propriété `Theme` d'une page, les paramètres de contrôle du thème et ceux de la page sont fusionnés pour former les paramètres du contrôle. Si un paramètre de contrôle est défini à la fois dans le contrôle et dans le thème, les paramètres de contrôle du thème ont la priorité et remplacent donc tous les paramètres du contrôle éventuellement spécifiés.

Cependant, vous pouvez appliquer un thème sous forme de thème de feuille de style en définissant la propriété `StyleSheetTheme` de la page. Dans ce cas, les paramètres de page locaux sont prioritaires sur ceux définis dans le thème lorsque le paramètre est défini aux deux emplacements. Il s'agit du modèle utilisé par les feuilles de style en cascade.

Vous pouvez ainsi combiner les deux en appliquant un thème sous forme de thème de feuille de style (`StyleSheetTheme`) pour pouvoir définir les propriétés de certains contrôles de la page tout en appliquant un thème (`Theme`) pour garder une apparence globale.

Déclarer un thème

Vous pouvez déclarer un thème à utiliser dans une page grâce à l'attribut `Theme` ou `StyleSheetTheme` de la directive de page. Par exemple :

```
<%@Page Theme="DefaultTheme"
    StyleSheetTheme="DefaultStyleSheetTheme" %>
```

Vous pouvez également utiliser la section <pages> du fichier *web.config* à la racine du site pour affecter un thème par défaut à toutes les pages :

```
<pages theme="Default" />
```

Affecter un thème par programmation

La propriété `Theme` est accessible pour chaque page. Elle permet d'affecter par programmation le thème de la page. Le gestionnaire d'événements le plus approprié pour affecter le thème est l'événement `PreRender` de la page.

Thèmes globaux

Un thème global peut être appliqué à tous les sites web d'un serveur de sorte à définir une apparence globale.

Les thèmes globaux sont identiques aux thèmes de page puisqu'ils incluent les mêmes types de fichiers, c'est-à-dire les fichiers d'apparence, les feuilles de style et les images. La seule différence est que thèmes globaux sont stockés dans un répertoire situé sous le répertoire *C:\Inetpub\wwwroot\aspnet_client\system_web\2_0_50727\Themes* spécialement prévu à cet effet.

Vous pouvez ensuite utiliser un tel thème comme s'il était présent dans votre site et en faire le thème par défaut.

Astuce

Répertoire des thèmes globaux

Si vous souhaitez vraiment que tous les sites d'un même serveur aient la même apparence, vous pouvez définir la propriété `theme` de l'élément `<pages>` du fichier *machine.config* du répertoire *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727* afin que le thème s'applique à toutes les pages des sites.

18.4 Stocker un thème par utilisateur

Vous allez réaliser un site dans lequel l'utilisateur pourra choisir lui-même parmi différents thèmes et ainsi modifier l'apparence des pages. Cela peut se faire facilement. Il suffit en effet de créer plusieurs thèmes dans le répertoire *App_Themes* et de les mettre à disposition de l'utilisateur pour qu'il puisse sélectionner celui qui lui convient.

Il faut ensuite stocker le thème de l'utilisateur et l'affecter par programmation à chaque nouvel appel de page. Le mode de stockage le plus approprié est le cookie, qui a la particularité d'être simple d'utilisation, persistant dans le temps et associé au navigateur de la machine cliente.

Il est conseillé au préalable d'affecter un thème par défaut au cas où l'utilisateur n'aurait pas encore sélectionné de thème.

Listage des thèmes

Vous pouvez mettre à disposition les différents thèmes d'un site à l'aide d'un contrôle `DropDownList`. Cette `DropDownList` sera placée dans une page maître que vous pourrez utiliser pour créer toutes vos autres pages. Vous ne définissez ainsi le code correspondant qu'une seule fois.

Pour récupérer les thèmes disponibles et initialiser la page avec le thème stocké dans un cookie, vous pouvez avoir recours au code suivant :

```
Protected Sub Page_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
        If Not Page.IsPostBack Then
            Dim vpp As _
                System.Web.Hosting.VirtualPathProvider = _
                Hosting.HostingEnvironment. _
                VirtualPathProvider
            ddlThemes.DataSource = vpp.GetDirectory _
                ("~/App_Themes").Directories
            ddlThemes.DataTextField = "Name"
            ddlThemes.DataBind()
            ddlThemes.SelectedValue = _
                My.Request.Cookies("Settings")("Theme")
        End If
End Sub
```

La liste est ainsi remplie avec les thèmes disponibles dans le répertoire de thèmes.

◄ **Figure 18-13 :** Liste des thèmes

Astuce

Ajout d'un thème global à la liste

Si vous voulez placer dans cette liste un thème global, vous pouvez ajouter un `ListItem` à la `DropDownList`, avec comme valeur le nom du thème, et activer sa propriété `AppendDataBoundItems`, ce qui introduira cet élément au début de la liste.

Stockage des thèmes et affectation du thème sélectionné

Vous pouvez activer la propriété `AutoPostBack` de la `DropDownList` afin qu'elle déclenche l'événement `SelectedIndexChanged` au moment de la sélection d'un thème. Vous réagissez ainsi à cet événement en stockant le choix de l'utilisateur :

```
Protected Sub ddlThemes_SelectedIndexChanged( _
    ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles ddlThemes.SelectedIndexChanged
        My.Response.Cookies("Settings")("Theme") = _
            ddlThemes.SelectedValue
        My.Response.Cookies("Settings").Expires = _
            Date.Now.AddDays(1)
        Response.Redirect(Request.RawUrl)
End Sub
```

Initialisation du thème à l'appel d'une page

Vous devez enfin initialiser chaque page avec le thème sélectionné par l'utilisateur. Le moyen le plus simple est d'utiliser une classe qui hérite de `System.Web.UI.Page`, d'y insérer le code d'initialisation et d'en faire hériter toutes vos pages. Comme vous l'avez vu précédemment, l'initialisation du thème se fait au moment d'un `PreRender` :

```
Public Class BasePage
    Inherits System.Web.UI.Page

    Protected Sub Page_PreInit(ByVal sender As Object, _
        ByVal e As _ System.EventArgs) Handles Me.PreInit
        If My.Request.Cookies("Settings") Is Nothing Then
            Dim c As HttpCookie = _
                New HttpCookie("Settings")
            c("Theme") = "Blue"
            c.Expires = Date.Now.AddDays(1)
            My.Response.Cookies.Add(c)
        End If
        Me.Theme = My.Request.Cookies("Settings")("Theme")
    End Sub

End Class
```

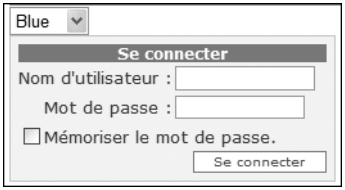
Pour que toutes vos pages héritent de cette classe de base, utilisez l'astuce suivante. L'élément page du fichier *web.config* permet de déclarer un attribut qui correspond au nom de la classe dont toutes les autres doivent hériter.

```
<pages pageBaseType="BasePage"></pages>
```

Il y a cependant une contrainte importante puisque cet attribut ne fonctionne malheureusement que pour des pages qui déclarent leur code à l'intérieur même de la page *.aspx*, et non pour les pages avec un ficher *.aspx.vb* associé (Code Behind).

18.5 Résultat final

Après avoir réalisé toutes ces étapes, vous avez à disposition un site web sur lequel l'utilisateur peut choisir un thème à sa convenance parmi ceux proposés. Si vous avez correctement spécifié les styles de chaque contrôle que vous utilisez dans vos pages, vous pouvez sélectionner les différents thèmes et voir le résultat.



◄ **Figure 18-14 :** Contrôle Login avec le thème Blue

◄ **Figure 18-15 :** Contrôle Login avec le thème Red

18.6 Check-list

Les thèmes ASP .NET sont une nouveauté extrêmement utile s'ils sont utilisés à bon escient. Pour créer un thème et l'utiliser de la façon la plus optimale dans un site, mieux vaut recourir aux méthodes conseillées dans ce chapitre. En outre, il faut respecter quelques règles lors de la création de vos pages, par exemple n'affecter aucun style à vos contrôles et laisser toute personnalisation de l'affichage à la charge du thème.

1. Création du thème :
   - ajout du répertoire *App_Themes* ;
   - ajout du thème dont le nom de répertoire correspond au nom du thème.
2. Déclaration du thème par défaut dans le *web.config* :

```
<pages theme="Default" />
```

3. Ajout de fichiers *.skin* :
   - déclaration des contrôles avec leurs propriétés `CssClass`.

4 Ajout de feuilles CSS :

   – utilisation des classes CSS pour affecter le style des contrôles.

5 Si vous voulez affecter le même thème à un autre site :

   – ajout du thème dans le répertoire des thèmes globaux : *C:\Inetpub\wwwroot\aspnet_client\system_web\2_0_50727\Themes*.

6 Si vous voulez proposer plusieurs thèmes à l'utilisateur :

   – création des thèmes

   – listage des thèmes, à savoir création d'une Master Page contenant la liste des thèmes, puis initialisation de la liste :

```
    ddlThemes.DataSource =
        vpp.GetDirectory("~/App_Themes").Directories
```

   – choix du mécanisme de stockage par utilisateur (cookie, session, cache, profil…).

   – création d'une page de base :

   affectation du thème par défaut à la sélection :

```
My.Response.Cookies("Settings")("Theme") = _
    ddlThemes.SelectedValue
```

   initialisation du thème à l'appel d'une page :

```
Me.Theme = My.Request.Cookies("Settings")("Theme")
```

   héritage des toutes les pages à la page de base :

```
    <pages pageBaseType="BasePage"></pages>
```

   ou

```
Partial Class _Default
    Inherits Base.Page
```

Représentation de données hiérarchiques à l'aide d'un contrôle TreeView

L'affichage de données hiérarchiques est un besoin qui apparaît souvent lors du développement d'une application. Dans ce chapitre, vous allez réaliser différentes fonctionnalités à l'aide de contrôles `TreeView` qui vous permettront de mieux apprécier tout leur potentiel.

19.1 Contrôle TreeView

Dans le Framework 1.1, il n'existait aucun contrôle natif permettant de réaliser ce genre de fonctionnalité facilement. Certains contrôles serveurs personnalisés ont donc été développés pour pallier ce manque, mais aucun ne répond à toutes les problématiques qu'un développeur peut rencontrer.

Le Framework 2.0 propose un nouveau contrôle évolué, `TreeView`, qui tire partie des améliorations apportées à la création de contrôles, comme la prise en charge du design, des ressources embarquées, des sources de données ou encore des fonctionnalités côté client.

Les fonctionnalités suivantes sont notamment prises en charge :

- liaison à une source de données XML ;
- navigation à l'intérieur d'un site lorsqu'il est associé à un contrôle `SiteMapDataSource` ;
- affichage des nœuds en tant que lien ou en tant que texte ;
- accès par programme au modèle objet du `TreeView` pour agir dynamiquement sur les nœuds et leurs propriétés ;
- remplissage de nœud côté client (sans rechargement) ;
- affichage d'une case à cocher devant chaque nœud ;
- apparence personnalisable grâce aux thèmes, aux images définies par l'utilisateur et aux styles.

Avant d'expliquer comment ajouter des nœuds à un arbre et comment les gérer, passons en revue les notions qui leur sont associées.

Notions relatives aux nœuds

Chaque entrée de l'arborescence est appelée "nœud". Il existe plusieurs types de nœuds qui, selon leur emplacement dans la hiérarchie, sont appelés différemment :

- Un nœud parent (`ParentNode`) contient d'autres nœuds.
- Un nœud enfant (`ChildNode`) est contenu dans un autre nœud.
- Un nœud feuille (`LeafNode`) n'a pas de nœud enfant.

- Le nœud racine (`RootNode`) n'est contenu dans aucun autre nœud. C'est donc l'ancêtre de tous les autres nœuds.

Il est important de connaître ces différents types de nœuds car plusieurs propriétés visuelles et comportementales sont déterminées par le type d'un nœud.

Classe TreeNode

La classe correspondant à la notion de nœud est `TreeNode`. Voici un aperçu de ses propriétés :

Propriétés d'un objet TreeNode

Propriété	Description
`Checked`	Obtient ou définit une valeur qui indique si la case à cocher du nœud est activée.
`ChildNodes`	Obtient une collection `TreeNodeCollection` qui contient les nœuds enfants de premier niveau du nœud courant.
`DataBound`	Obtient une valeur qui indique si le nœud a été créé via une liaison de données.
`DataItem`	Obtient l'élément de données qui est lié au contrôle.
`DataPath`	Obtient le chemin d'accès aux données liées au nœud.
`Depth`	Obtient la profondeur du nœud.
`Expanded`	Obtient ou définit une valeur qui indique si le nœud est développé.
`ImageToolTip`	Obtient ou définit le texte d'info-bulle pour l'image affichée en regard d'un nœud.
`ImageUrl`	Obtient ou définit l'URL vers une image qui est affichée à côté du nœud.
`NavigateUrl`	Obtient ou définit l'URL à laquelle accéder lorsqu'un clic est effectué sur le nœud.
`Parent`	Obtient le nœud parent du nœud courant.
`PopulateOnDemand`	Obtient ou définit une valeur qui indique si le nœud est rempli dynamiquement.
`SelectAction`	Obtient ou définit l'événement ou les événements à déclencher lorsqu'un nœud est sélectionné.
`Selected`	Obtient ou définit une valeur qui indique si le nœud est sélectionné.
`ShowCheckBox`	Obtient ou définit une valeur qui indique si une case à cocher est affichée en regard du nœud.

Propriétés d'un objet TreeNode	
Propriété	**Description**
`Target`	Obtient ou définit la fenêtre (ou frame cible) dans laquelle afficher le contenu de la page web associée à un nœud.
`Text`	Obtient ou définit le texte affiché pour le nœud.
`ToolTip`	Obtient ou définit le texte d'info-bulle pour le nœud.
`Value`	Obtient ou définit une valeur non affichée utilisée pour stocker des données supplémentaires relatives au nœud, telles que les données utilisées pour la gestion des événements de publication.
`ValuePath`	Obtient le chemin d'accès du nœud racine vers le nœud courant.

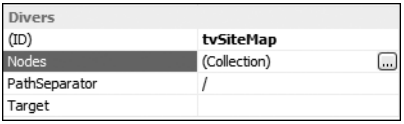
19.2 Ajouter des nœuds à un contrôle TreeView

Le moyen le plus simple pour ajouter des nœuds à un contrôle `TreeView` est de les spécifier à l'aide d'une syntaxe déclarative. Pour cela, il suffit d'ajouter des balises `<asp:TreeNode>` à la collection `<Nodes>`.

1 TreeView Tout d'abord, déposez un contrôle `TreeView` sur votre page. La balise par défaut ressemble à celle-ci :

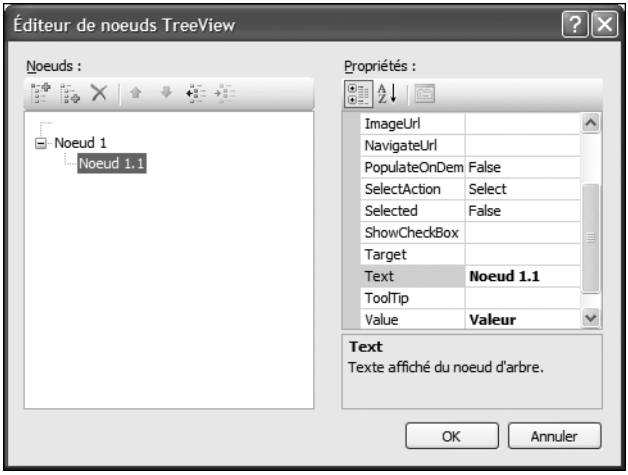
```
<asp:TreeView ID="TreeView1" runat="server"></asp:TreeView>
```

2 En mode Design, une fenêtre de création de nœuds statiques est disponible pour vous aider à gérer la hiérarchie que vous voulez mettre en place. Pour l'ouvrir, cliquez sur le bouton situé à droite de la propriété `Nodes` dans l'Éditeur de propriétés. Cette fenêtre permet visualiser la hiérarchie des nœuds, de les créer ou de les supprimer, et de modifier leurs propriétés.



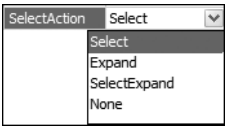
◀ **Figure 19-1 :** Propriété Nodes dans l'Éditeur de propriétés

3 Ajoutez ensuite à la collection de nœuds `<Nodes>` les balises `<asp:TreeNode>` que vous désirez.



▲ **Figure 19-2 :** Fenêtre Éditeur de nœuds TreeView

4 Vous pouvez notamment modifier la propriété `SelectAction`, qui vous permet de spécifier l'action à effectuer au moment de la sélection du nœud.



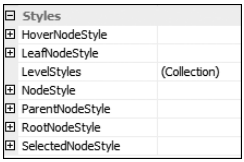
◄ **Figure 19-3 :** Propriété SelectAction d'un nœud

Modes possibles pour un nœud

Un nœud peut se présenter sous l'un des deux modes suivants : mode Sélection et mode Navigation. Par défaut, un nœud est en mode de sélection. Pour faire passer un nœud en mode de navigation, affectez une valeur différente d'une chaîne vide (`""`) à la propriété `NavigateUrl` du nœud. Pour faire passer un nœud en mode de sélection, affectez une chaîne vide (`""`) à la propriété `NavigateUrl` du nœud.

19.3 Modifier des styles associés aux différents types de nœuds

Plusieurs styles sont configurables pour un contrôle `TreeView`. Certains correspondent au style affecté à un type de feuille (racine, parent, feuille).

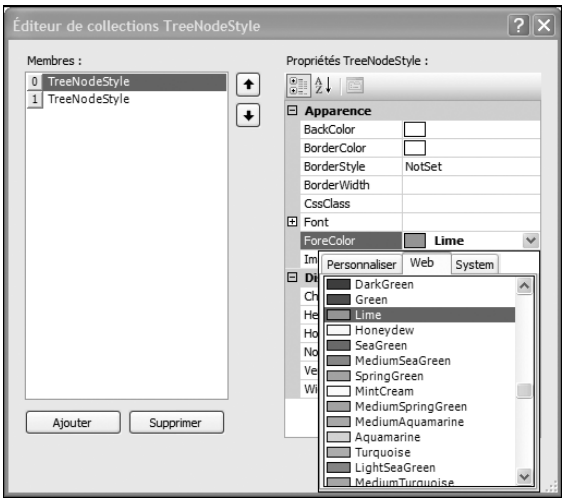


◄ **Figure 19-4 :** Liste des propriétés de style d'un contrôle TreeView

Vous pouvez également agir sur les styles affectés au moment du survol (`HoverNodeStyle`) et de la sélection (`SelectedNodeStyle`). Il est ainsi facile de spécifier que le nœud sélectionné aura un arrière-plan jaune et que son texte sera en gras et en italique.

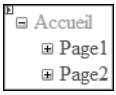
◄ **Figure 19-5 :** Propriétés de style de la propriété SelectedNodeStyle

La propriété `LevelStyles` est une collection de styles qui permet de spécifier le style des nœuds selon le niveau de hiérarchie dans lequel ils se trouvent. Le bouton situé à droite de celle-ci dans l'Éditeur de propriétés ouvre une fenêtre dans laquelle vous pouvez ajouter les niveaux et configurer leurs styles respectifs. Vous pouvez par exemple affecter une couleur de texte verte aux éléments racines et une couleur rouge aux éléments enfants.



▲ **Figure 19-6 :** Fenêtre Éditeur de collections TreeNodeStyle

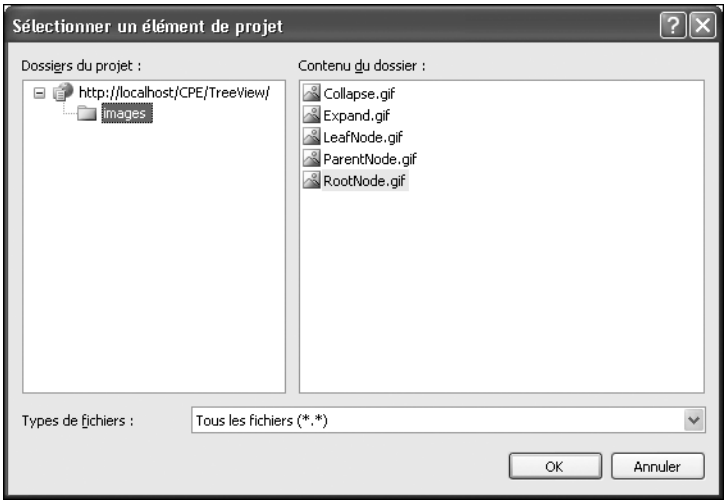
L'apparence du contrôle dans l'Éditeur en mode Design est mise à jour pour refléter les changements de style qui vous avez effectué.



◄ **Figure 19-7 :** Contrôle TreeView en mode Design

19.4 Affecter des images aux nœuds

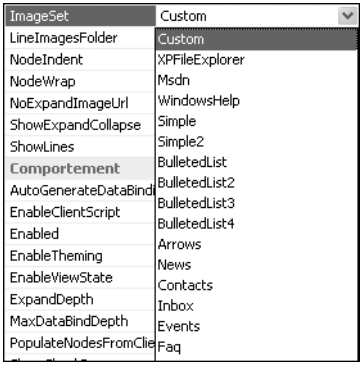
Vous avez la possibilité d'affecter des images au début de chaque nœud. La propriété `ImageUrl` d'un `TreeNode` permet de sélectionner l'emplacement de l'image à l'aide d'une fenêtre qui liste les images disponibles.



▲ **Figure 19-8 :** Fenêtre Sélectionner un élément de projet

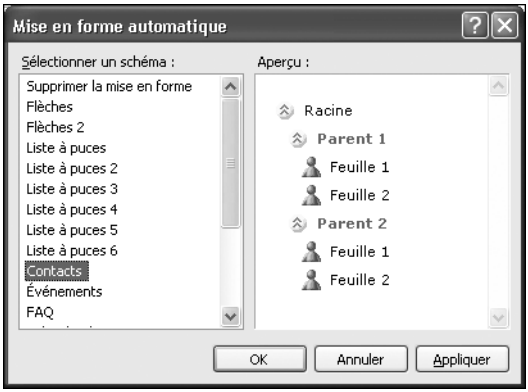
Vous pouvez également spécifier les images de réduction et de développement à l'aide des propriétés `ExpandImageUrl` et `CollapseImageUrl`, ainsi qu'un répertoire (`LineImagesFolder`) contenant les images des lignes.

Enfin, pour associer simplement des images à vos nœuds, vous avez à votre disposition une liste d'images prédéfinies grâce à la propriété `ImageSet`. Vous pouvez ainsi donner à votre arborescence une apparence qui la fait ressembler à un Explorateur Windows ou à une liste de contacts.



◄ **Figure 19-9 :** Liste des images prédéfinies

Cette liste est aussi accessible à partir du lien *Mise en forme automatique* de la balise active et permet en plus d'avoir un aperçu du résultat.



◄ **Figure 19-10 :** Assistant Mise en forme automatique

Remarque

Ressources embarquées

Ces images sont stockées dans l'assembly *System.Web* en tant que ressources embarquées.

Renvoi ***Pour plus d'informations sur les ressources embarquées au sein d'assembly, reportez-vous au chapitre Création d'un contrôle serveur personnalisé.***

19.5 Lier des données à un contrôle TreeView

Le contrôle `TreeView` peut également être lié aux données, soit de manière déclarative (avec une source de données appropriée), soit par programmation (à un objet `XmlDocument` ou à un objet `DataSet` avec des relations), grâce à des mécanismes de liaison de données avancés.

Balises DataBindings et TreeNodeBinding

La collection `DataBindings` contient des objets `TreeNodeBinding` qui définissent la relation entre un élément de données et le nœud auquel il est lié. Vous pouvez spécifier les critères de liaison et la propriété de l'élément de données affichée dans le nœud à l'aide d'un élément `TreeNodeBinding`.

Le tableau suivant présente l'ensemble des propriétés d'un objet `TreeNodeBinding`. Plusieurs d'entre elles sont similaires à celles de l'objet `TreeNode`. Des propriétés supplémentaires permettent de relier ces propriétés à un champ de la source de données.

Propriétés d'un l'élément TreeNodeBinding	
`DataMember`	Obtient ou définit la valeur à associer à une propriété `IHierarchyData.Type` pour un élément de données afin de déterminer si la liaison de nœud d'arbre doit être appliquée.
`Depth`	Obtient ou définit la profondeur de nœud.
`FormatString`	Obtient ou définit la chaîne qui spécifie le format d'affichage du texte d'un nœud.
`ImageToolTip`	Obtient ou définit le texte d'info-bulle de l'image affiché à côté d'un nœud.
`ImageToolTipField`	Obtient ou définit le nom du champ de la source de données à lier à la propriété `ImageToolTip`.
`ImageUrl`	Obtient ou définit l'URL vers l'image affichée à côté d'un nœud.
`ImageUrlField`	Obtient ou définit le nom du champ de la source de données à lier à la propriété `ImageUrl`.
`NavigateUrl`	Obtient ou définit l'URL vers laquelle effectuer le lien lorsque l'utilisateur clique sur un nœud.
`NavigateUrlField`	Obtient ou définit le nom du champ de la source de données à lier à la propriété `NavigateUrl`.
`PopulateOnDemand`	Obtient ou définit une valeur indiquant si le nœud est rempli dynamiquement.
`SelectAction`	Obtient ou définit le ou les événements à déclencher lorsqu'un nœud est sélectionné. Les valeurs possibles sont `None`, `Expand`, `Select` et `SelectExpand`.

Propriétés d'un l'élément TreeNodeBinding	
DataMember	Obtient ou définit la valeur à associer à une propriété IHierarchyData.Type pour un élément de données afin de déterminer si la liaison de nœud d'arbre doit être appliquée.
ShowCheckBox	Obtient ou définit une valeur indiquant si une case à cocher est affichée à côté d'un nœud.
Target	Obtient ou définit la cible du lien d'un nœud.
TargetField	Obtient ou définit le nom du champ de la source de données à lier à la propriété Target.
Text	Obtient ou définit le texte affiché pour le nœud.
TextField	Obtient ou définit le nom du champ de la source de données à lier à la propriété Text.
ToolTip	Obtient ou définit le texte d'info-bulle d'un nœud.
ToolTipField	Obtient ou définit le nom du champ de la source de données à lier à la propriété ToolTip.
Value	Obtient ou définit une valeur qui n'est pas affichée mais utilisée pour stocker des données supplémentaires concernant un nœud, telles que les données utilisées pour gérer des événements de publication.
ValueField	Obtient ou définit le nom du champ de la source de données à lier à la propriété Value.

19.6 Remplir dynamiquement des nœuds

Lorsque la source de données est trop conséquente, que son traitement de récupération est trop pénalisant ou encore lorsque les données à afficher dépendent des informations que vous obtenez au moment de l'exécution, vous ne pouvez pas définir statiquement l'arborescence du TreeView.

Vous pouvez alors utiliser la fonctionnalité de remplissage dynamique de nœuds. En guise d'exemple, vous allez créer une arborescence qui récupère dynamiquement les clients puis les commandes à partir de la base de données SQL Northwind.

Tout d'abord, déposez un contrôle TreeView dans votre page et ajoutez un nœud racine comme vous l'avez vu précédemment.

```
<asp:TreeView ID="tvCustomers" runat="server">
        <Nodes>
        …
        </Nodes>
</asp:TreeView>
```

Affectez la valeur true à la propriété PopulateOnDemand d'un nœud, ce qui signifie que vous allez spécifier vous-même comment remplir le nœud à l'aide de code serveur.

```
<asp:TreeNode Text="Clients" Value="Customers" PopulateOnDemand="True"
➥ Expanded="False" />
```

Pour remplir un nœud dynamiquement, vous devez définir une méthode de gestion pour l'événement TreeNodePopulate. En fonction du niveau auquel le nœud a été développé, vous pouvez spécifiez des méthodes de remplissage différentes.

```
Protected Sub tvCustomers_TreeNodePopulate( _
ByVal sender As Object, ByVal e As TreeNodeEventArgs) _
 handles tvCustomers.TreeNodePopulate
        Select Case e.Node.Depth
            Case 0
                FillCustomers(e.Node)
            Case 1
                FillOrders(e.Node)
        End Select
    End Sub
```

La méthode GetDataSet permet de récupérer un DataSet à partir d'une requête.

```
Private Function GetDataSet(ByVal sql As String) As DataSet
Dim connstr As String =   _ "Data Source=(local);Initial
➥ Catalog=Northwind;User ID=sa"
Dim da As SqlDataAdapter = New SqlDataAdapter(sql, connstr)
Dim ds As DataSet = New DataSet()
da.Fill(ds)
Return ds
End Function
```

La méthode FillCustomers de remplissage des clients prend en paramètre le nœud auquel elle doit ajouter les nœuds enfants.

```
Private Sub FillCustomers(ByVal parent As TreeNode)
Dim ds As DataSet = GetDataSet( _ "SELECT customerId, companyName FROM
➥ customers order by companyName")
For Each row As DataRow In ds.Tables(0).Rows
            Dim node As TreeNode = New TreeNode()
            node.Text = row("companyname")
            node.Value = row("customerid")
            node.PopulateOnDemand = True
node.SelectAction = TreeNodeSelectAction.SelectExpand
            parent.ChildNodes.Add(node)
Next
End Sub
```

La méthode `FillOrders` de remplissage des factures d'un client fait appel à la valeur du nœud parent pour effectuer la requête.

```
Private Sub FillOrders(ByVal parent As TreeNode)
Dim ds As DataSet = GetDataSet( _
"SELECT customerId, orderId FROM orders WHERE customerId='" & parent.Value &
➥ "'")
For Each row As DataRow In ds.Tables(0).Rows
Dim node As TreeNode = New TreeNode()
node.Text = row("orderid")
node.Value = row("orderid")
node.PopulateOnDemand = False
node.SelectAction = TreeNodeSelectAction.SelectExpand
parent.ChildNodes.Add(node)
Next
End Sub
```

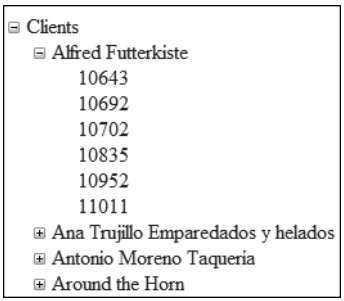
Enfin, le remplissage des nœuds côté client permet au contrôle `TreeView` de remplir un nœud à l'aide de script client lorsque les utilisateurs développent le nœud, sans nécessiter d'allers-retours avec le serveur. Pour cela, il suffit d'affecter la valeur `true` à la propriété `PopulateNodesFromClient`.

Remarque

Fonctionnalité disponible uniquement pour les navigateurs avancés

Seuls les navigateurs évolués peuvent utiliser cette fonctionnalité de remplissage de nœuds côté client.

Vous obtenez ainsi un contrôle `TreeView` dont les nœuds ne sont chargés que lorsque l'on clique dessus, ce qui améliore considérablement les performances, et tout cela sans rechargement de la page.



◄ **Figure 19-11** : Remplissage dynamique des clients puis de leurs factures

Source de données hiérarchiques

Le fait d'avoir un contrôle `TreeView` est une chose, encore faut-il des sources de données adéquates qui puissent représenter une hiérarchie. Pour cela, la source de données doit implémenter l'interface `IHierarchicalDataSource`.

Voici une présentation des deux sources de données hiérarchiques principales incluses dans le Framework ASP .NET.

XmlDataSource

Le contrôle `XmlDataSource` utilise un fichier XML, ce qui permet d'obtenir une source de données hiérarchiques que vous pouvez lier à des contrôles tels que `TreeView` ou `Menu`. Il prend notamment en charge des fonctions de filtrage à l'aide d'expressions XPath. Enfin, il permet de mettre à jour des données en enregistrant le document XML entier avec ses modifications.

SiteMapDataSource

Le contrôle `SiteMapDataSource` est une source de données spécialement adaptée aux plans de site hiérarchiques et aux contrôles de navigation, comme `TreeView` mais aussi comme `Menu` ou `SiteMapPath`.

Voici comment associer un `TreeView` à un plan de site.

Tout d'abord, créez un fichier *web.sitemap* à la racine du site dans lequel vous définirez la hiérarchie du plan de site.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns=
"http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
<siteMapNode url="default.aspx" title="Accueil"
description="">
<siteMapNode url="page1.aspx" title="Page1"
description="">
<siteMapNode url="page11.aspx" title="Page1.1"
description="" />
<siteMapNode url="page12.aspx" title="Page1.2"
description="" />
</siteMapNode>
<siteMapNode url="page2.aspx" title="Page2"
description="">
<siteMapNode url="page21.aspx" title="Page2.1"
description="" />
<siteMapNode url="page22.aspx" title="Page2.2"
description="" />
</siteMapNode>
</siteMapNode>
</siteMap>
```

Déposez ensuite un contrôle `TreeView` :

```
<asp:TreeView ID="tvSiteMap" runat="server"></asp:TreeView>
```

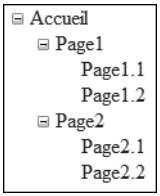
Puis déposez un contrôle `SiteMapDataSource` :

```
<asp:SiteMapDataSource ID="SiteMapDataSource1"
 runat="server" />
```

Enfin, affectez à la propriété `DataSourceID` du `TreeView` la valeur de l'identifiant du contrôle `SiteMapDataSource` :

```
DataSourceID="SiteMapDataSource1
```

Votre page affiche alors une arborescence représentant le plan de site.

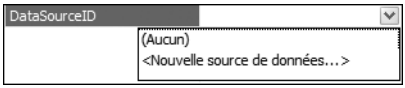


◄ **Figure 19-12 :** Arborescence de plan de site

Le contrôle `SiteMapDataSource` se lie aux données de plan de site et affiche les nœuds en fonction d'un nœud de démarrage spécifié dans la hiérarchie de plan de site. Par défaut, ce nœud correspond au nœud racine de la hiérarchie, mais il peut également s'agir d'un autre nœud de la hiérarchie. Le nœud de démarrage peut être spécifié par les propriétés `StartFromCurrentNode`, `StartingNodeUrl` ou `StartingNodeOffset`.

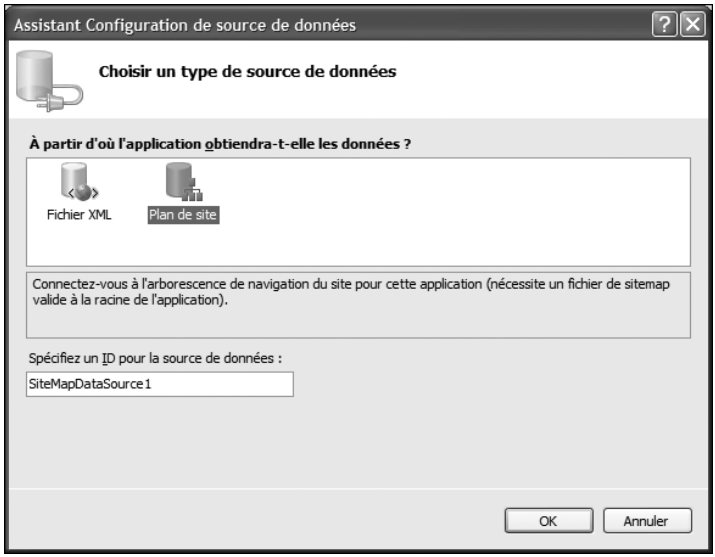
Assistant Configuration de source de données

Pour lier un contrôle `TreeView` à une source de données, vous pouvez également utiliser l'Assistant Configuration de source de données. Vous pouvez l'ouvrir à partir de la propriété `DataSourceId` dans l'Éditeur de propriétés et soit utiliser une source de données existante, soit en créer une nouvelle.



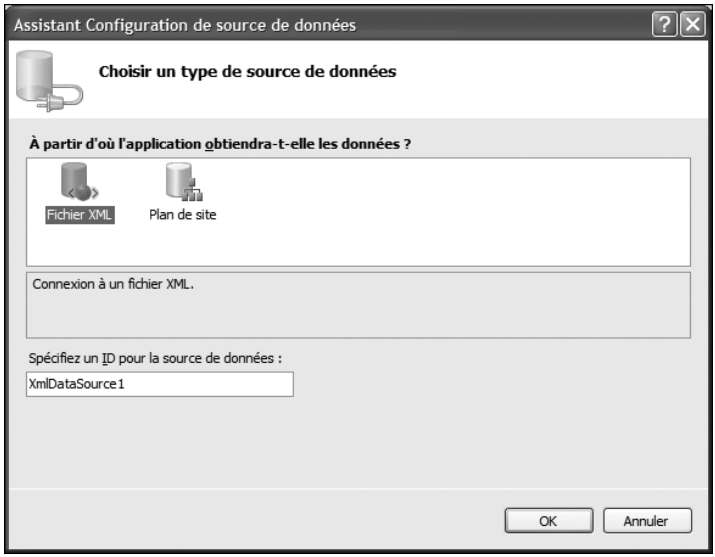
◄ **Figure 19-13 :** Propriété DataSourceId

Vous pouvez sélectionner un contrôle `SiteMapDataSource`.



▲ **Figure 19-14 :** Ajout d'un contrôle SiteMapDataSource

Vous pouvez également sélectionner un contrôle `XmlDataSource`.



▲ **Figure 19-15 :** Ajout d'un contrôle XmlDataSource

19.7 Check-list

Le contrôle `TreeView` était très attendu par les développeurs car de nombreuses fonctionnalités nécessitent de faire appel à une d'arborescence pour afficher des données hiérarchiques. Il s'agissait de fournir un contrôle capable de satisfaire tous les cas possibles.

De ce point de vue, `TreeView` répond parfaitement à cette demande puisqu'il dispose notamment des fonctionnalités suivantes, dont vous avez pu avoir un aperçu dans ce chapitre :

- affichage des données statiques et dynamiques ;
- personnalisation des styles ;
- association d'images aux nœuds ;
- remplissage dynamique des nœuds ;
- remplissage des nœuds sans rechargement de page ;
- liaison de données entre les propriétés du contrôle et la source de données.

Site multilingue avec stockage des ressources en base de données

La localisation d'un site web est souvent aussi primordiale que difficile à mettre en œuvre. Il s'agit de traduire tous les textes à afficher mais également le format des dates, des nombres ou encore des prix, qui varie selon la culture. On parle en effet de culture et non de langue, car une même langue peut regrouper plusieurs pays et cultures différents.

Les développeurs qui souhaitent créer une application multilingue devront adapter toutes leurs pages à la culture de l'utilisateur en cours. Cela peut même aller jusqu'au sens de lecture (on lit en effet de droite à gauche dans certaines cultures).

Dans ce chapitre, nous aborderons tous les concepts relatifs à la localisation d'un site web. Vous aurez ainsi l'occasion de vous familiariser avec le modèle de localisation d'ASP .NET 2.0 et avec les notions à maîtriser. Vous apprendrez tout d'abord à localiser les pages d'un site web selon la manière habituelle, c'est-à-dire à l'aide des fichiers XML dans lesquels vous déclarerez les clés et les valeurs de chaque ressource. Vous verrez ensuite comment étendre ces expressions de ressource en implémentant un fournisseur de ressources personnalisé qui stockera ces dernières dans une table de base de données.

20.1 Classes et espaces de noms utilisés

L'espace de noms `System.Globalization` regroupe l'ensemble des classes qui ont un rapport avec tout ce qui est spécifique et qui peut varier selon la culture de l'ordinateur ou de l'utilisateur en cours, notamment des classes qui interviennent dans différents domaines de la programmation :

- les classes de formatage de données (dates, nombres…) ;
- les classes d'encodage de fichiers et de caractères (UTF-8, ISO 8859-1, Unicode…) ;
- les classes de comparaison et des ordres de tri ;
- les calendriers spéciaux (chinois, grégorien, hébreux, japonais…) ;
- les cultures et sous-cultures existantes.

Les espaces de noms `System.Resources` et `System.Web.UI.Design` seront également utilisés lors de l'implémentation du fournisseur de ressources personnalisé. Pour commencer, voyons une définition des concepts mis en œuvre lors de la localisation d'un site, et des exemples expliquant la manière traditionnelle de déclarer des ressources destinées à traduire le contenu d'un site.

Noms de culture

Un petit rappel s'impose sur les normes concernant les noms de culture. Un système de code simple et efficace a été mis en place, permettant de distinguer les cultures et leurs sous-cultures.

Les codes des cultures correspondent aux conventions ISO (International Standard Organization). Les deux premières lettres en minuscules correspondent à la langue (exemple : fr) et s'appuient sur la norme ISO 639-1, tandis que les deux lettres en majuscules (exemple : FR) qui suivent le tiret séparateur (-) correspondent au code du pays et doivent respecter la norme ISO 3166. La culture française spécifique à la Belgique s'écrit fr-BE. Le tout est repris dans la RFC 3066.

Extension de fichiers .resx

Vous retrouverez ces codes dans les extensions des noms de fichier *.resx*. C'est en effet de cette façon que vous distinguerez les fichiers de localisation des différentes cultures. Pour récupérer une ressource, le fournisseur de localisation recherche en fonction de la clé, la culture la plus adéquate.

Si la culture en cours est es-Mx par exemple, il cherche un fichier avec cette extension puis, s'il n'en trouve pas, il remonte à la culture parente et enfin à la culture neutre. C'est pourquoi il est indispensable de fournir à chaque fois un fichier sans extension de code de culture afin de pouvoir proposer une traduction par défaut.

20.2 Configuration

La section de configuration `globalization` du *web.config* permet de spécifier des règles d'encodage et de localisation à appliquer par défaut au sein de l'application.

```
<globalization
    requestEncoding = "format d'encodage"
    responseEncoding = "format d'encodage""
    fileEncoding = "format d'encodage"

    responseHeaderEncoding = "format d'encodage"
    resourceProviderFactoryType = "type"
    enableBestFitResponseEncoding = "true|false"

    culture="culture"
    uiCulture="culture"/>
```

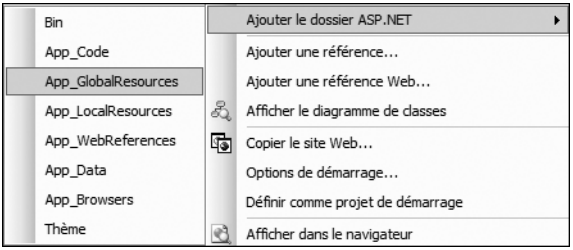
Attributs de l'élément globalization	
Attribut	**Description**
`culture`	Attribut facultatif. Spécifie la culture par défaut.
`fileEncoding`	Attribut facultatif. Spécifie le format d'encodage par défaut pour l'analyse des fichiers *.aspx*, *.asmx* et *.asax*.
`requestEncoding`	Attribut facultatif. Spécifie le format d'encodage supposé de chaque demande entrante, y compris les données publiées et la chaîne de requête.
`responseEncoding`	Attribut facultatif. Spécifie le format d'encodage du contenu des réponses.
`uiCulture`	Attribut facultatif. Spécifie la culture par défaut pour le traitement des recherches de ressources dépendant des paramètres régionaux.

Vous verrez plus loin comment initialiser de manière déclarative ou par programmation la culture de la page en cours.

20.3 Ressources globales et ressources locales

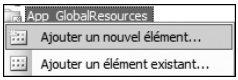
Le modèle de localisation ASP .NET 2.0 permet aux développeurs de déclarer des expressions déclaratives implicites ou explicites.

1. Pour utiliser des fichiers de ressources globales ou locales, il faut commencer par créer respectivement le dossier *App_GlobalRessources* ou *App_LocalRessources*. Pour cela, cliquez du bouton droit sur la racine du site et sélectionnez le dossier que vous voulez créer dans la liste de répertoires spéciaux que vous propose le menu **Ajouter un dossier ASP.NET**.



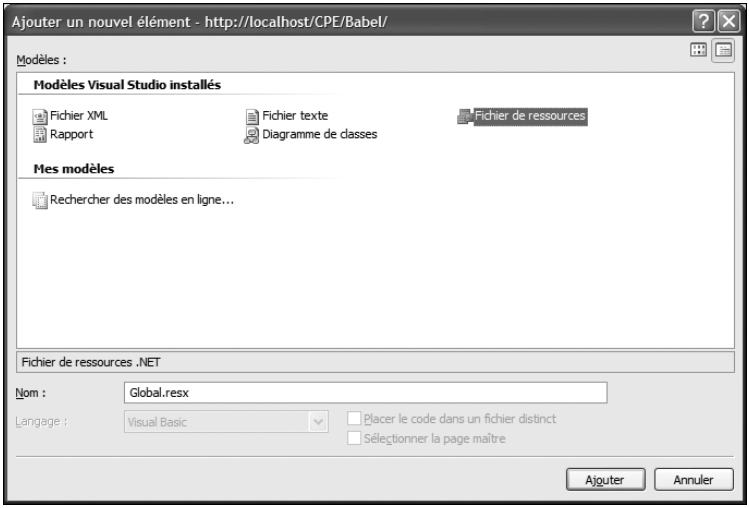
▲ **Figure 20-1** : Menu Ajouter un dossier ASP.NET

2. Pour ajouter un fichier de ressources, cliquez du bouton droit sur le dossier *App_GlobalResources* dans l'Explorateur de solutions, puis sélectionnez la commande **Ajouter un nouvel élément**.



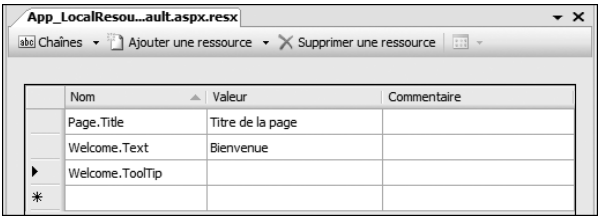
◄ **Figure 20-2 :** Commande Ajouter un nouvel élément

3 La boîte de dialogue **Ajouter un nouvel élément** s'ouvre et vous pouvez ajouter un fichier de ressources avec une extension *.resx*.



▲ **Figure 20-3 :** Boîte de dialogue Ajouter un nouvel élément

4 Visual Web Developer permet ensuite de modifier les fichiers *.resx* grâce à un éditeur spécifique complet et simple d'utilisation. Les ressources sont associées à des clés, c'est-à-dire à des identifiants uniques à partir desquels vous pourrez récupérer leurs valeurs. En ce qui concerne les chaînes de caractères, les ressources sont présentées dans une grille de données qui affiche les clés, les valeurs ainsi qu'une colonne de commentaires si vous voulez donner des détails.



▲ **Figure 20-4 :** Éditeur de ressources pour les chaînes de caractères

L'ajout d'une nouvelle ressource se fait très facilement grâce à une ligne vide en bas de la grille prévue à cet effet.

Ressources globales

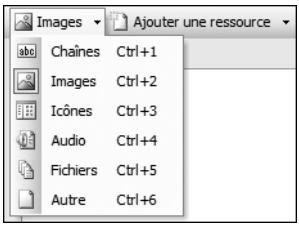
Le répertoire *App_GlobalRessources* à insérer à la racine du site permet de définir les ressources globales, comme les messages d'erreur ou les plans de site.

Ressources localisées

Le répertoire *App_LocalRessources* doit être placé dans chacun des répertoires du site web où se trouve une page à localiser. Vous pouvez alors localiser les propriétés des contrôles d'une page *Default.aspx* en créant le fichier *Default.aspx.resx* correspondant.

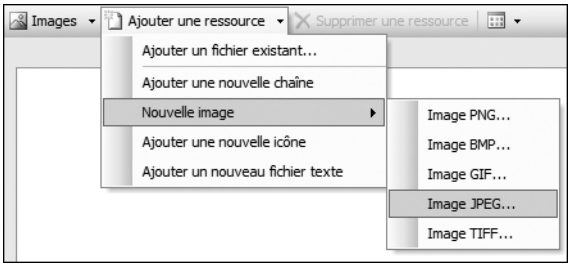
Autres types de ressources

Une ressource peut être associée non seulement à du texte mais aussi à d'autres fichiers, comme des images ou des fichiers audio. L'éditeur de ressources permet d'associer et de visualiser ces différents types de fichiers.



◄ **Figure 20-5 :** Sélecteur de type de ressource de l'éditeur

Un menu **Ajouter une ressource** permet également d'ajouter facilement les différents types de ressources possibles, comme ici une image. Il vous sera demandé la clé que vous voulez lui attribuer et votre éditeur d'images par défaut se lancera.



◄ **Figure 20-6 :** Menu Ajouter une ressource

Dans ce chapitre, nous nous intéresserons essentiellement aux chaînes de caractères qui représentent la majeure partie des ressources à localiser.

20.4 Expressions de ressource

Il existe plusieurs syntaxes déclaratives qui permettent de spécifier les ressources associées à un contrôle ou à ses propriétés. De nombreuses propriétés de contrôle serveur peuvent être localisées. Il faut pour cela que la propriété possède l'attribut `Localizable` avec la valeur `true`.

Il suffit alors d'utiliser la bonne expression pour relier la propriété à la ressource à laquelle elle doit être associée. Vous allez découvrir les différentes manières de déclarer une expression de ressource pour une propriété localisable. Mais auparavant, faisons un point sur ce concept d'expression en présentant ce sur quoi elles reposent, c'est-à-dire les Expression Builders.

Expression Builders

Les Expression Builders sont une fonctionnalité de parsing qui permet aux développeurs d'utiliser une syntaxe déclarative afin d'affecter certaines propriétés de contrôle. Les Expression Builders supportés par défaut sont notamment les suivants :

Expression Builders utilisables dans les propriétés des contrôles serveurs

Expression Builder	Description
`Resources`	Utilisé pour récupérer une ressource. Exemple : `<%$ resources: Resourcekey %>`.
`Connectionstrings`	Utilisé pour accéder à une chaîne de connexion. Exemple : `<%$ connectionstrings: ConnectionStringName %>`.
`Appsettings`	Utilisé pour accéder à un paramètre d'application. Exemple : `<%$ appsettings: AppKey %>`.

La syntaxe à utiliser contient le préfixe $, ce qui la distingue des méthodes de liaison de données (#) et d'interprétation de code (=).

Vous pouvez implémenter vos propres Expression Builders si vous voulez avoir à disposition ce genre de fonctionnalité de syntaxe déclarative. Vous devrez pour cela hériter des classes `ExpressionBuilder` et `ExpressionBuilderEditor`.

Cette fonctionnalité étant explicitée, passons maintenant aux expressions qui nous intéressent tout particulièrement, c'est-à-dire les expressions de ressource.

Expressions implicites

Les expressions implicites servent à déclarer un préfixe de ressource associé à un contrôle.

```
<asp:Control ID="Id" runat="server"
        meta:resourcekey="prefixe de ressource" />
```

Vous pouvez ensuite utiliser ce préfixe pour spécifier les propriétés que vous voulez affecter. Il vous suffira de déclarer une ressource dont la clé sera constituée de ce préfixe et de la propriété du contrôle que vous voulez localiser, le tout séparé par un point.

Par exemple, pour localiser la propriété `Text` d'un contrôle `Label` pour lequel vous avez affecté à l'attribut `meta:resourcekey` la valeur `MonLabel`, vous utiliserez comme clé de ressource `MonLabel.Text`. Vous aurez ainsi accès à l'ensemble des propriétés localisables tout en n'ayant fourni qu'un seul préfixe.

Expressions explicites

Les expressions explicites utilisent une syntaxe qui permet d'affecter aux propriétés des clés de ressource globale ou locale. La syntaxe de déclaration est la suivante :

```
<%$ Resources:[fichier resX global,] clé de ressource%>
```

Voici un exemple de l'affectation de la propriété `Text` d'un contrôle `Label` par une ressource globale :

```
<asp:Label ID="Label1" runat="server"
Text='<%$ Resources: monFichierResX, maClé%>' />
```

Pour comprendre un peu mieux cette syntaxe, consultez la section suivante sur les Expression Builders.

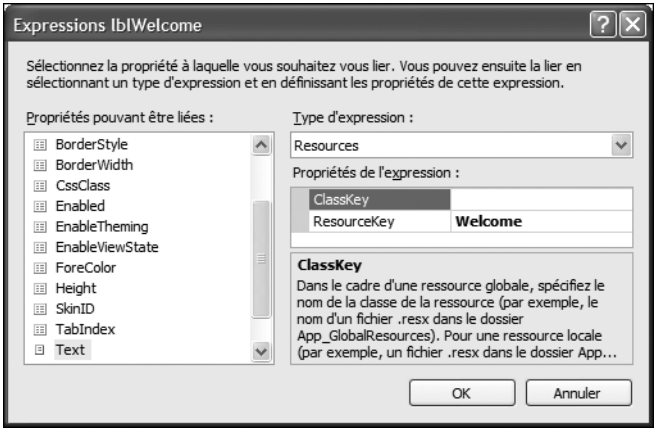
Assistant d'affectation des ressources

Un Assistant est également fourni dans l'éditeur de propriétés en mode Design pour faciliter l'affectation des ressources aux propriétés des contrôles. Pour y avoir accès, sélectionnez le contrôle dont vous voulez localiser les propriétés, puis affichez la fenêtre **Propriétés**. Une zone d'édition spéciale nommée *(Expressions)* de la catégorie *Données* permet alors d'afficher l'Assistant à l'aide d'un bouton :

◄ **Figure 20-7 :** Propriétés (Expressions) d'un contrôle

Cet Assistant dispose de nombreuses fonctionnalités intéressantes qui permettent d'associer des clés de ressource globale ou locale à chaque propriété localisable du contrôle en cours d'édition.

Si vous avez utilisé la syntaxe avec l'Assistant Expression Builder, l'Assistant saura récupérer les attributs que vous avez configurés et affichera des icônes différentes devant les propriétés localisées.

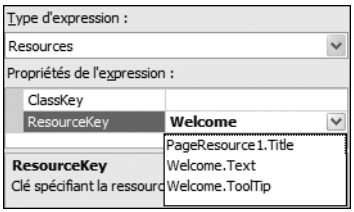


▲ **Figure 20-8 :** Assistant de ressources

Dans l'éditeur de propriétés, si vous associez une expression de ressource à la propriété, elle s'affiche de manière un peu différente dans la liste :

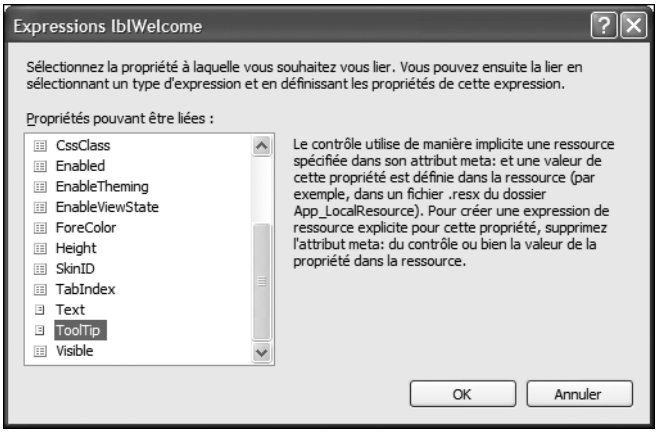
◄ **Figure 20-9 :** Propriété Text associée à une expression de ressource

Si le ficher *.resx* contient des clés de ressource, une liste vous permettra de sélectionner la clé que vous voulez associer à la propriété en affichant toutes celles disponibles dans le fichier.



◄ **Figure 20-10 :** Liste des ressources disponibles

Si, comme nous vous l'avons conseillé, vous avez plutôt affecté l'attribut `meta:resourcekey`, l'Assistant affiche alors un texte qui indique que les propriétés ne peuvent pas être éditées. Vous pouvez néanmoins distinguer les propriétés inscrites dans le fichier *.resx* grâce à une icône spécifique.



▲ **Figure 20-11 :** Assistant de ressources d'un contrôle dont l'attribut meta:resourcekey est défini

20.5 Fonctionnalités intéressantes relatives à la localisation

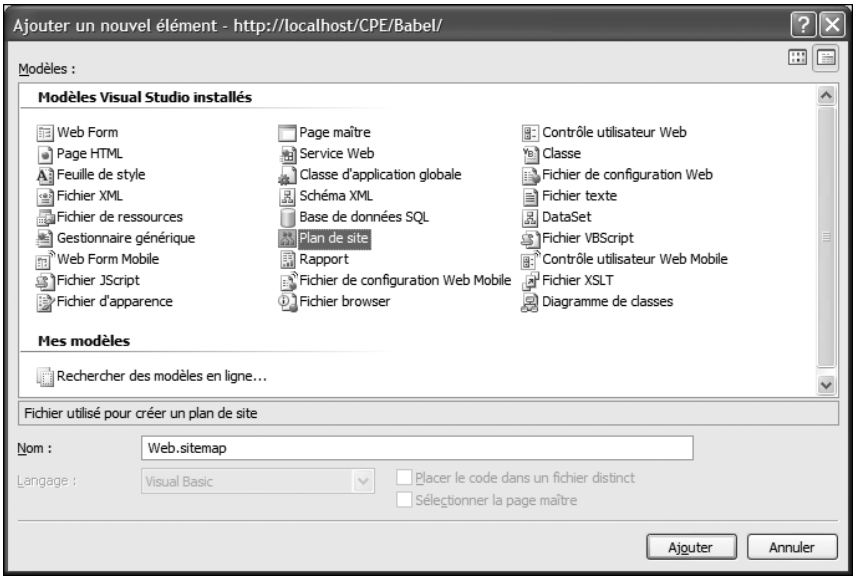
Accès par programmation aux ressources

Vous pouvez récupérer les valeurs des ressources que vous avez créées dans votre code grâce à deux méthodes : `GetGlobalResourceObject` et `GetLocalResourceObject`. Elles prennent en paramètre la clé de la ressource :

```
Button1.Text = GetLocalResourceObject("Button1.Text").ToString()

Image1.ImageUrl = CType(GetGlobalResourceObject _
("RessourcesGlobales", "LogoUrl"), String)
```

Localisation de plans de site

Les propriétés des nœuds des plans de site peuvent également être localisées. Pour ajouter un plan de site, cliquez du bouton droit sur la racine du site dans l'Explorateur de solutions, puis sélectionnez la commande **Ajouter un nouvel élément**. La boîte de dialogue **Ajouter un nouvel élément** s'ouvre alors. Vous pouvez ajouter un fichier de plan de site, avec une extension *.sitemap*.



▲ **Figure 20-12 :** Ajout d'un plan de site

Astuce

Fournisseur de navigation par défaut

Le fournisseur de navigation par défaut est chargé de récupérer le fichier *web.sitemap* situé à la racine du site web. Si vous appelez votre plan de site comme cela et si vous le placez au bon endroit, vous n'avez pas à l'ajouter dans la section de configuration des fournisseurs de navigation.

Vous pouvez utiliser un fichier de ressources pour que les titres et les descriptions des pages dans vos contrôles de navigation soient traduits. La syntaxe à utiliser pour localiser un nœud ressemble beaucoup aux expressions de ressource mais varient un peu car elle est adaptée au fichier XML :

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/
    SiteMap-File-1.0" enableLocalization="true">
    <siteMapNode url="~/default.aspx"
        title="$resources: SiteMap, Home, Accueil"
        description="">
    </siteMapNode>
</siteMap>
```

Pour cela, vous devez activer la localisation en affectant la value `true` à l'attribut `enableLocalization` du nœud racine du plan de site.

Vous pouvez utiliser un fichier de ressources locales ou aller chercher vos clés dans un fichier de ressources globales. Pour les ressources globales, la syntaxe à employer est :

```
$resources:ClassName, KeyName, DefaultValue
```

Vous avez également la possibilité de spécifier l'attribut `resourcekey`, qui à la même fonction que l'attribut `meta:resourcekey` mais pour les nœuds XML.

Génération automatique de ressources locales

Visual Web Developer intègre une fonctionnalité de génération automatique des ressources locales contenues dans une page. Cela permet de générer de manière simple l'ensemble des ressources relatives à une page. Il suffit pour cela de faire basculer l'éditeur de la page en cours en mode Design, plutôt qu'en mode Source, puis de cliquer sur le bouton **Générer la ressource locale** de la barre d'outils *Mise en forme*.

◄ **Figure 20-13 :** Bouton Générer la ressource locale de la barre d'outils Mise en forme

Un fichier *.resx* correspondant au nom de la page en cours sera alors automatiquement créé. Vous retrouverez dans ce fichier la totalité des ressources affectées aux propriétés des contrôles. Ces propriétés seront également modifiées pour correspondre à la syntaxe d'affectation des ressources. En effet, si vous n'avez spécifié aucune expression de ressource pour une propriété localisable, l'attribut `meta:resourcekey` sera ajouté automatiquement avec comme valeur l'identifiant du contrôle. Même la directive de page subira cette modification, ce qui permettra entre autres de pouvoir localiser le titre de la page. Vous n'aurez alors plus qu'à vous servir de ce fichier comme modèle afin de créer les fichiers de localisation pour les autres cultures.

Remarque

Barre d'outils Mise en forme

Mise en forme Si la barre d'outils *Mise en forme* n'est pas disponible, cliquez du bouton droit sur un emplacement libre de la zone des barres d'outils. Un menu apparaît alors dans lequel vous pouvez sélectionner cette barre.

Il est donc préférable de ne réaliser cette action que lorsque vous êtes certain que vous n'allez plus ajouter des contrôles à la page. L'avantage lors du développement de vos pages est que vous devez uniquement vous soucier de placer les textes qui seront à traduire dans des propriétés ou des contrôles localisables. Afin de mieux contrôler le nom des ressources, veillez néanmoins à spécifier l'attribut `meta:resourcekey` de chaque contrôle. La localisation peut alors être déléguée à n'importe quelle personne sachant se servir de l'éditeur de ressources, et donc sans connaissance requise en programmation.

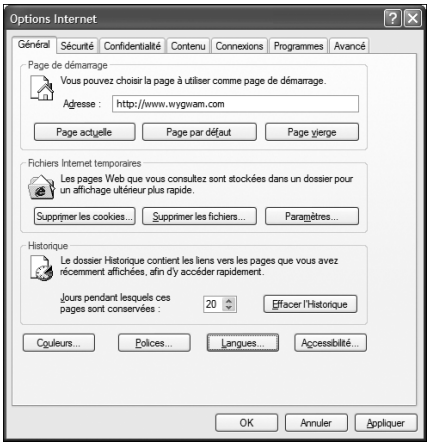
Auto-détection de la culture du navigateur

Vous pouvez facilement adapter le comportement de votre application à la culture de l'utilisateur en cours en vous basant sur le langage du navigateur Internet qu'il utilise pour y accéder. Pour cela, il suffit de spécifier les attributs `culture` et `uiCulture` avec la valeur `auto` dans la directive de page. Vous pouvez également préciser une culture par défaut en l'ajoutant après un deux-points (`:`) :

```
<%@ Page Culture="auto:fr-FR" %>
```

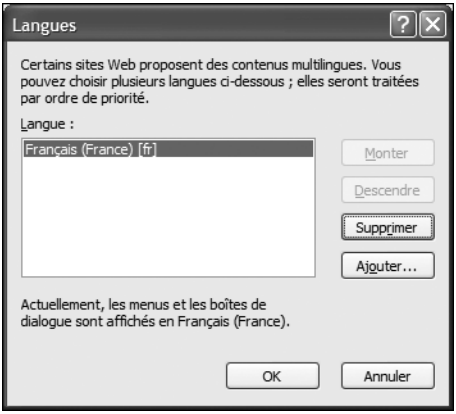
Voici les quelques étapes à suivre afin de déclarer plusieurs cultures pour votre navigateur et de basculer entre elles pour effectuer vos tests. Cette procédure montre comment configurer Internet Explorer, mais tous les navigateurs Internet ont cette même fonctionnalité.

1. Dans le menu **Outils**, cliquez sur la commande **Options Internet**. Dans la fenêtre qui s'ouvre, sous l'onglet **Général**, cliquez sur le bouton **Langues** situé en bas.



▲ **Figure 20-14 :** Fenêtre Options Internet

2 La boîte de dialogue **Langues** s'ouvre alors avec la liste des cultures disponibles. Des boutons **Monter** et **Descendre** permettent de placer les cultures dans l'ordre de préférence.



◄ **Figure 20-15 :** Boîte de dialogue Langues

3 Le bouton **Ajouter** ouvre une liste à partir de laquelle vous pouvez sélectionner les cultures que vous voulez ajouter et les ordonner pour placer en tête celle que vous préférez.

◄ **Figure 20-16 :** Liste des cultures disponibles susceptibles d'être ajoutées

Initialisation de la culture d'une page

Pour initialiser la culture d'une page si, par exemple, vous l'avez stockée par un mécanisme de persistence personnalisé (session, profil, viewstate, cookie…), le moyen le plus indiqué est de substituer – redéfinir – la méthode `InitializeCulture` d'une page. Il suffit alors d'indiquer la culture à utiliser via les

propriétés CurrentCulture et CurrentUICulture de la classe System.Threading.Thread.CurrentThread, qui correspondent à la culture en cours.

Voici un exemple sur lequel vous pouvez vous baser pour écrire ce genre de code d'initialisation de la culture :

```
Protected Overrides Sub InitializeCulture()
    System.Threading.Thread.CurrentThread.CurrentCulture _
        = System.Globalization.CultureInfo. _
          CreateSpecificCulture("fr-FR")
    System.Threading.Thread.CurrentThread. _
        CurrentUICulture = New System.Globalization. _
        CultureInfo("fr-FR")
    MyBase.InitializeCulture()
End Sub
```

20.6 Implémentation d'un fournisseur de ressources

Vous allez maintenant passer à l'implémentation du fournisseur de ressources qui va permettre de stocker les ressources dans une base de données. Le principe est de créer une classe qui hérite de la classe abstraite ResourceProviderFactory et d'y implémenter vos propres méthodes de stockage et de récupération des ressources. Ensuite, il suffit de modifier dans le *web.config* l'attribut resourceProviderFactoryType de la section globalization afin d'indiquer au site web que vous utilisez, non plus le mécanisme de localisation par défaut, mais le vôtre.

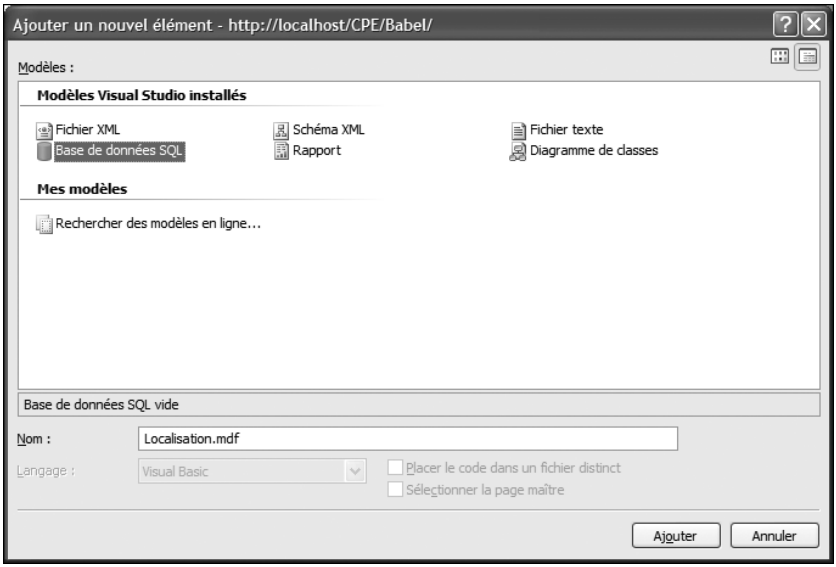
```
<globalization
 resourceProviderFactoryType="SQLResourceProviderFactory"/>
```

Avant d'entrer dans le détail des développements à effectuer, vous allez tout d'abord créer la base de données destinée à stocker les ressources.

Accès aux données

Afin de stocker les ressources, vous allez créer la base de données SQLExpress nommée *Localisation* et la table *Resource* correspondante qui stockera les ressources.

1. Vous allez tout d'abord ajouter le dossier ASP .NET *App_Data* et y ajouter votre base de données.

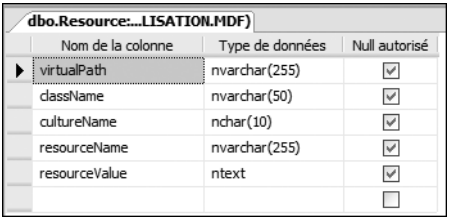


▲ **Figure 20-17 :** Boîte de dialogue Ajouter un nouvel élément

Cette table servira à la fois aux ressources globales et aux ressources locales puisque ce qui les différencie n'est en réalité que le nom de la classe de ressource globale. La structure de la table est donc prévue pour recevoir l'ensemble des informations spécifiques aux expressions de localisation.

Les champs sont les suivants :

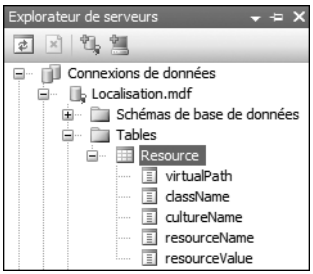
– *virtualPath* : le nom de la page pour une ressource locale.
– *className* : le nom de la classe pour une ressource globale.
– *cultureName* : le code de la culture.
– *resourceName* : la clé de la ressource.
– *resourceValue* : la valeur de la ressource.


dbo.Resource:...LISATION.MDF)

Nom de la colonne	Type de données	Null autorisé
virtualPath	nvarchar(255)	☑
className	nvarchar(50)	☑
cultureName	nchar(10)	☑
resourceName	nvarchar(255)	☑
resourceValue	ntext	☑
		☐

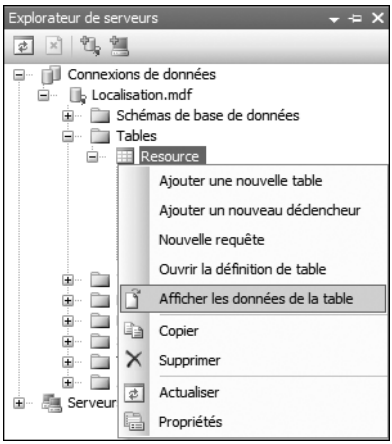
◄ **Figure 20-18 :** Structure de la table Resource

2 Après avoir créé la table, utilisez l'Explorateur de serveur pour naviguer dans la base et sélectionner la table *Resource*.



◄ **Figure 20-19 :** Affichage de la base de données dans l'Explorateur de serveur

3 Vous pouvez ensuite afficher les données de la table en cliquant dessus du bouton droit et en sélectionnant la commande **Afficher les données de la table**.



◄ **Figure 20-20 :** Commande Afficher des données de la table

4 Il suffit alors d'ajouter les ressources dont vous voulez disposez à l'aide de la grille de données qui affiche chaque enregistrement ainsi que les colonnes.

Resource: Req...LISATION.MDF)

	virtualPath	className	cultureName	resourceName	resourceValue
▶	default.aspx	*NULL*	fr-FR	Welcome	Bienvenue
	default.aspx	*NULL*	en-US	Welcome	Welcome
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

▲ **Figure 20-21 :** Affichage des données de la table

ResourceProviderFactory

Deux méthodes principales doivent notamment êtres substituées : `CreateLocalResourceProvider` et `CreateGlobalResourceProvider`.

Cette classe est assez simple puisqu'elle utilise un objet `SQLResourceProvider` qui englobe toute la logique de récupération des ressources :

```
Public NotInheritable Class SQLResourceProviderFactory
    Inherits ResourceProviderFactory

    Public Overrides Function CreateLocalResourceProvider _
        ( ByVal virtualPath As String) As IResourceProvider
        virtualPath = _
            System.IO.Path.GetFileName(virtualPath)
        Return New SQLResourceProvider(virtualPath, _
            Nothing)
    End Function

    Public Overrides Function _
        CreateGlobalResourceProvider ( _
        ByVal className As String) As IResourceProvider
        Return New SQLResourceProvider(Nothing, className)
    End Function
End Class
```

SQLResourceProvider

La classe `SQLResourceProvider` implémente l'interface `IResourceProvider` et doit donc fournir l'implémentation de la méthode `GetObject`, qui renverra une ressource, et de la propriété `ResourceReader`, qui renverra un énumérateur de ressources.

SQLResourceHelper

Cette classe d'aide à la récupération et au stockage des ressources est chargée de réaliser les accès en base. Elle contient donc une méthode `GetResources` de récupération des ressources. Son prototype est le suivant :

```
Public Shared Function GetResources( _
ByVal virtualPath As String, _
ByVal className As String, _
ByVal cultureName As String, _
ByVal designMode As Boolean, _
ByVal serviceProvider As IServiceProvider) As IDictionary
```

Selon les paramètres passés (si leur valeur est nulle ou non), cette méthode renvoie des ressources globales ou locales pour une certaine culture. Par

exemple, la récupération d'une ressource locale nécessitera le nom du fichier, et non le nom de la classe, tandis que pour une ressource globale, ce sera l'inverse.

SQLDesignTimeResourceProviderFactory

Pour que la fonctionnalité de génération automatique soit également prise en charge par votre fournisseur de ressources, vous devez créer une classe `SQLDesignTimeResourceProviderFactory` qui hérite de `DesignTimeResourceProviderFactory` dans l'espace de noms `System.Web.UI.Design`.

Cette classe peut être aussi simple que `ResourceProviderFactory` puisqu'elle aussi délègue toutes ces opérations complexes à deux classes, `DesignTimeGlobalResourceProvider` et `DesignTimeLocalResourceProvider`.

```
Public Class SQLDesignTimeResourceProviderFactory
    Inherits DesignTimeResourceProviderFactory
    Public Overrides Function _
        CreateDesignTimeGlobalResourceProvider(_
        ByVal serviceProvider As IServiceProvider, _
        ByVal applicationKey As String) _
        As IResourceProvider
        Return New DesignTimeGlobalResourceProvider _
           (applicationKey)
    End Function
    Public Overrides Function _
        CreateDesignTimeLocalResourceProvider( _
        ByVal serviceProvider As IServiceProvider) _
        As IResourceProvider
        If _localResourceProvider Is Nothing Then
            _localResourceProvider = _
                New DesignTimeLocalResourceProvider _
                    (serviceProvider)
        End If
        Return _localResourceProvider
    End Function
    Public Overrides Function _
        CreateDesignTimeLocalResourceWriter(_
        ByVal serviceProvider As IServiceProvider) _
        As IDesignTimeResourceWriter
        Return New DesignTimeLocalResourceProvider _
            (serviceProvider)
    End Function
End Class
```

Après toutes ces étapes, votre fournisseur de ressources stockées dans une base de données est maintenant effectif. Il ne reste plus qu'à créer les pages et à

ajouter les ressources pour localiser les propriétés des contrôles que vous voudrez traduire pour différentes cultures.

20.7 Check-list

La localisation des sites web a été considérablement simplifiée avec le Framework ASP .NET 2.0. Il fournit désormais nativement une fonctionnalité de déclaration des ressources par culture dans des fichiers XML et un mécanisme d'affectation de ces ressources à des propriétés de contrôle grâce à une syntaxe déclarative.

Vous pouvez de plus implémenter vos propres méthodes de stockage et de récupération des ressources.

Dans ce chapitre, nous avons abordé les sujets suivants :

- la localisation d'un site web à l'aide de fichier *.resx* ;
- l'accès déclaratif et impératif aux ressources dans les pages web ;
- l'accès par programmation aux ressources ;
- la localisation de plans de site ;
- l'initialisation de la culture d'une page ;
- la génération automatique de ressources locales ;
- l'auto-détection de la culture d'un navigateur ;
- l'implémentation d'un fournisseur de ressources pour stocker des ressources dans une base.

Atlas

Atlas est un Framework gratuit qui permet de développer des applications web de nouvelle génération, plus riches, plus interactives et multinavigateurs.

Après une présentation d'Atlas, vous utiliserez les contrôles Atlas à travers quelques exemples. Vous découvrirez avec quelle facilité il est possible de développer des fonctionnalités qui nécessitaient auparavant des connaissances avancées de langages côté client, comme JavaScript.

21.1 Présentation d'Atlas

Atlas se présente comme une extension du Framework .NET 2.0. Il permet aux développeurs de créer des applications web riches, qui tirent avantage à la fois des fonctionnalités du navigateur et du code serveur.

Ajax et XMLHttpRequest

Ce type de développement qu'utilise Atlas est fortement lié au concept d'Ajax (Asynchronous JavaScript and XML). Ce terme est maintenant largement employé et parfois même usurpé. Il représente à la fois la technologie d'appel de procédure de manière asynchrone et un ensemble de technologies.

Les navigateurs évolués sont capables d'utiliser l'objet `XMLHttpRequest`, qui peut être appelé à l'aide de code JavaScript pour faire des appels au serveur. Cela permet notamment de réaliser des rafraîchissements partiels de page. Le problème principal que Ajax tente de résoudre est enraciné dans le protocole HTTP lui-même. Les navigateurs communiquent en effet via HTTP avec les serveurs web pour récupérer le contenu des pages et publier leurs données. Le protocole est sans état, ce qui signifie que la conservation de la saisie de l'utilisateur entre chaque appel de page est à la charge du serveur.

Bibliothèque de scripts clients

La bibliothèque de scripts clients d'Atlas est divisée en plusieurs parties.

Le script principal comprend les couches inférieures sur lesquelles le reste est construit. On trouve ensuite la couche de compatibilité internavigateur. Il s'agit d'une abstraction permettant d'écrire des scripts sans se soucier de savoir s'ils vont fonctionner sur tous les navigateurs. En outre, grâce à cette couche, Atlas pourra évoluer en fonction des améliorations des navigateurs.

Le système de types se trouve au-dessus de cette couche de compatibilité. Il permet de développer du JavaScript avec une approche orientée objet. Vous pouvez ainsi créer des espaces de noms et des classes. L'héritage est également pris en charge ainsi que les interfaces, les délégués et les énumérations, en d'autres termes la plupart des concepts que vous retrouvez dans les langages du Framework .NET.

La bibliothèque de classes de base complète la bibliothèque de scripts. Vous retrouverez donc des classes et espace de noms habituels.

Les fonctionnalités du Framework Atlas ne se limitent pas à une bibliothèque de scripts Ajax. On y trouve aussi des fonctionnalités côté serveur qui permettent d'améliorer facilement les applications ASP .NET existantes.

Contrôles Atlas

L'architecture d'Atlas se compose d'un modèle de composants et de contrôles. Les contrôles serveurs permettent notamment d'éviter les rafraîchissements de page.

Le contrôle `ScriptManager` modifie le comportement de publication client tandis que le contrôle `UpdatePanel` gère le cycle de vie de la page côté serveur avant d'effectuer les changements nécessaires.

Affectez la propriété `EnablePartialRendering` du `ScriptManager` à `true` :

```
<atlas:ScriptManager EnablePartialRendering="true" runat="server" />
```

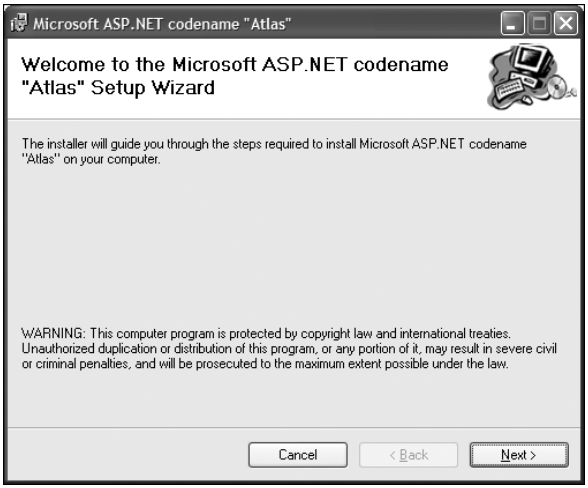
L'état de la page est rendu persistant au travers des requêtes de rendu partiel. Le code HTML est quant à lui mis à jour à l'aide de code JavaScript agissant sur le Document Object Model (DOM).

Atlas supporte également la sérialisation des données des web services en JSON. Les données JSON peuvent être désérialisées directement vers les objets JavaScript plus ou moins complexes. Cela simplifie l'accès aux services web à partir du navigateur.

21.2 Installation

Tout d'abord, vous devez installer Atlas sur votre machine de développement. Pour cela, téléchargez l'exécutable d'installation d'Atlas. Vous le trouverez facilement sur le site officiel **http://atlas.asp.net**. Vous y trouverez en outre de nombreux articles, les dernières actualités ainsi que des exemples d'applications et d'autres ressources. Voici les différentes étapes pour procéder à l'installation du Framework Atlas.

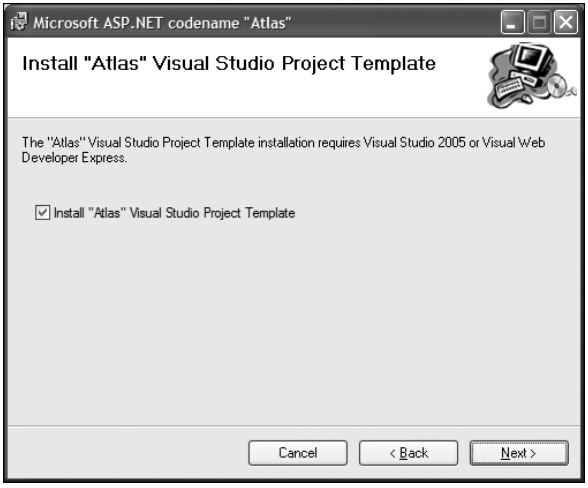
1. Lancez l'exécutable *AtlasSetup.msi*. L'Assistant d'installation s'ouvre.



▲ **Figure 21-1 :** Lancement de l'Assistant d'installation d'Atlas

2 Cliquez sur **Suivant** puis accepter le contrat d'utilisation.

3 Cliquez de nouveau sur **Suivant**. L'Assistant vous demande si vous souhaitez installer le modèle de projet Atlas pour Visual Studio.



▲ **Figure 21-2 :** Case à cocher Installer le modèle de projet Visual Studio Atlas

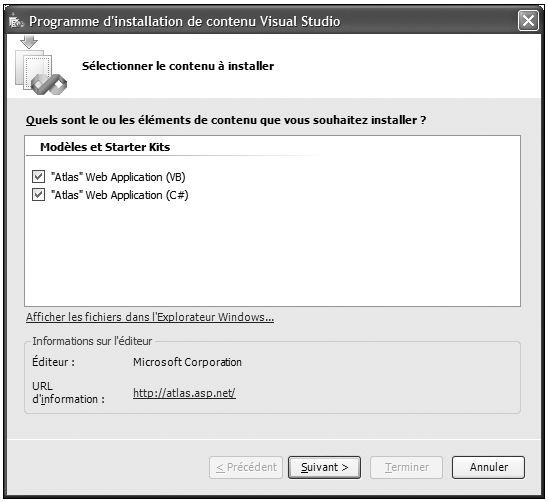
4 Cliquez sur **Suivant**. Une case à cocher vous propose d'enregistrer l'extension de fichier *.asbx* dans IIS.

▲ **Figure 21-3 :** Ajout de l'extension de fichier .asbx dans IIS

L'extension *.asbx* désigne les fichiers Atlas qu'on appelle "bridges". Il s'agit de composants qui communiquent avec des services web situés en dehors de votre application.

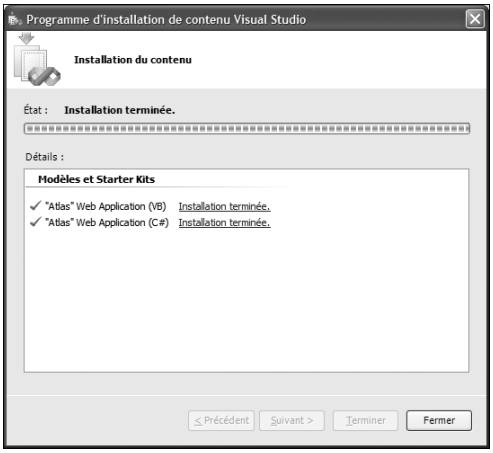
5 Terminez l'installation. Les fichiers associés sont installés dans le répertoire *C:\Program Files\Microsoft ASP.NET\Atlas*.

6 Si vous avez choisi d'installer le modèle de projet Visual Studio Content, l'Assistant d'installation se lance. Le premier écran affiche les modèles qui vont être installés et vous permet de ne sélectionner que ceux qui vous intéressent.



▲ **Figure 21-4 :** Liste des projets à installer

7 Vous pouvez ensuite cliquer sur **Terminer** pour exécuter l'installation. Les liens situés derrière chaque projet installé fournissent des informations sur le déroulement de l'installation.



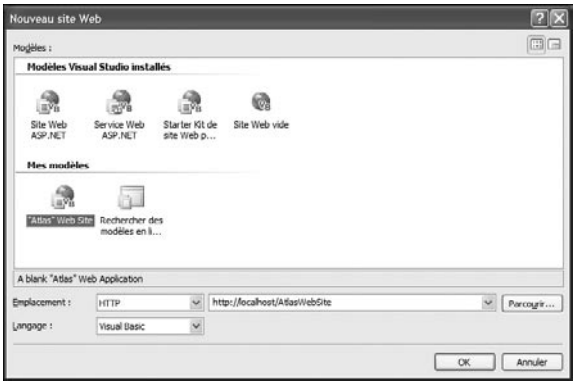
▲ **Figure 21-5 :** Écran récapitulatif de l'installation

21.3 Création d'un site web Atlas

Après avoir installé le modèle de projet Visual Studio, vous pouvez créer un nouveau site web Atlas.

1 Pour cela, cliquez du bouton droit sur la racine de la solution, puis sélectionnez la commande **Ajouter un nouveau site web** du menu **Ajouter**.

2 Dans la catégorie *Mes modèles*, sélectionnez le modèle *Atlas Web Site*.



▲ **Figure 21-6 :** Écran récapitulatif de l'installation

3 Sélectionnez l'emplacement du site, le langage Visual Basic, puis cliquez sur OK.

Le modèle de site web Atlas Web inclut l'assembly *Microsoft.Web.Atlas.dll* dans son dossier *Bin* ainsi qu'un fichier *Web.config* qui contient des sections spécifiques permettant d'activer les fonctionnalités d'Atlas.

Configuration du Web.config

Les sections de configuration personnalisées permettent de déclarer les nouvelles sections que vous pouvez utiliser, comme les services d'authentification et de profil asynchrones.

```
<configSections>
  <sectionGroup name="microsoft.web" type=
"Microsoft.Web.Configuration.MicrosoftWebSectionGroup">
    <section name="converters"
    ➥ type="Microsoft.Web.Configuration.ConvertersSection"
    ➥ requirePermission="false"/>
    <section name="webServices" type=
"Microsoft.Web.Configuration.WebServicesSection"
requirePermission="false"/>
    <section name="authenticationService" type=
"Microsoft.Web.Configuration.AuthenticationServiceSection"
➥ requirePermission="false"/>
    <section name="profileService"
    ➥ type="Microsoft.Web.Configuration.ProfileServiceSection"
    ➥ requirePermission="false"/>
  </sectionGroup>
</configSections>
```

Il est également intéressant d'importer les espaces de noms contenant les nouveaux contrôles serveurs intégrés au Framework Atlas. Cela évite d'inclure les directives `Register` dans chaque page qui utilisera ces contrôles.

```
<pages>
  <controls>
    <add namespace="Microsoft.Web.UI"
assembly="Microsoft.Web.Atlas" tagPrefix="atlas"/>
    <add namespace="Microsoft.Web.UI.Controls"
assembly="Microsoft.Web.Atlas" tagPrefix="atlas"/>
  </controls>
</pages>
```

Les fichiers *.asmx* sont associés à un nouvel handler (un gestionnaire de requêtes) afin que les serveurs proxy JavaScript puissent fonctionner.

```
<httpHandlers>
    <remove verb="*" path="*.asmx"/>
    <add verb="*" path="*.asmx"
    ➥ type="Microsoft.Web.Services.ScriptHandlerFactory"
validate="false"/>
    <add verb="*" path="atlasbatchcall.axd"
    ➥ type="Microsoft.Web.Services.MultiRequestHandler"
validate="false"/>
    <add verb="*" path="atlasglob.axd"
type="Microsoft.Web.Globalization.GlobalizationHandler"
validate="false"/>
    <add verb="*" path="*.asbx"
    ➥ type="Microsoft.Web.Services.ScriptHandlerFactory"
validate="false"/>
    </httpHandlers>
```

Des modules de service sont ajoutés.

```
<httpModules>
    <add name="ScriptModule"
type="Microsoft.Web.Services.ScriptModule"/>
    <add name="BridgeModule"
type="Microsoft.Web.Services.BridgeModule"/>
    <add name="WebResourceCompression" type=
"Microsoft.Web.Services.WebResourceCompressionModule"/>
</httpModules>
```

Ajout des contrôles Atlas à la boîte à outils

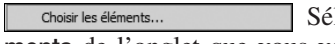
Pour utiliser facilement les contrôles Atlas dans votre site, vous allez les ajouter à la boîte à outils de Visual Studio.

1 

Sélectionnez la commande **Ajouter un onglet** de la boîte à outils.

2 Nommez cet onglet `Atlas`.

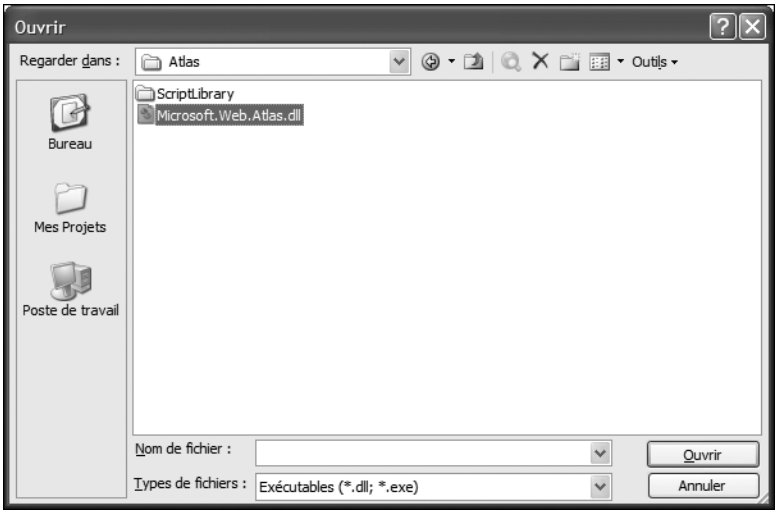
◀ **Figure 21-7 :** Création de l'onglet Atlas

3 

Sélectionnez la commande **Choisir les éléments** de l'onglet que vous venez de créer.

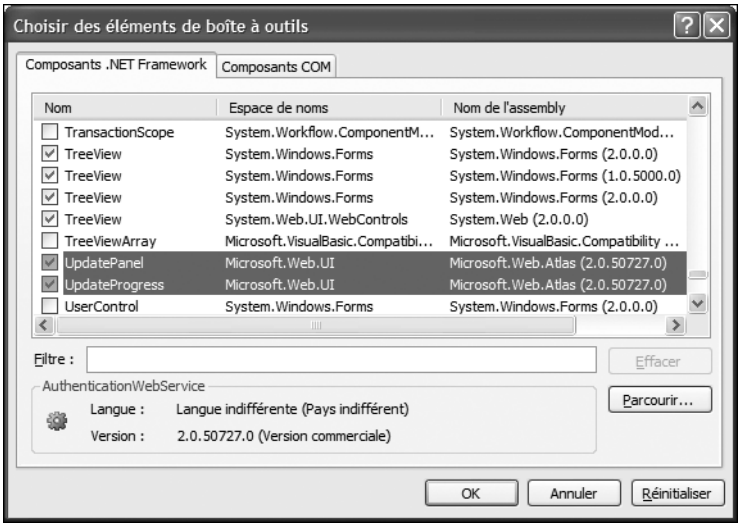
La fenêtre qui s'ouvre affiche l'ensemble des contrôles disponibles. Ceux accompagnés d'une case à cocher sélectionnée sont déjà ajoutés à un onglet de la boîte à outils.

Cliquez sur le bouton **Parcourir** puis recherchez l'emplacement *C:\Program Files\Microsoft ASP.NET\Atlas\v2.0.50727\Atlas* pour sélectionner l'assembly qui contient les contrôles Atlas que vous voulez avoir à disposition.



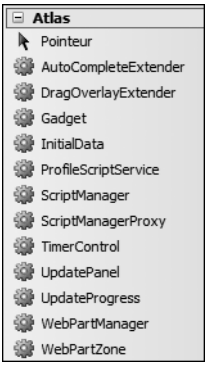
▲ **Figure 21-8 :** Sélection de l'assembly Microsoft.Web.Atlas.dll

4 La fenêtre sélectionne les nouveaux contrôles contenus dans cet assembly et les coche pour ajouter leurs références dans la boîte à outils. Cliquez sur OK.



▲ **Figure 21-9 :** Importation des contrôles serveurs Atlas

5 Les contrôles Atlas sont alors ajoutés à l'onglet.



◄ **Figure 21-10** : Liste des contrôles Atlas

Avant d'utiliser le Framework Atlas dans des pages web, voyons les différentes syntaxes du code Atlas.

21.4 Syntaxe du code Atlas

Il existe trois façons d'écrire du code Atlas.

Mode impératif

Le code impératif n'est ni plus ni moins que l'utilisation du Framework JavaScript Atlas dans vos pages. Cette méthodologie nécessite une connaissance avancée du langage JavaScript et des différents éléments du Framework Atlas. Ce mode d'utilisation est cependant nécessaire au développement de contrôles serveurs Atlas.

```
<input id="button" type="button" onclick="GetValues" />
<script type="text/javascript"
  src="WebServiceAtlas.asmx/js"></script>
<script type="text/javascript">
function GetValues() {
   WebServiceAtlas.getValues(GetValues_CallBack);
}
function GetValues_CallBack(result){
//…
}
</script>
```

Mode déclaratif

Le mode déclaratif consiste à ajouter une balise de type `"text/xml-script"` permettant la description du comportement de votre page. Ce mode de travail

par description est certainement le plus compréhensible pour quelqu'un débutant avec Atlas. Il permet la mise en place de comportement côté client au sein d'une page de façon simple et rapide.

```
<input id="button" type="button" value="OK" />
<script type="text/xml-script">
<page…>
<dataSource id="dataSource"
  serviceURL="WebServiceAtlas.asmx" />
<button id="buttonAction" targetElement="button">
<click>
<invokeMethod target="dataSource" method="GetValues" />
//…
</click>
</button>
</page>
</script>
```

Mode serveur

Les contrôles serveurs disponibles dans le Framework Atlas sont pour la plupart des extenders, c'est-à-dire des contrôles qui vont étendre le comportement de contrôles déjà existants. Par exemple, `AutoCompleteExtender`, dont vous verrez un exemple d'utilisation plus loin dans ce chapitre, permet la mise en place d'une zone de saisie qui propose à l'utilisateur de compléter sa frappe par le biais de suggestions, au fur et à mesure de la saisie. Il étend le comportement d'une `TextBox` en passant par une méthode de Web Service pour récupérer les suggestions en fonction de ce qui est déjà saisi.

```
<asp:TextBox ID="TextBox1" runat="server" />
<atlas:AutoCompleteExtender ID="AutoComplete"
  runat="server">
  <atlas:AutoCompleteProperties TargetControlID="TextBox1"
Enabled="true" ServicePath="WebService.asmx"
ServiceMethod="GetValues" />
</atlas:AutoCompleteExtender>
```

21.5 Utilisation du contrôle UpdatePanel

Le contrôle `UpdatePanel` indique quelles sections de la page doivent être mises à jour indépendamment. Lorsqu'un événement qui doit normalement déclencher une publication est lancé, cette publication est en réalité exécutée de manière asynchrone.

Le contrôle UpdatePanel peut contenir des éléments Triggers ainsi qu'un ContentTemplate :

```
<atlas:UpdatePanel ID="UpdatePanel1" runat="server">
 <Triggers>
  …
 </Triggers>
 <ContentTemplate>
  …
 </ContentTemplate>
</atlas:UpdatePanel>
```

Le code déclaratif à l'intérieur du ContentTemplate est rafraîchi lorsque le ScriptManager gère une publication asynchrone. Les triggers déclarent quant à eux les événements et les propriétés que l'UpdatePanel doit gérer grâce aux éléments ControlValueTrigger et ControlEventTrigger.

Horloge

Vous allez maintenant réaliser une horloge en utilisant le contrôle UpdatePanel. Il s'agit d'un exemple typique proposé à ceux qui débutent en programmation JavaScript. Vous aurez ainsi l'occasion de constater à quel point il est facile de réaliser ce genre de fonctionnalité à l'aide de contrôle Atlas.

Vous allez utiliser pour cela le contrôle Atlas TimerControl, qui est une sorte de compteur à rebours. Il possède une propriété Interval qui spécifie un intervalle de temps en millisecondes et un événement Tick qui se produit après chaque intervalle.

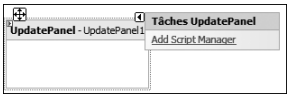
1. Pour commencer, déposez un contrôle UpdatePanel dans votre page à partir de la boîte à outils Atlas que vous avez créée précédemment.

Remarque

Importation d'assembly lors du dépôt d'un contrôle

Si vous étiez parti d'un site web normal, le dépôt du contrôle UpdatePanel aurait eu pour effet d'importer directement l'assembly *Microsoft.Web.Atlas.dll* puisqu'il est nécessaire à son fonctionnement et qu'ils ont été associés au moment de l'ajout dans la boîte à outils.

2. Affichez le smart tag puis cliquez sur le lien *Add Script Manager* pour ajouter un contrôle ScriptManager.

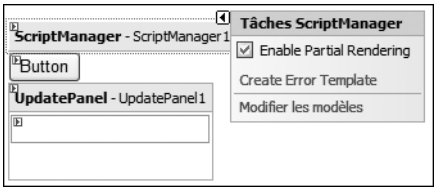


◄ **Figure 21-11** : Smart tag du contrôle ScriptManager

3 À l'intérieur de la balise `ContentTemplate` de l'`UpdatePanel`, insérez le code serveur qui affiche l'heure courante au format hh:mm:ss.

```
<ContentTemplate>
    <%=DateTime.Now.ToLongTimeString()%>
</ContentTemplate>
```

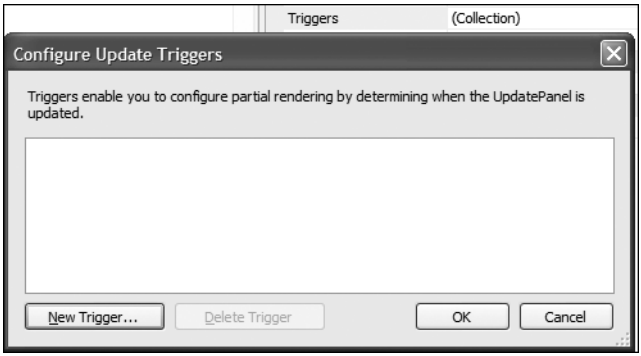
4 Activez ensuite la propriété `EnablePartialRendering` du contrôle `ScriptManager`.



◄ **Figure 21-12** : Activation de la propriété EnablePartialRendering d'un contrôle ScriptManager

5 Déposez un contrôle `TimerControl` dans la page, en dehors de l'`UpdatePanel` et spécifiez un intervalle de 1 000 millisecondes, soit 1 seconde, dans sa propriété `Interval`.

6 En mode Design, sélectionnez l'`UpdatePanel` puis cliquez sur le bouton de la propriété `Triggers` pour ouvrir la boîte de dialogue qui gère les déclencheurs.



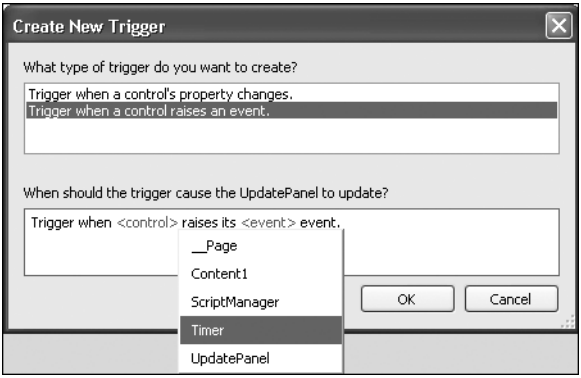
▲ **Figure 21-13** : Boîte de dialogue de gestion des déclencheurs

7 Un bouton **New Trigger** ouvre une nouvelle boîte de dialogue qui permet d'ajouter facilement un déclencheur. Sélectionnez d'abord le type de

déclencheur. Vous pouvez le baser sur le changement d'une propriété ou sur le déclenchement d'un événement d'un contrôle situé en dehors de l'`UpdatePanel`.

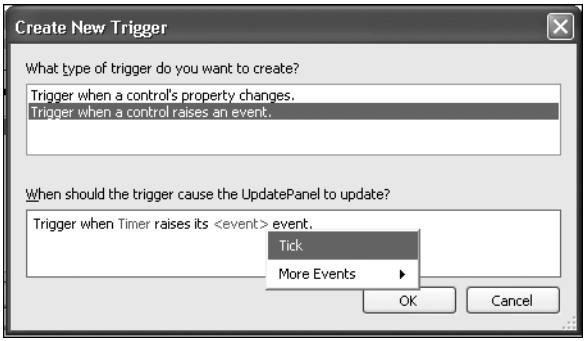
8 Ajoutez un trigger qui gère l'événement `Tick` du contrôle `Timer`. Pour cela, vous disposez de deux liens situés à l'intérieur d'une phrase qui permet de comprendre quel type de trigger va être ajouté.

Le premier propose les contrôles possibles :



▲ **Figure 21-14 :** Contrôles disponibles pour le trigger

Le second propose les événements associés au contrôle sélectionné :



▲ **Figure 21-15 :** Événements associés au contrôle sélectionné

Cela a pour effet d'ajouter un trigger qui gère l'événement `Tick` du `Timer` dans la collection des triggers de l'`UpdatePanel`.

Au final, voici le code que vous obtenez :

```
<atlas:ScriptManager ID="ScriptManager" runat="server"
➥ EnablePartialRendering="true" />
<atlas:UpdatePanel ID="UpdatePanel" runat="server">
    <ContentTemplate>
       <%=DateTime.Now.ToLongTimeString()%>
    </ContentTemplate>
    <Triggers>
       <atlas:ControlEventTrigger ControlID="Timer"
    EventName="Tick" />
    </Triggers>
</atlas:UpdatePanel>
<atlas:TimerControl ID="Timer" runat="server"
Enabled="true" Interval="1000" />
```

L'affichage de la page web permet de voir que l'heure se met bien à jour toutes les secondes.

◀ **Figure 21-16 :** Affichage de l'heure

21.6 Utilisation du contrôle AutoCompleteExtender

Passez ensuite à la création de la `TextBox` avec suggestions automatiques en fonction de la saisie.

1. Ajoutez tout d'abord un contrôle `ScriptManager` et le contrôle `TexBox` qui va récupérer la saisie de l'utilisateur.
2. Déclarez ensuite un contrôle `AutoCompleteExtender`. Précisez qu'il s'applique à la `TextBox` et qu'il utilise la méthode `GetContacts` du service web *GetContacts.asmx*.

   Votre code doit ressembler à ceci :

```
<atlas:ScriptManager ID="ScriptManager1"
EnablePartialRendering="true" runat="server" />

Nom :
<asp:TextBox ID="tbName" runat="server" />

<atlas:AutoCompleteExtender ID="ac1" runat="server">
  <atlas:AutoCompleteProperties Enabled="true"
  MinimumPrefixLength="1"
  ServicePath="GetContacts.asmx" TargetControlID="tbName"
  ➥ ServiceMethod="GetContacts" />
</atlas:AutoCompleteExtender>
```

Remarque

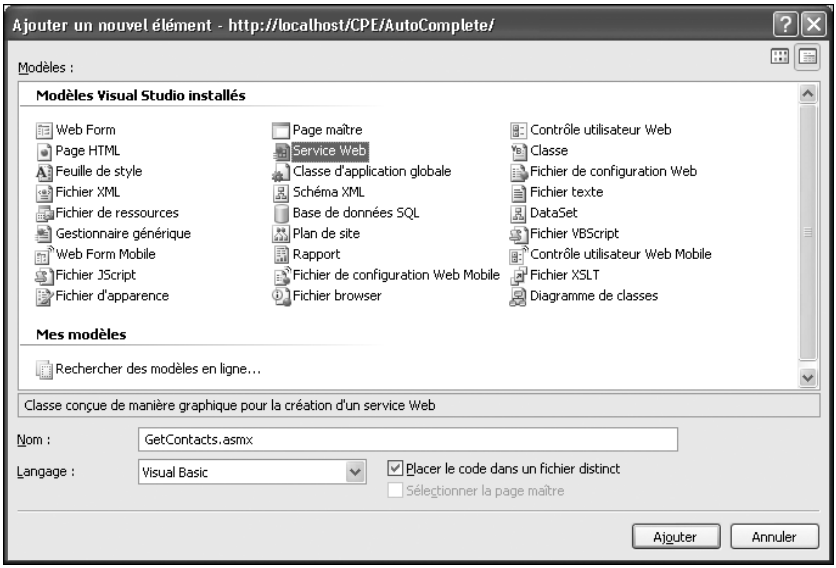
Spécification du nombre minimum de lettres à saisir

La méthode `GetContacts` du service web sera appelée à chaque lettre saisie dans le contrôle `TextBox` dès que le nombre de lettres a atteint la limite fixée dans la propriété `MinimumPrefixLength` de l'`AutoCompleteExtender`.

Il ne reste plus qu'à créer le service web et sa méthode `GetContacts`.

Création du service web

1 Ajoutez un nouvel élément *GetContacts.asmx* de type service web à la racine de votre projet.



▲ **Figure 21-17 :** Ajout d'un service web

2 Le code de ce service web se trouvera dans le dossier *App_Code*. Ajoutez-lui une méthode `GetContacts` au format suivant :

```
<WebMethod()> _
Public Function GetContacts(ByVal prefixText As String, _
  ByVal count As String) As String()
End Function
```

Remarque

Syntaxe de la méthode de récupération des suggestions

La méthode du service web de récupération des suggestions doit être exactement celle-ci, avec les mêmes noms de paramètre.

3. En ce qui concerne le code de récupération des noms qui correspondent aux lettres déjà saisies, vous pouvez utiliser une liste générique d'objets `String`, par exemple la liste de vos contacts, et un prédicat qui vérifie si le début de la chaîne correspond au paramètre `prefixText`. Dans cet exemple, déclarez une liste manuellement :

```
Dim l As New List(Of String)
Dim names As String() = _
  {"Adeline", "Antoine", "Aurélien", "Grégory",…}
l.AddRange(names)
```

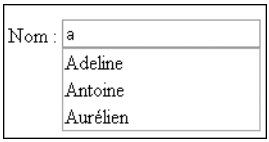
4. Créez ensuite une classe qui prend en paramètre de constructeur le préfixe à vérifier et déclarez une méthode `IsBeginningOf` qui vérifie si une chaîne passée en paramètre commence par ce préfixe :

```
Public Class StringFilter
    Private strPrefix As String = ""

    Public Sub New(ByVal value As String)
        strPrefix = value
    End Sub

    Public Function IsBeginningOf(ByVal _name As String) _
    As Boolean
        Return _name.StartsWith(strPrefix, _
      StringComparison.InvariantCultureIgnoreCase)
    End Function

End Class
```

5. Vous pouvez ensuite utiliser la fonction `FindAll` d'une liste générique qui prend en paramètre un prédicat. En lui passant un prédicat qui vérifie si une chaîne commence par le préfixe, vous obtiendrez donc seulement les contacts qui commencent par les lettres saisies. Utilisez alors la méthode `ToArray` pour renvoyer un tableau de `String`.

```
Dim p As System.Predicate(Of String) = _
  New Predicate(Of String)(AddressOf _
    New StringFilter(prefixText).IsBeginningOf)
Return l.FindAll(p).ToArray()
```

6 Vous pouvez tester votre zone de saisie en entrant quelques lettres. Une liste de tous les contacts correspondants s'affichera alors.



◄ **Figure 21-18 :** Suggestion des contacts commençant par la lettre "a"

21.7 Check-list

Dans ce chapitre, vous avez appris à :

- installer Atlas ;
- ajouter des contrôles Atlas à la boîte à outils ;
- utiliser le contrôle `UpdatePanel` ;
- utiliser le contrôle `AutoCompleteExtender`.

Réalisation d'un questionnaire à l'aide d'un contrôle Wizard

Lorsque vous réalisez une application, votre souci principal doit être de répondre aux besoins de l'utilisateur en lui fournissant des fonctionnalités les plus simples possibles. On parle dans ce cas d'expérience utilisateur.

Si vous devez afficher une quantité importante de données sur un même écran, cela peut vite devenir assez incompréhensible pour l'utilisateur. Afin de l'assister dans la saisie d'informations, il est intéressant de décomposer une opération compliquée en plusieurs étapes simples et détaillées.

De nombreuses fonctionnalités nécessitent d'être décomposées en plusieurs étapes, que ce soit pour des raisons de décomposition logique (si l'on doit procéder à une étape avant une autre par exemple) ou d'organisation.

Dans cet exemple, vous allez développer un questionnaire à l'aide du contrôle `Wizard`. Vous aurez par ce biais l'occasion d'apprendre à améliorer la saisie de formulaire pour une meilleure expérience utilisateur.

22.1 Classes et espaces de noms utilisés

L'espace de noms `System.Web.UI.WebControls` s'est considérablement enrichi avec l'ASP .NET 2.0. De nombreux nouveaux contrôles sont en effet disponibles et l'architecture a été repensée et réorganisée afin de fournir par exemple des classes de base dont peuvent hériter d'autres contrôles enfants ou des interfaces à implémenter pour ajouter des fonctionnalités intéressantes.

`CompositeControl`, dont hérite le contrôle `Wizard`, est une classe abstraite qui sert à développer des contrôles personnalisés englobant des contrôles enfants ou utilisant les fonctionnalités d'autres contrôles, tels que les contrôles `Login` ou `SiteMapPath`.

22.2 Contrôle Wizard

Le Framework ASP .NET 1.1 ne disposait pas de contrôle permettant de récolter des informations en plusieurs étapes. Le souci majeur était notamment la persistance des données ainsi qu'une navigation intuitive. Il s'agissait de fournir un contrôle simple à configurer et qui autorisait en même temps une personnalisation totale de chaque élément ainsi qu'un ensemble d'événements permettant de gérer les enchaînements d'écrans et la récupération des valeurs saisies. Le nouveau contrôle serveur `Wizard` répond à la plupart de ces problématiques.

Le contrôle `Wizard` fournit une navigation au travers d'une série d'étapes qui récupèrent l'information de l'utilisateur final.

Le contrôle `Wizard` peut être constitué des éléments suivants :

- une collection d'étapes (Wizard Steps) qui contient l'interface utilisateur pour chacune d'entre elles, comme défini par le développeur de pages ;
- une zone de navigation (Navigation Area) intégrée qui détermine les boutons appropriés à afficher en fonction de l'étape à laquelle le Wizard se trouve ;
- un en-tête (Header) qui peut être personnalisé pour afficher des informations spécifiques à l'étape où l'utilisateur se trouve.
- une barre latérale (Side Bar) qui peut être utilisée pour naviguer rapidement à travers les différentes étapes du contrôle.

Wizard Steps

L'élément enfant le plus important est bien sûr la collection des étapes. Chaque étape dans le contrôle `Wizard` a une propriété `StepType` qui détermine le type de fonctionnalité de navigation qu'offre l'étape. Si vous ne spécifiez pas de valeur pour la propriété `StepType`, la valeur par défaut est `Auto`.

Le tableau suivant répertorie les différentes valeurs possibles de la propriété `StepType` :

Différentes valeurs disponibles pour l'énumération WizardStepType	
Type d'étape	**Description**
`WizardStepType.Start`	Il s'agit de la première étape. Le bouton **Précédent** n'est donc pas affiché.
`WizardStepType.Step`	Une étape standard située entre la première et la dernière. Les boutons **Précédent** et **Suivant** permettent de naviguer entre les étapes.
`WizardStepType.Auto`	Le type de cette étape est déterminé selon l'ordre dans lequel l'étape est déclarée dans le code de la page.
`WizardStepType.Finish`	C'est la dernière étape de saisie où l'on peut récupérer les données de l'utilisateur. Le bouton **Terminer** apparaît.
`WizardStepType.Complete`	C'est la toute dernière étape. Aucun bouton n'est présent. Vous pouvez vous en servir pour afficher un résumé des étapes par exemple.

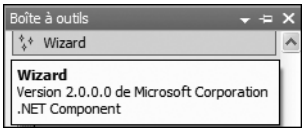
Remarque

Types d'étapes automatiques

Si vous utilisez des étapes de type `Auto`, la première sera considérée comme une étape de démarrage (Start Step) et la dernière comme une étape de fin (Finish Step). Pour ajouter une étape de type `Complete`, vous devez la déclarer explicitement.

22.3 Réalisation du questionnaire

Passons maintenant à la réalisation du questionnaire. Tout d'abord, vous allez sélectionner un contrôle `Wizard` dans la boîte à outils et le déposer dans la page.



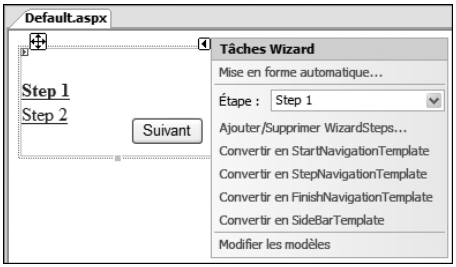
◄ **Figure 22-1 :** Contrôle Wizard dans la boîte à outils de Visual Studio

Par défaut, le Wizard contient deux étapes standard vides, ce qui permet d'avoir un premier aperçu :

```
<asp:Wizard ID="Wizard1" Runat="server">
    <WizardSteps>
        <asp:WizardStep Runat="server" Title="Step 1">
        </asp:WizardStep>
        <asp:WizardStep Runat="server" Title="Step 2">
        </asp:WizardStep>
    </WizardSteps>
</asp:Wizard>
```

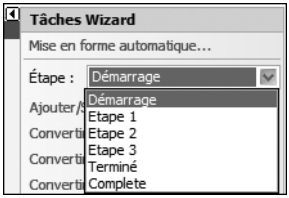
Ajout des étapes

Le smart tag s'affiche avec les différentes options de configuration possibles.



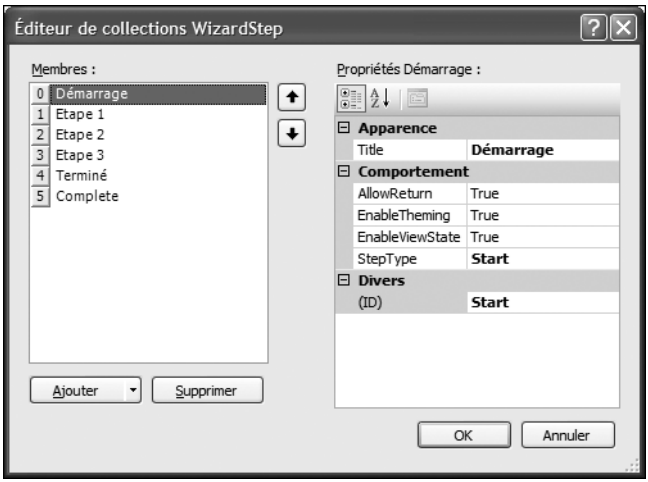
◄ **Figure 22-2 :** Smart tag d'un contrôle Wizard

Une liste permet notamment de sélectionner l'étape que vous voulez modifier et des liens d'action permettent de changer facilement le type de l'étape en cours d'édition.



◄ **Figure 22-3** : Navigation entre les étapes grâce à une liste du smart tag

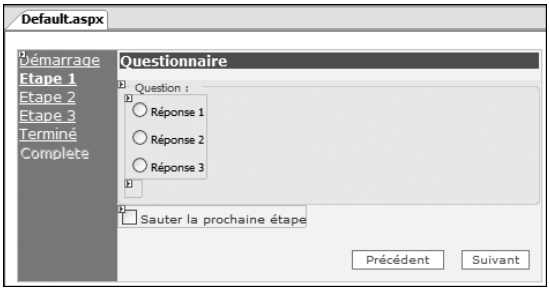
Un lien d'action permet tout d'abord d'ajouter et de supprimer facilement des étapes à l'aide d'un éditeur. Vous pouvez ainsi définir les étapes, les organiser et modifier leurs propriétés comme leur identifiant, leur titre ou encore leur type.



▲ **Figure 22-4** : Éditeur de collections WizardStep

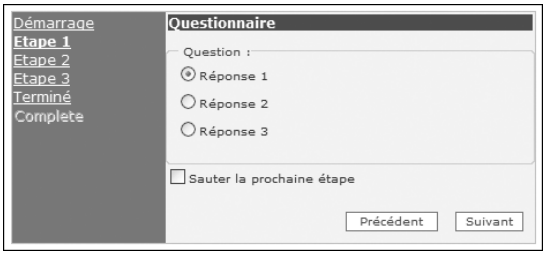
Il suffit de cliquer sur l'étape pour pouvoir éditer son contenu. Vous pouvez alors écrire du texte ou même déposer n'importe quel contrôle à partir de la boîte à outils.

Vous pouvez ainsi obtenir rapidement une étape qui ressemble à celle-ci dans Visual Studio :



▲ **Figure 22-5 :** Étape 1 du Wizard dans l'éditeur en mode Design

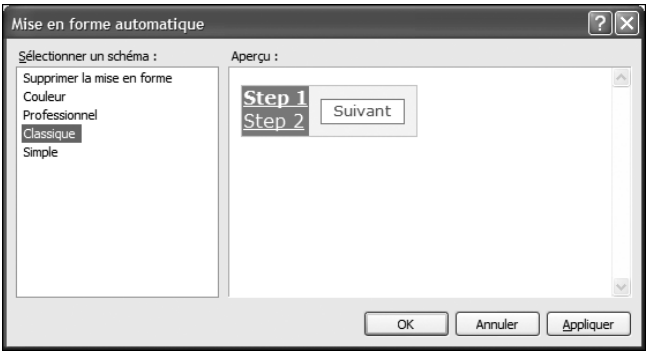
Au final, le questionnaire ressemblera à ce vous pouvez visualiser en mode Design.



▲ **Figure 22-6 :** Étape 1 du questionnaire

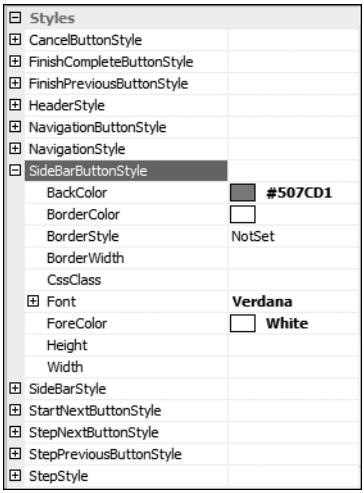
Mise en forme du Wizard

Afin de mettre en forme le questionnaire avec un style par défaut, vous pouvez sélectionner une mise en forme automatique à l'aide d'un Assistant qui affiche certaines apparences possibles et qui définit les propriétés de style du contrôle `Wizard`.



▲ **Figure 22-7 :** Assistant de mise en forme automatique

Le contrôle `Wizard` possède plusieurs propriétés de style correspondant à chacun de ces composants.



◄ **Figure 22-8 :** Propriétés de style du contrôle Wizard

Le fait de sélectionner un style prédéfini à l'aide de l'Assistant de mise en forme automatique détermine ces propriétés de style. Un contrôle `Wizard` vide auquel est associé le style *classique* ressemblerait ainsi à ceci :

```
<asp:Wizard ID="Wizard1" runat="server"
    ActiveStepIndex="0" Height="400px" Width="600px"
    BackColor="#EFF3FB" BorderColor="#B5C7DE"
    BorderWidth="1px" Font-Names="Verdana"
    Font-Size="0.8em">
    <StepStyle Font-Size="0.8em" ForeColor="#333333" />
    <SideBarStyle BackColor="#507CD1" Font-Size="0.9em"
        VerticalAlign="Top" />
    <NavigationButtonStyle BackColor="White"
        BorderColor="#507CD1" BorderStyle="Solid"
        BorderWidth="1px" Font-Names="Verdana"
        Font-Size="0.8em" ForeColor="#284E98" />
    <SideBarButtonStyle BackColor="#507CD1"
        Font-Names="Verdana" ForeColor="White" />
    <HeaderStyle BackColor="#284E98" BorderColor="#EFF3FB"
        BorderStyle="Solid" BorderWidth="2px"
        Font-Bold="True" Font-Size="0.9em"
        ForeColor="White" HorizontalAlign="Center" />
</asp:Wizard>
```

Étant donné le nombre important de propriétés de style et de leurs attributs, cela peut vite devenir complexe et difficile à maintenir. Il est donc conseillé de déclarer ces propriétés dans un fichier d'apparence (*.skin*).

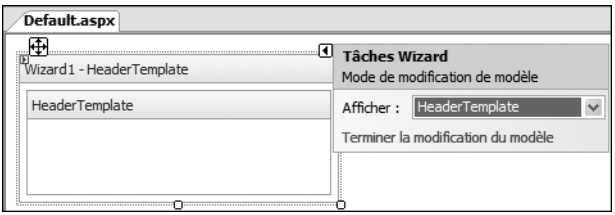
Lisez à ce sujet le chapitre ***Site web avec sélecteur de thèmes.***

Vous pourrez ainsi appliquer facilement la même apparence à tous vos contrôles `Wizard` ou en créer d'autres.

Personnalisation du Wizard

Le contrôle `Wizard` contient certains éléments configurables comme le `HeaderTemplate`. Ce modèle est l'endroit dans lequel vous pouvez spécifier ce qui va être affiché dans l'en-tête du Wizard. D'autres modèles, comme le `StartNavigationTemplate`, le `StepNavigationTemplate` ou le `FinishNavigationTemplate`, permettent de personnaliser l'apparence des différentes étapes.

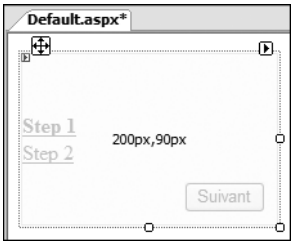
Vous pouvez éditer ces modèles en mode Design à l'aide de la balise active et d'une liste qui affiche les différents templates et permet de basculer entre eux.



▲ **Figure 22-9 :** Édition du HeaderTemplate

Certains sont prédéfinis et doivent répondre à une structure précise si vous voulez les redéfinir. Il en est ainsi du `SideBarTemplate`, chargé d'afficher les différentes étapes dans une zone latérale et de distinguer l'étape en cours.

Une autre fonctionnalité intéressante est la possibilité de modifier la taille du contrôle à l'aide des poignées de redimensionnement prévues à cet effet, comme vous le feriez pour un formulaire Windows.



◄ **Figure 22-10 :** Redimensionnement du contrôle Wizard

22.4 Navigation au sein du contrôle Wizard

Le contrôle `Wizard` étant essentiellement destiné à afficher des étapes successives, de nombreuses méthodes, événements et propriétés peuvent vous aider à personnaliser la navigation et l'enchaînement des étapes.

Navigation linéaire

Si vous souhaitez enregistrer les informations saisies à chaque étape, sans laisser la possibilité de revenir en arrière, vous devez affecter la valeur `false` à la propriété `AllowReturn` de l'étape. Vous pouvez alors utiliser le gestionnaire d'événements `NextButtonClick` afin de réaliser l'action en fonction de l'étape qu'il vient de valider.

Si vous voulez passer une étape, selon ce que l'utilisateur a saisi, vous pouvez utiliser la méthode `MoveTo` ou la propriété `ActiveStepIndex` afin de modifier dynamiquement l'étape suivante dans la navigation.

Navigation personnalisée

S'il faut sauter des étapes selon les choix de l'utilisateur ou les informations saisies, vous pouvez effectuer ce genre de navigation personnalisée à l'aide d'un gestionnaire d'événements de l'événement `NextButtonClick` à l'intérieur duquel vous testerez si les conditions sont remplies pour passer une étape et indiquerez ensuite l'index de l'étape suivante à atteindre.

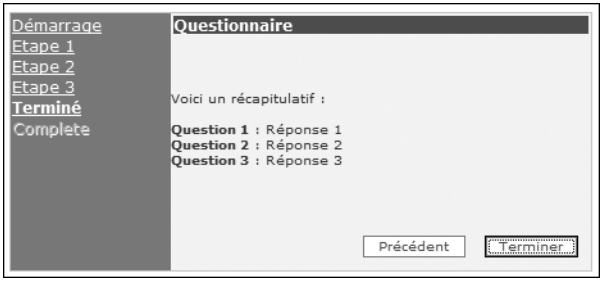
Voici par exemple un gestionnaire d'événements qui teste à la fin de l'étape 1 si une case à cocher a été sélectionnée et passe à l'étape 3 si c'est le cas.

```
Protected Sub Questionnaire_NextButtonClick( _
    ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls. _
    WizardNavigationEventArgs)_
    Handles Questionnaire.NextButtonClick
        If Questionnaire.ActiveStepIndex = 1 Then
            If cbSkipStep.Checked Then
                Questionnaire.ActiveStepIndex = 3
            Else
                Questionnaire.ActiveStepIndex = 2
            End If
        End If
    End Sub
```

Récapitulatif des résultats

Après avoir créé plusieurs étapes, vous pouvez écrire un peu de code pour afficher les choix de l'utilisateur :

```
<asp:WizardStep runat="server" StepType="Finish"
    Title="Terminé" ID="Finish">
          Voici un récapitulatif :<br /><br />
          <b>Question 1</b> :
          <%=Reponse1.SelectedItem.Text%><br />
          <b>Question 2</b> :
          <%=Reponse2.SelectedItem.Text%><br />
          <b>Question 3</b> : <%=Reponse3.Text%>
</asp:WizardStep>
```



▲ **Figure 22-11 :** Écran récapitulatif des résultats de l'étape Finish

22.5 Amélioration de l'expérience utilisateur

Souvent, les développeurs mettent de côté l'expérience utilisateur, en se souciant peu de fournir un outil adapté à ceux qui vont se servir de leurs applications. À leurs yeux, cette étape importante d'amélioration de l'expérience utilisateur n'est pas prioritaire, au point d'en être parfois occultée. Les développeurs ont, en effet, parfois du mal à se mettre à la place des utilisateurs et ce qui paraît évident aux premiers ne l'est pas forcément pour les seconds.

Vous allez maintenant découvrir quelques astuces simples et bien utiles pour améliorer la saisie de formulaires web. Il s'agit de nouvelles fonctionnalités associées aux contrôles et aux pages ASP .NET qui apparaissent avec le Framework 2.0.

Toutes ces propriétés n'étaient pas forcément présentes dans le Framework 1.1 et le retour d'expérience des développeurs a fait qu'elles ont été intégrées dans la nouvelle version et surtout qu'elles ont été pensées afin d'être facilement configurables.

Vous n'aurez ainsi plus d'excuses pour ne pas fournir des interfaces "user friendly". Voici à présent les différents domaines que vous devez intégrer afin de développer des formulaires qui permettent de saisir rapidement de l'information et de guider l'utilisateur pas à pas.

Contrôles de validation

Validation Les développeurs sont souvent amenés à valider une donnée saisie par l'utilisateur avant de l'enregistrer ou de déclencher une action. Les contrôles de validation, autrement appelés "validateurs", fournissent un moyen simple et efficace de vérifier la validité d'une entrée et, si nécessaire, de renvoyer des messages d'erreur explicites.

Certaines propriétés sont communes aux différents types de validateurs :

Propriétés communes aux contrôles validateurs	
Propriété	**Description**
`ControlToValidate`	L'identifiant du contrôle à valider.
`ValidationGroup`	Le groupe de validation associé. Si vous avez des zones différentes à vérifier, cela permet de séparer leur validation.
`Text`	Le texte à afficher à l'endroit où le validateur est inséré. Par exemple, le caractère "*" pour une donnée obligatoire.
`ErrorMessage`	Un message plus détaillé qui apparaît dans le résumé des erreurs rencontrées.

La validation des champs d'un formulaire intervient lorsque l'utilisateur déclenche un événement dont le contrôle associé possède une propriété `CausesValidation` avec la valeur `true` et qui est relié au groupe de validation.

Voici quelques exemples de contrôles de validation souvent utilisés pour vérifier la saisie de l'utilisateur et afficher des messages d'erreur le cas échéant.

RequiredFieldValidator

RequiredFieldValidator Le contrôle `RequiredFieldValidator` permet de vérifier qu'une zone de saisie a bien été remplie et ainsi de rendre cette donnée obligatoire :

```
<asp:RequiredFieldValidator id="rfvEmail" runat="server"
    ControlToValidate="Email"
    ErrorMessage="Le champ 'E-mail' est obligatoire.">*
</asp:RequiredFieldValidator>
```

RegularExpressionValidator

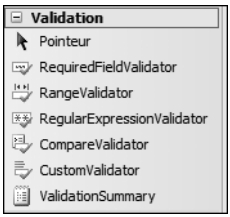
RegularExpressionValidator Le contrôle `RegularExpressionValidator` sert à contrôler le format d'une donnée. Vous pouvez ainsi vérifier qu'une adresse e-mail a bien été saisie sous la forme *nom@domaine.ext* :

```
<asp:RegularExpressionValidator ID="revEmail"
    runat="server" ControlToValidate="Email"
    ErrorMessage="Le champ 'E-mail' n'a pas le bon format."
    ValidationExpression=
        "\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">!
</asp:RegularExpressionValidator>
```

ValidationSummary

ValidationSummary Le contrôle `ValidationSummary` permet de récupérer l'ensemble des erreurs survenues et de les afficher dans un résumé à un seul endroit. Vous allez en insérer un dans la page maître afin que chaque page de données puisse s'en servir. De ce fait, vous le configurerez pour que le message soit renvoyé sous la forme d'un message d'alerte :

```
<asp:ValidationSummary ID="vs" runat="server"
DisplayMode="List" ShowMessageBox="True" ShowSummary="False">
</asp:ValidationSummary>
```



◄ **Figure 22-12 :** Onglet Validation de la boîte à outils

Propriétés des contrôles serveurs relatives à l'expérience utilisateur

Certaines propriétés des principaux contrôles serveurs que vous utilisez régulièrement dans un formulaire peuvent vous aider à améliorer l'expérience utilisateur. En voici quelques exemples avec leur description et les avantages que vous pouvez en tirer.

Bouton par défaut

La propriété `DefaultButton` du formulaire de la page ou d'un contrôle `Panel` permet de spécifier le bouton d'action par défaut, ce qui aura pour effet de déclencher un clic sur ce bouton lorsque l'utilisateur appuiera sur la touche Entrée.

La propriété `UseSubmitBehavior` d'un bouton est équivalente : elle le transforme en un bouton de type `Submit`.

Focus

La propriété `DefaultFocus` de la page permet d'indiquer l'identifiant du contrôle qui recevra le focus par défaut, c'est-à-dire celui qui sera prêt à être saisi. Vous ferez ainsi gagner du temps à l'utilisateur s'il doit régulièrement intervenir sur des formulaires (ajout rapide, modification…). Il prendra en effet rapidement l'habitude des étapes par lesquelles il doit passer pour réaliser une action et le fait de pouvoir saisir des informations sans se servir de la souris est appréciable.

La propriété `SetFocusOnError` d'un validateur permet, quant à elle, de placer le focus sur le contrôle si une erreur de validation est détectée afin de pouvoir la corriger immédiatement.

La page contient également une méthode `SetFocus`, qui prend en paramètre un identifiant de contrôle.

Propriétés d'accessibilité

Deux propriétés appartiennent à la catégorie *Accessibilité* de chaque contrôle serveur web.

◄ **Figure 22-13 :** Catégorie Accessibilité d'un contrôle

La propriété `AccessKey` permet de spécifier une lettre de raccourci. Il suffit ensuite d'appuyer la combinaison Alt+Lettre de raccourci pour accéder au contrôle.

La propriété `TabIndex` sert à indiquer l'ordre de tabulation. Vous pouvez ainsi spécifier l'ordre dans lequel les contrôles s'enchaînent lors d'un appui sur la touche Tab en incrémentant l'index dans les propriétés `TabIndex` de chaque contrôle "saisissable" de la page. Il est ainsi inutile de suivre l'ordre d'apparition des contrôles dans la page.

22.6 Check-list

L'utilisateur final doit rester au centre des préoccupations du développeur. De nombreux nouveaux contrôles et améliorations permettent de fournir des fonctionnalités abouties.

Le contrôle serveur `Wizard` est extrêmement utile pour décomposer la saisie d'informations réalisée par l'utilisateur.

Dans ce chapitre, vous avez découvert les fonctionnalités suivantes :

- la définition des différentes étapes du Wizard ;
- la personnalisation du Wizard ;
- les propriétés de style ;
- la gestion des événements ;
- la récupération de la saisie de l'utilisateur ;
- l'amélioration de l'expérience utilisateur dans la saisie de formulaire.

Création d'un contrôle serveur personnalisé

Le Framework ASP .NET met à disposition un ensemble complet de contrôles serveurs web dont le but est de faciliter le développement de pages web. La plupart des balises HTML ont en effet leur équivalent en tant que contrôle serveur web.

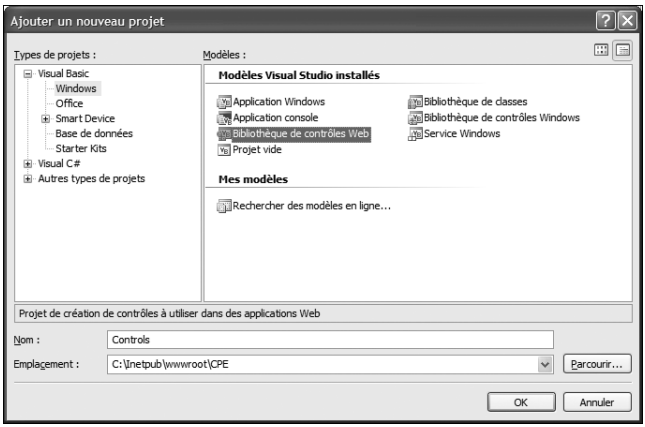
Certains autres contrôles sont destinés à encapsuler des fonctionnalités assez complexes. Une multitude de nouveaux contrôles font ainsi leur apparition avec la version 2.0 du Framework .NET. Le modèle de programmation a également été revu. Il propose des interfaces et des classes qui permettent de développer facilement des contrôles serveurs personnalisés.

De nouveaux espaces de noms et de nouvelles classes ont par exemple été créés pour séparer les différentes fonctionnalités auxquelles on fait appel lors du développement de contrôles comme le design dans Visual Studio ou encore le rendu dans le navigateur.

Dans ce chapitre, vous allez apprendre à développer un contrôle serveur composite, c'est-à-dire un contrôle qui en contient d'autres pour en fournir un seul un peu plus évolué. Notre exemple sera celui d'un contrôle qui permet de saisir facilement une date. Vous aurez l'occasion de découvrir en même temps des techniques de programmation assez avancées qui faciliteront au final l'utilisation de ce contrôle par les développeurs, comme la création d'un smart tag associé ou l'utilisation d'Éditeurs de propriétés.

23.1 Création de la librairie de contrôles

La première étape consiste à créer la librairie qui va contenir le contrôle. Pour cela, ajoutez tout d'abord à votre solution un projet de type *Bibliothèque de contrôles web* que vous appellerez `Controls`.



▲ **Figure 23-1 :** Ajout d'un projet Bibliothèque de contrôles web

Vous pouvez ensuite afficher les propriétés de votre librairie à l'aide du formulaire *MyProject*. Pour l'ouvrir, vous pouvez soit double-cliquer sur le dossier *MyProject*, soit cliquer dessus du bouton droit puis sélectionner la commande **Ouvrir**, soit cliquer du bouton droit sur le projet puis sélectionner la commande **Propriétés**.

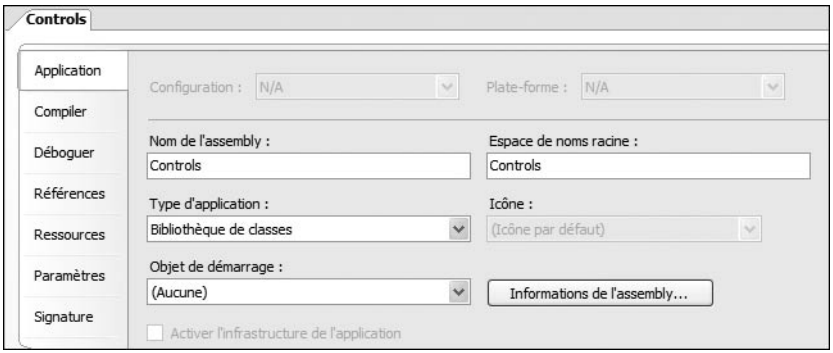
◄ **Figure 23-2 :** Commande Ouvrir du dossier MyProject

L'interface se compose de plusieurs onglets qui vous aideront à personnaliser la librairie selon vos besoins. Voici un aperçu des principaux onglets et des options qu'ils permettent de configurer.

Configuration de la librairie

Onglet Application

L'onglet **Application** permet notamment de gérer le nom de l'assembly qui sera généré ainsi que l'espace de noms racine.

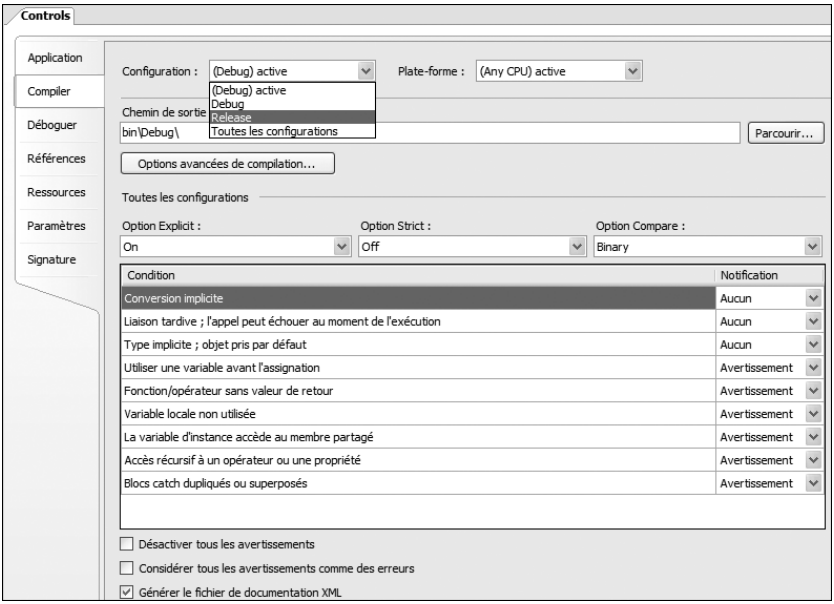


▲ **Figure 23-3 :** Onglet Application

Onglet Compiler

L'onglet **Compiler** sert à gérer les options de compilation. Deux configurations de génération sont disponibles par défaut :

- `Debug`, pour générer l'assembly avec un fichier de débogage (*.pdb*) associé ;
- `Release`, pour une version sans fichier de débogage et donc plus performante.



▲ **Figure 23-4 :** Onglet Compiler

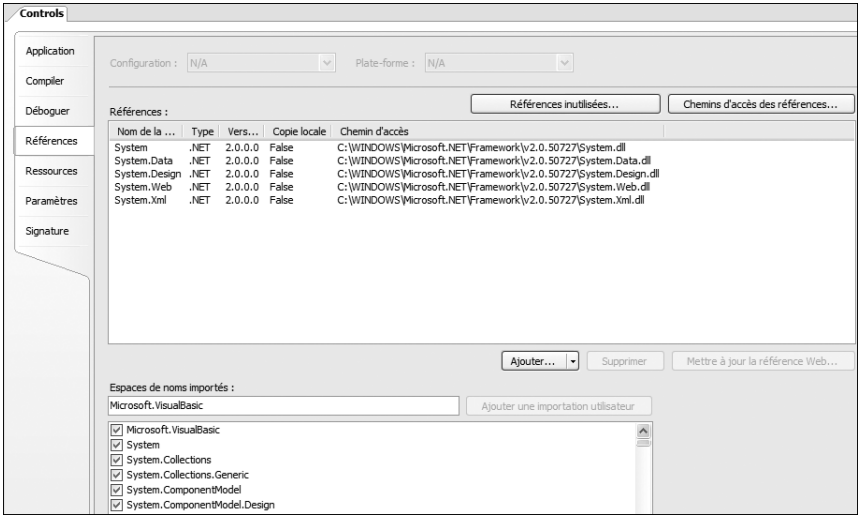
Onglet Références

L'onglet **Références** permet de cocher les espaces de noms que vous voulez importer par défaut pour ne pas avoir à introduire d'instruction `Imports` en haut de vos pages.

Remarque

Proposition des classes

Si vous oubliez d'importer un espace de noms qui contient une classe que vous voulez utiliser ou que celle-ci est mal orthographiée, l'éditeur de code de Visual Basic vous proposera automatiquement, à l'aide d'une balise, les classes qui peuvent correspondre à celle que vous désirez.

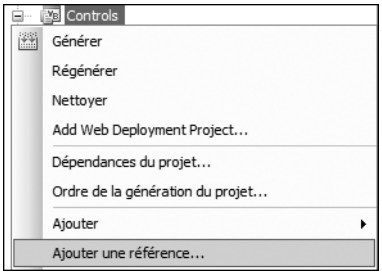


▲ **Figure 23-5 :** Onglet Références

Vous devez également faire référence à l'assembly *System.Design.dll* pour avoir à disposition l'espace de noms `System.Web.UI.Design` et ses classes associées. Pour cela, cliquez sur le bouton **Ajouter** puis sur la commande **Référence**.

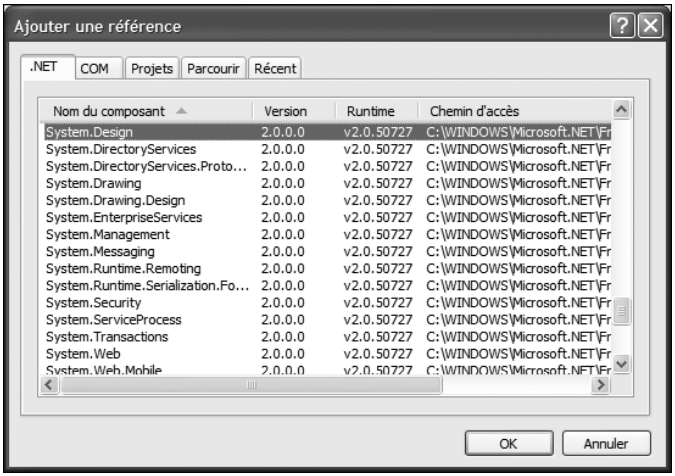
◄ **Figure 23-6 :** Bouton Ajouter/Référence

La même commande est accessible à partir de l'Explorateur de solutions : cliquez du bouton droit sur le projet et sélectionnez la commande **Ajouter une référence**.



◄ **Figure 23-7 :** Commande Ajouter une référence

Une boîte de dialogue de sélection avec plusieurs onglets vous propose alors les assemblys auxquels vous pouvez faire référence.



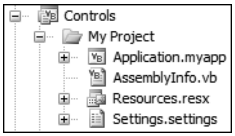
▲ **Figure 23-8 :** Boîte de dialogue Ajouter une référence

Fichier AssemblyInfo.vb

Il est stocké dans le dossier *MyProject*. Pour y accéder, affichez tous les fichiers du projet à l'aide du bouton situé dans la barre du haut, disponible lorsque le projet est sélectionné.

◄ **Figure 23-9 :** Bouton Afficher tous les fichiers

Ce dossier contient les designers qui permettent d'afficher les onglets et les Éditeurs de ressources ou de settings.



◄ **Figure 23-10 :** Fichiers du dossier MyProject

Le fichier *AssemblyInfo.vb* permet de spécifier certaines informations sur l'assembly qui va être généré, comme le nom, l'éditeur ou la version, à l'aide de différents attributs.

Un attribut sert à définir ce que l'on appelle une métadonnée sur une certaine partie de code. Les attributs sont associés à différentes fonctionnalités :

- les attributs d'information, pour apporter des précisions sur la donnée ;
- les attributs de permission, pour définir les droits d'accès à l'élément ;

- les attributs de personnalisation, pour définir un comportement particulier ;
- les attributs système.

Ils peuvent être placés devant une classe, une méthode, un membre ou tout autre élément de code. Leur syntaxe de déclaration est toujours la même :

```
<Attribut1(), _
 Attribut2(param)> _
Class | Property | Function | …
```

L'attribut `TagPrefix`, par exemple, permet de spécifier le préfixe par défaut de la librairie de contrôle, qui sera appliqué à tous vos contrôles personnalisés, pour empêcher qu'ils prennent celui par défaut (`cc1`).

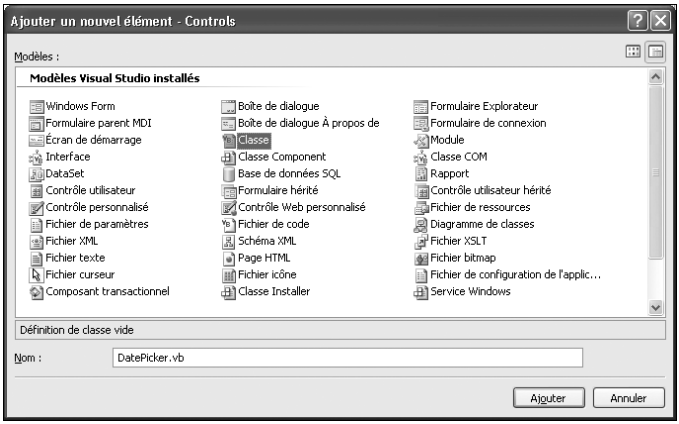
```
<assembly: TagPrefix("Namespace.MyControls", "prefix")>
```

Un contrôle déposé dans l'Éditeur ajoutera la directive `Register` et la balise du contrôle avec la syntaxe suivante :

```
<%@ Register Assembly="Controls" Namespace="Controls"
 TagPrefix="prefix" %>

<prefix:MyControl ID="MyControl" runat="server" />
```

23.2 Création du contrôle composite

Vous pouvez maintenant créer votre contrôle composite. Ajoutez donc une classe `DatePicker` à votre librairie.



▲ **Figure 23-11 :** Ajout de la classe DatePicker

Vous allez réaliser un contrôle qui permet de saisir une date. Il sera composé d'une zone de texte dans laquelle il sera possible de faire une saisie manuelle et d'une image qui affichera un calendrier à partir duquel vous pourrez sélectionner une date. Voici les étapes de développement successives décomposées en plusieurs parties pour mieux comprendre le rôle de chacune.

Héritage de la classe

Tout d'abord, votre classe doit hériter de la classe abstraite `CompositeControl` :

```
Public Class DatePicker
        Inherits CompositeControl

End Class
```

Ajout des contrôles enfants

Ajoutez ensuite les trois contrôles en tant que membres privés. Les contrôles `ImageButton` et `Calendar` sont déclarés `WithEvents` car vous allez leur associer des événements.

```
Private tbDate As TextBox
Private WithEvents imgBtnCalendar As ImageButton
Private WithEvents calendar As Calendar
```

Substituez la méthode `CreateChildControl` pour y créer les contrôles et les ajouter au contrôle composite :

```
Protected Overrides Sub CreateChildControls()
        Controls.Clear()
        tbDate = New TextBox
        tbDate.ID = "tbDate"
        Controls.Add(tbDate)

        imgBtnCalendar = New ImageButton
        imgBtnCalendar.ID = "imgBtnCalendar"
        Controls.Add(imgBtnCalendar)

        calendar = New Calendar
        calendar.ID = "calendar"
        calendar.Visible = False
        Controls.Add(calendar)
End Sub
```

Ajout des propriétés

Ajoutez les propriétés publiques que vous voulez rendre configurables par les développeurs. Les valeurs de ces propriétés seront stockées dans l'état d'affichage (`ViewState`). Voici par exemple la propriété `Text` :

```
Public Property [Text]() As String
        Get
            Dim obj As Object = ViewState("Text")
            If Not IsNothing(obj) Then
                Return CType(obj, String)
            End If
            Return ""
        End Get
        Set(ByVal Value As String)
            ViewState("Text") = Value
        End Set
End Property
```

Vous mettrez également à disposition une propriété `ImageUrl`, qui permettra d'associer l'URL de l'image à l'`ImageButton`, et une propriété `SelectedDate`, qui renverra la date sélectionnée.

Attributs de classe

Configurez à présent les attributs de la classe. Il est intéressant d'importer l'espace de noms `System.ComponentModel`, qui contient de nombreux attributs dont vous aurez besoin. Ces attributs servent à configurer différentes options. Ils nécessitent parfois certains paramètres.

```
<DefaultProperty("Text"), _
 ToolboxData( _
"<{0}:DatePicker runat=server></{0}:DatePicker>"), _
 ToolboxBitmap(GetType(DatePicker), "DatePicker.bmp"), _
 Designer(GetType(DatePickerDesigner))> _
Public Class DatePicker
```

DefaultProperty

L'attribut `DefaultProperty` permet de spécifier la propriété par défaut du contrôle.

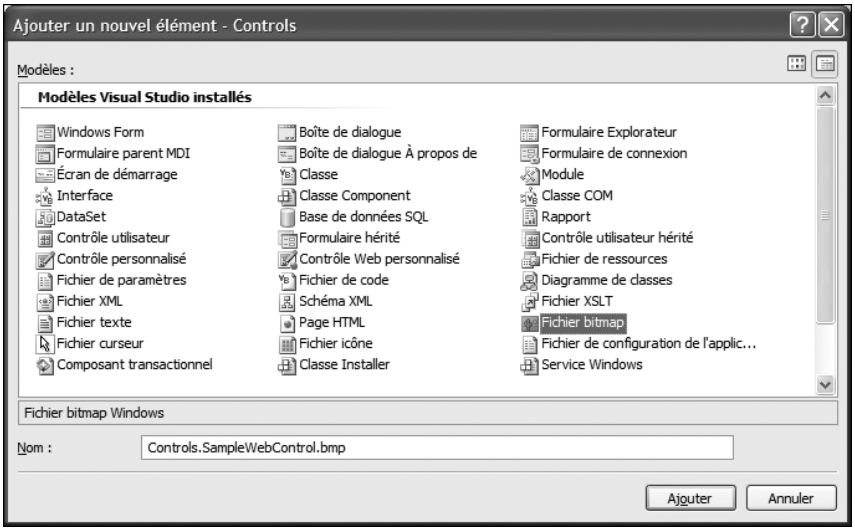
ControlValueProperty

L'attribut `ControlValueProperty` est utilisé pour déterminer qu'il s'agit de la propriété par défaut lorsque le contrôle est sélectionné en tant que `ControlParameter` d'une source de données (`DataSource`). Vous pouvez spécifier le nom, le type et la valeur par défaut.

ToolboxBitmap

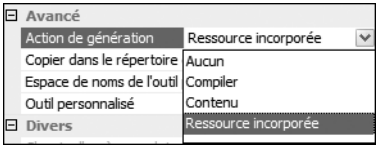
L'attribut `ToolboxBitmap` permet d'afficher une icône spécifique pour le contrôle dans la boîte à outils de Visual Studio afin de remplacer l'icône utilisée par défaut.

Pour cela, vous devez avoir à disposition une image bitmap de 16×16 pixels en 16 couleurs. Vous pouvez copier une image ou la dessiner vous-même.



▲ **Figure 23-12 :** Ajout d'une image bitmap

Placez-la dans la racine de votre projet puis déclarez-la comme ressource incorporée dans la propriété *Action de génération*.



◄ **Figure 23-13 :** Ressource incorporée

Enfin, ajoutez l'attribut devant le nom de la classe du contrôle avec, en paramètre, le type du contrôle et le nom de l'image associée.

```
<ToolboxBitmap(gettype(MyControl), "MyControl.bmp")>
```

ToolboxData

L'attribut `ToolboxData` sert à personnaliser les balises générées par Visual Studio quand vous ajoutez une instance de votre contrôle dans l'éditeur. Le {0} sera remplacé par le préfixe spécifié.

```
<ToolboxData( _
"<{0}:DatePicker runat=""server""></{0}:DatePicker>")>
```

Attributs de propriété

Les propriétés peuvent être personnalisées avec des nombreux attributs, qui ont notamment des impacts sur l'Éditeur de propriétés. Voici les attributs associés à la propriété `Text` :

```
<Description("Le texte du bouton."), _
    Category("Appearance"), _
    DefaultValue(""), _
    Browsable(True), _
    Bindable(True), _
    Localizable(True)> _
Public Property [Text]() As String
```

Le tableau suivant présente la plupart des attributs de propriété :

Attributs appliqués à des propriétés	
Attribut	**Description**
`BrowsableAttribute`	Spécifie si une propriété ou un événement doivent être affichés dans l'Explorateur de propriétés.
`CategoryAttribute`	Spécifie le nom de la catégorie dans laquelle regrouper une propriété ou un événement.
`DescriptionAttribute`	Définit un petit bloc de texte à afficher en bas de l'Explorateur de propriétés lorsque l'utilisateur sélectionne une propriété ou un événement.
`BindableAttribute`	Spécifie si une propriété est intéressante à lier.
`DefaultPropertyAttribute`	Spécifie la propriété par défaut pour le composant. Cette propriété est sélectionnée dans l'Explorateur de propriétés lorsqu'un utilisateur clique sur le contrôle.
`DefaultValueAttribute`	Affecte une valeur par défaut simple à une propriété.
`EditorAttribute`	Spécifie l'Éditeur à utiliser pour modifier une propriété.

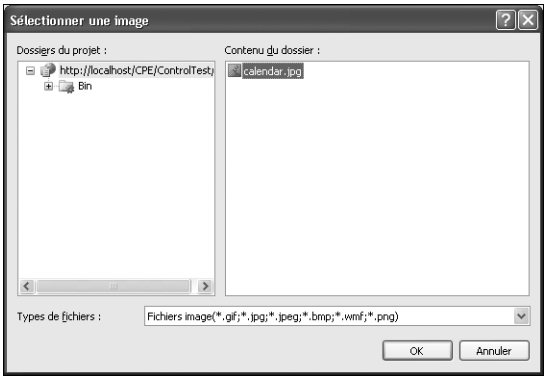
Attributs appliqués à des propriétés	
Attribut	**Description**
`LocalizableAttribute`	Spécifie qu'une propriété peut être localisée. Toutes les propriétés qui possèdent cet attribut sont automatiquement rendues persistantes dans le fichier de ressources lorsqu'un utilisateur choisit de localiser un formulaire.
`DesignerSerialization VisibilityAttribute`	Spécifie si (et comment) une propriété affichée dans l'Explorateur de propriétés doit être persistante dans le code.
`TypeConverterAttribute`	Spécifie le convertisseur de type à utiliser pour convertir le type de la propriété dans un autre type de donnée.
`DefaultEventAttribute`	Spécifie l'événement par défaut pour le composant. Il s'agit de l'événement qui est sélectionné dans l'Explorateur de propriétés lorsqu'un utilisateur clique sur le composant.

Ajout d'un Éditeur de propriétés

Pour ajouter un éditeur spécifique à la propriété `ImageUrl`, vous pouvez utiliser l'attribut `Editor` avec comme type d'éditeur un `ImageUrlEditor` :

```
EditorAttribute(GetType(
System.Web.UI.Design.ImageUrlEditor), GetType( _
System.Drawing.Design.UITypeEditor)
```

Lorsque vous utiliserez votre contrôle, l'Éditeur de propriétés vous proposera de sélectionner l'URL de l'image à partir d'une boîte de dialogue.



▲ **Figure 23-14 :** Boîte de dialogue Sélectionner une image

Remarque

Éditeurs de propriétés

Il existe de nombreux autres Éditeurs de propriétés dans l'espace de noms `System.Web.UI.Design`. Vous pouvez également développer votre propre éditeur.

Gestion du rendu du contrôle

Vous devez ensuite substituer la méthode `OnPreRender` dans laquelle vous initialiserez les propriétés par défaut de contrôles enfants, comme l'adresse de l'image.

```
Protected Overrides Sub OnPreRender(ByVal e As EventArgs)
        MyBase.OnPreRender(e)

        tbDate.Columns = 8
        tbDate.Enabled = Enabled

        imgBtnCalendar.AlternateText = Text
        If Not String.IsNullOrEmpty(ImageUrl) Then
            imgBtnCalendar.ImageUrl = ImageUrl
        End If
End Sub

Protected Overrides Sub Render( _
ByVal writer As HtmlTextWriter)
        MyBase.Render(writer)
End Sub
```

Gestion des événements

Il faut également ajouter deux méthodes :

- Sur le clic de l'image, il faut afficher le calendrier.
- Lors de la sélection d'une date dans le calendrier, il faut le cacher et afficher cette date dans la `TextBox`.

Vous devez pour cela ajouter deux méthodes de gestion d'événements et associer les événements à ces gestionnaires :

```
' Dans la méthode CreateChildControls
AddHandler imgBtnCalendar.Click, _
AddressOf onImageButtonClick
AddHandler calendar.SelectionChanged, _
 AddressOf onSelectionChanged
```

```
' Gestionnaires d'événements
Private Sub onImageButtonClick( _
ByVal sender As System.Object, _
ByVal e As System.Web.UI.ImageClickEventArgs)
        calendar.Visible = True
End Sub

Private Sub onSelectionChanged( _
ByVal sender As System.Object, ByVal e As System.EventArgs)
tbDate.Text = calendar.SelectedDate.ToShortDateString
calendar.Visible = False
OnDateChanged(EventArgs.Empty)
End Sub
```

Ajout d'une image par défaut

Vous pouvez ajouter une image par défaut en tant que ressource web. Les ressources web sont une nouveauté en ASP .NET 2.0 et permettent d'intégrer facilement des fichiers dans votre librairie de contrôles et de les utiliser.

Vous devez d'abord procéder de la même façon que pour l'icône, en ajoutant une image et en la transformant en ressource incorporée.

Déclarez ensuite la ressource au niveau de l'assembly dans le fichier *AssemblyInfo.vb*, à l'aide de l'attribut `WebResource` :

```
<assembly: WebResource("Calendar.gif", "image/gif")>
```

Vous pourrez ainsi nommer la ressource (attention, il faut lui ajouter comme préfixe l'espace de noms) et préciser son type MIME.

Pour vous servir de la ressource, utilisez la méthode `GetWebResourceUrl`, qui récupère l'URL d'accès à la ressource :

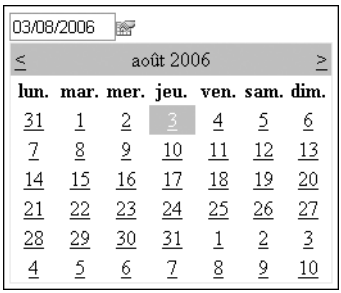
```
imgBtnCalendar.ImageUrl =
    Page.ClientScript.GetWebResourceUrl( _
Me.Gettype, "Calendar.jpg")
```

Ajout dans un projet

Votre contrôle est terminé. Vous pouvez créer un site web et utiliser le contrôle dans une page. Visual Studio a ajouté une référence au contrôle sous l'onglet **Composants Controls** de la boîte à outils. Il est ainsi facile de le glisser-déposer dans votre page.

◄ **Figure 23-15** : Onglet Composants Controls de la boîte à outils

Lorsque vous afficherez la page dans votre navigateur, vous pourrez cliquer sur l'image et sélectionner une date grâce au calendrier



◄ **Figure 23-16 :** Contrôle DatePicker

23.3 Création du designer

Si vous désirez à présent faciliter la personnalisation des propriétés de votre contrôle, vous pouvez lui associer un smart tag, autrement dit une balise active. Un smart tag correspond à un volet de l'Éditeur de Visual Studio. Il sert à afficher les tâches courantes associées à un contrôle. Les balises actives permettent aux composants et aux contrôles d'afficher des informations contextuelles et des commandes pour les utilisateurs. La plupart des contrôles serveurs possèdent donc un smart tag, accessible grâce une petite flèche placée en haut à droite des contrôles.

Vous allez tout d'abord créer la classe choisie précédemment avec l'attribut `Designer`.

Substituez ensuite la propriété `ActionLists` de type `DesignerActionListCollection`, qui donne accès à la liste des éléments du smart tag.

```
Private lists As DesignerActionListCollection

    Public Overrides ReadOnly Property ActionLists() _
       As DesignerActionListCollection
        Get
            If lists Is Nothing Then
                lists = New DesignerActionListCollection()
                lists.Add( _
New DatePickerActionList(Me.Component))
            End If
            Return lists
        End Get
End Property
```

Elle fait appel à une classe `DatePickerActionList`. Cette classe peut être interne puisqu'elle n'est pas utilisée ailleurs. Elle doit hériter de la classe `System.`

ComponentModel.Design.DesignerActionList.

```
Public Class DatePickerActionList
Inherits System.ComponentModel.Design.DesignerActionList
```

Redéfinissez la méthode GetSortedActionItems. Elle va remplir les éléments du smart tag.

```
Public Overrides Function GetSortedActionItems() _
             As DesignerActionItemCollection
Dim items As New DesignerActionItemCollection()

items.Add(New DesignerActionHeaderItem("Propriétés"))

               items.Add( _
               New DesignerActionPropertyItem( _
               "Text", _
               "Texte", _
               "Appearance", _
               "Texte"))

               Return items
End Function
```

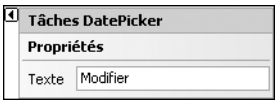
Vous pouvez ajouter différents éléments qui ont chacun leur spécificité. Ainsi, un smart tag permet non seulement d'exposer des propriétés et de les regrouper par catégories, comme dans la fenêtre de propriétés standard de Visual Studio, mais également d'en améliorer l'affichage et l'efficacité. Voici les différentes classes que vous pouvez utiliser pour ajouter un élément à une liste d'éléments d'un smart tag.

Classes utilisées pour créer des éléments dans un smart tag

Classe	Description
DesignerActionItem	Représente un élément de panneau sur un panneau des balises actives.
DesignerActionList	Définit une liste d'éléments utilisés pour créer un panneau des balises actives.
DesignerActionService	Établit un service au moment du design, qui gère la collection d'objets DesignerActionItem pour les composants.
DesignerActionTextItem	Représente un élément de texte statique sur un panneau.
DesignerActionPropertyItem	Représente un élément de panneau associé à une propriété dans une classe dérivée de DesignerActionList.

Classes utilisées pour créer des éléments dans un smart tag	
Classe	**Description**
`DesignerActionMethodItem`	Représente un élément de panneau associé à une méthode dans une classe dérivée de `DesignerActionList`.
`DesignerActionHeaderItem`	Représente un élément d'en-tête statique sur un panneau des balises actives.

Vous obtenez ainsi un smart tag permettant de modifier facilement la propriété `Text`.



◄ **Figure 23-17 :** Smart tag du contrôle DatePicker

23.4 Check-list

Dans ce chapitre, vous avez découvert des nouvelles méthodes de développement des contrôles serveurs personnalisés. Vous pouvez développer simplement des contrôles qui correspondent à vos propres besoins et qui encapsulent des fonctionnalités intéressantes.

Voici un aperçu des notions que vous avez pu acquérir :

- création d'un contrôle composite ;
- affectation d'attributs de classe et de propriété ;
- association d'un Éditeur de propriétés ;
- ajout et utilisation de ressources web ;
- association d'une icône à un contrôle serveur personnalisé ;
- designer de contrôle ;
- création de smart tag.

Fichiers compressés et modèles de projets Visual Studio

Pour comprendre à quel point le Framework .NET 2.0 est vaste et permet de développer la plupart des applications couramment utilisées par un ordinateur, vous allez réaliser un utilitaire de compression et de décompression de fichiers.

Cette fonctionnalité de compression de fichiers n'était pas disponible dans la version 1.1 du Framework, mais comme vous le verrez tout au long de ce chapitre, il est désormais facile de réaliser ce genre de tâche.

La compression permet de minimiser la taille de fichiers et de répertoires pour pouvoir par exemple les transférer plus facilement ou encore lorsque l'espace de stockage est limité.

L'environnement de développement intégré Visual Studio utilise lui-même de nombreux fichiers compressés. Nous aurons l'occasion d'expliquer de quels genres de fichiers Visual Studio se sert pour, par exemple, gérer ses modèles de projets et de fichiers. Nous ferons également le point sur la fonctionnalité d'exportation de modèles de projets, disponible dans les dernières versions de Visual Studio.

24.1 Classes et espaces de noms utilisés

L'espace de noms `System.IO.Compression` contient les nouvelles classes auxquelles vous ferez appel tout au long de ce chapitre.

Seuls deux algorithmes de compression sont pris en charge : Deflate et Gzip.

Si vous êtes habitué à manipuler des fichiers grâce aux nombreuses classes mises à disposition par l'espace de noms `System.IO`, vous n'aurez aucun souci pour comprendre comment fonctionne la compression de fichiers.

24.2 Utilitaire de compression de fichiers

L'application WinForm que vous allez développer permettra de compresser un fichier au format GZIP puis de le décompresser.

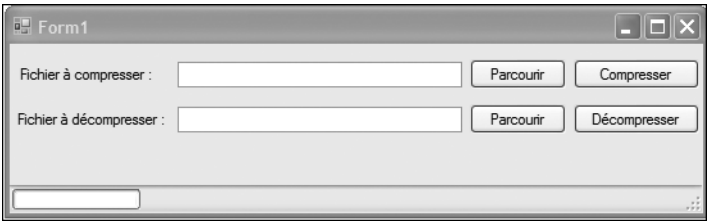
Création du formulaire

Le formulaire WinForm contient les contrôles suivants :

- des zones de saisie pour sélectionner le chemin du fichier ;
- des boutons d'ouverture de boîte de dialogue d'exploration pour trouver plus facilement un fichier ;

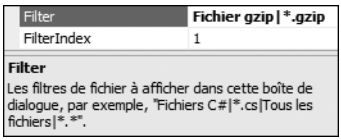
- des boutons qui déclenchent la compression ou la décompression.

L'interface utilisateur obtenue est basique :



▲ **Figure 24-1 :** Formulaire de l'utilitaire de compression

Pour faciliter la recherche de fichiers sur le disque afin que l'utilisateur n'ait pas à saisir manuellement le chemin physique du fichier, vous pouvez déposer un composant `OpenFileDialog` dans votre formulaire. Pour ne lister que certains types de fichiers, vous pouvez configurer les paramètres ad hoc, comme les extensions de fichiers acceptées. La propriété `Filter` permet par exemple d'être sûr que le fichier sélectionné dans la zone de saisie du fichier à décompresser a bien une extension *.gzip*. La valeur à passer correspond au libellé descriptif que vous voulez voir afficher, suivi du modèle de nom de fichier à respecter, le tout séparé par le caractère "|", par exemple : `Fichier gzip|*.gzip`, dans ce cas.

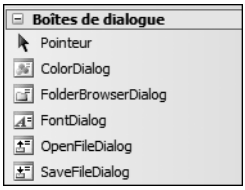


◄ **Figure 24-2 :** Propriété Filter d'un composant OpenFileDialog

Le code pour ouvrir la boîte de dialogue et récupérer sa valeur ressemble à celui-ci :

```
Private Sub BrowseSource_Click(ByVal sender _
    As System.Object, ByVal e As System.EventArgs) _
    Handles btnBrowseFile.Click

    If Me.OpenFileDialog.ShowDialog = DialogResult.OK Then
        Me.tbFileToZip.Text = Me.OpenFileDialog.FileName
    End If

End Sub
```

D'autres boîtes de dialogue existent pour réaliser ce genre d'opération, comme sélectionner un emplacement d'enregistrement de fichier (`SaveFileDialog`) ou sélectionner un dossier (`FolderBrowserDialog`). Elles sont disponibles sous l'onglet **Boîtes de dialogue** de la boîte à outils.



◄ **Figure 24-3 :** Onglet Boîtes de dialogue de la boîte à outils

Compression d'un fichier

La compression consiste à transformer le fichier en un flux afin de stocker son contenu dans un tableau d'octets, puis à déclarer un flux de compression et à y écrire la totalité du fichier.

```
Dim zipFileName As String = Path.GetFileName(fileToZip) & _
    ".gzip"

' Récupération du fichier à compresser dans un flux
Dim sourceStream As FileStream = _
    New FileStream(fileToZip, FileMode.Open, _
        FileAccess.Read, FileShare.Read)

Dim buffer As Byte() = New Byte(sourceStream.Length) {}
Dim length As Integer = sourceStream.Read(buffer, 0, _
    buffer.Length)

' Ouverture du flux de sortie
Dim destinationStream As FileStream = _
    New FileStream(zipFileName, FileMode.OpenOrCreate, _
        FileAccess.Write)

' Création d'un flux de compression
Dim compressedStream As GZipStream = _
    New GZipStream(destinationStream, _
        CompressionMode.Compress, True)

'Écriture des données compressées vers le fichier
'de destination
compressedStream.Write(buffer, 0, buffer.Length)
```

Le fichier compressé dans l'archive au format GZIP portera le même nom que l'archive, mais sans l'extension *.gzip*. C'est pourquoi il est préférable de nommer l'archive avec le nom de fichier original accompagné de l'extension *.gzip*. Ainsi, après avoir été décompressé, le fichier portera bien le nom original et ne perdra pas son extension initiale.

Décompression d'un fichier

La décompression réalise l'opération inverse. Le morceau de code le plus important est donc :

```
decompressedStream = New GZipStream(sourceStream, _
    CompressionMode.Decompress, True)
```

On déclare le flux de décompression dont on va servir pour, tout d'abord, lire le contenu du fichier compressé, puis l'enregistrer dans un fichier qui sera donc la copie de l'original.

Les fichiers compressés (que ce soit dans un format ou dans un autre) sont présents dans tous les logiciels avancés dès qu'il s'agit de stocker des données dans un minimum d'espace. Voyons quels genres de fichiers compressés vous pouvez retrouver dans Visual Studio. Par là même, vous comprendrez mieux le fonctionnement des projets et de leurs éléments, et découvrirez qu'il est possible d'étendre Visual Studio en fonction de vos propres besoins ou de partager vos ressources.

24.3 Utilisation de fichiers compressés par Visual Studio

Visual Studio inclut par défaut de nombreux modèles prédéfinis de projets ou d'éléments de projet. À la création de chaque projet, vous sélectionnez le modèle de projet qui convient le mieux à votre application pour développer un site web, une bibliothèque de classes ou de contrôles ou encore une application WinForm. Vous ajoutez ensuite les éléments de projet appropriés, comme des pages des formulaires WebForm ou WinForm, des classes ou tout autre fichier pris en charge par Visual Studio.

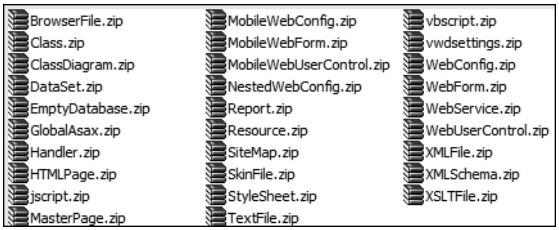
En plus des modèles installés disponibles et accessibles grâce aux boîtes de dialogue **Nouveau projet** et **Ajouter un nouvel élément**, vous pouvez accéder aux modèles que vous avez créés ou à d'autres qui ont été développés, comme des Starters Kits.

ProjectTemplates et ItemTemplates

Le répertoire principal de Visual Studio est *C:\Program Files\Microsoft Visual Studio 8*.

Par défaut, les modèles de projets et d'éléments de projet Visual Basic sont stockés dans les sous-répertoires *Common7\IDE\VWDExpress\ItemTemplates\Web\VisualBasic\1036* et *Common7\IDE\VWDExpress\ProjectTemplates\Web\VisualBasic\1036*.

Vous y retrouverez par exemple l'ensemble des éléments que vous pouvez ajouter à un site web :



◄ **Figure 24-4 :** Éléments de projet web

Remarque

Identificateur de paramètres régionaux

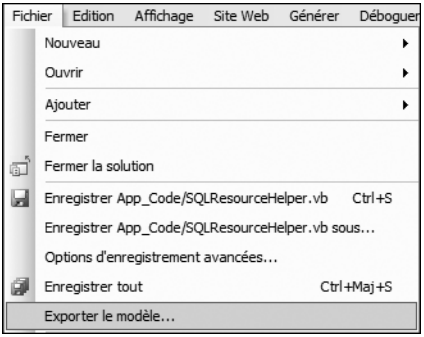
Le nombre 1036 qui figure dans certains noms de dossier correspond en fait à l'identificateur de paramètres régionaux (LCID) de la culture française. De nombreux autres dossiers portent le nombre 1033 réservé à la culture anglaise.

Templates Visual Studio

Une fonctionnalité intéressante fait son apparition dans les nouvelles versions de Visual Studio. Vous avez en effet la possibilité de créer vos propres modèles de projets en exportant un projet existant dont vous voudriez vous servir comme base pour de futurs projets.

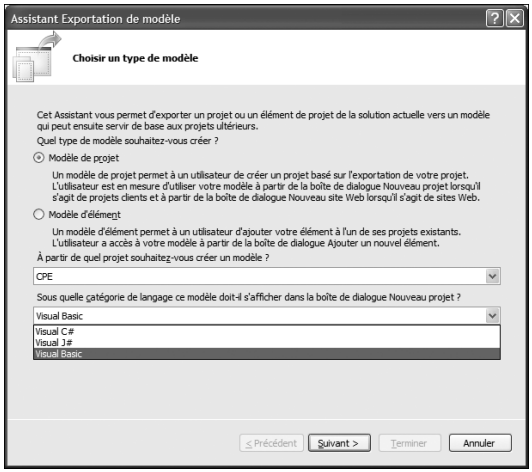
Voici les différentes étapes qui permettent de créer un modèle de projet :

1. La commande **Exporter le modèle** du menu **Fichier** ouvre un Assistant qui va vous aider à personnaliser l'exportation de votre modèle de projet.



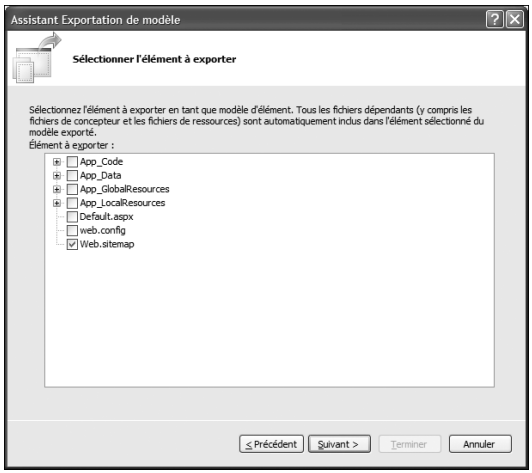
◄ **Figure 24-5 :** Commande Exporter le modèle du menu Fichier

2 La première étape est la sélection du type de modèle à exporter. Vous avez le choix entre exporter un projet ou un élément de projet. Vous pouvez également choisir le langage cible à utiliser.



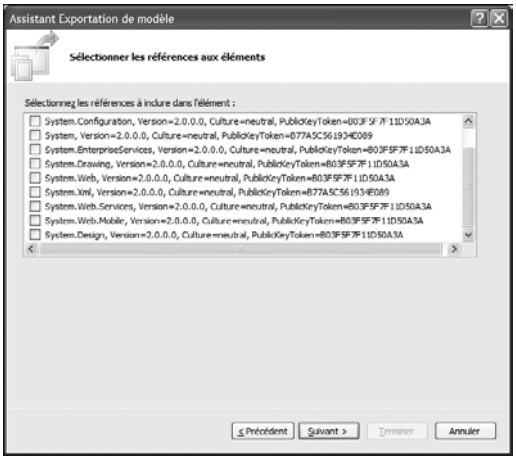
▲ **Figure 24-6 :** Assistant d'exportation de modèle

3 Si vous exportez un élément de projet, deux étapes supplémentaires de configuration sont nécessaires. La première sert à sélectionner l'élément à exporter dans une arborescence des fichiers disponibles précédés par des cases à cocher de sélection.



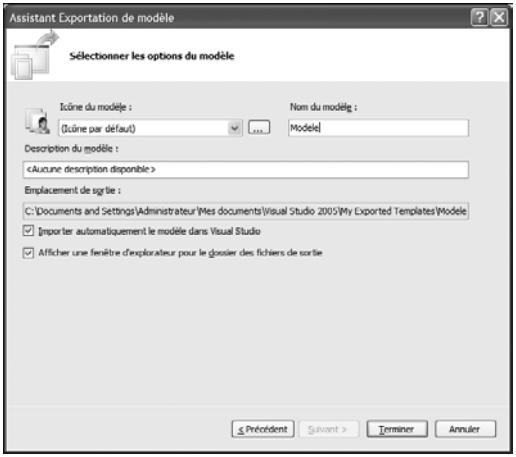
▲ **Figure 24-7 :** Assistant d'exportation de modèle : Sélectionner l'élément à exporter

4 L'autre étape intermédiaire permet de sélectionner des références dont vous voulez vérifier l'existence et de les inclure, si elles n'existent pas, lors de l'insertion de l'élément. Vous vous assurez ainsi qu'il n'y aura pas d'erreur de compilation pour cause de référence manquante.



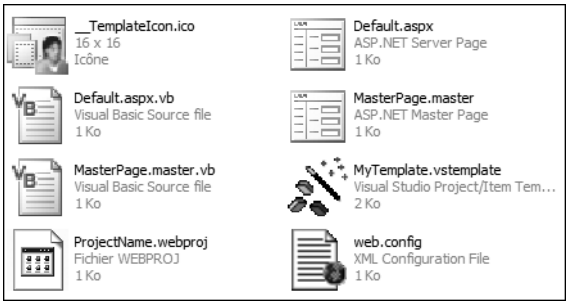
▲ **Figure 24-8 :** Assistant d'exportation de modèle : Sélectionner les références aux éléments

5 La dernière étape **Sélectionner les options du modèle** permet de spécifier certaines informations sur le modèle à exporter, comme son nom, sa description ou une icône autre que celle par défaut.



▲ **Figure 24-9 :** Assistant d'exportation de modèle : Sélectionner les options du modèle

L'emplacement du fichier ZIP généré se situe dans *[Mes documents]\Visual Studio 2005\My Exported Templates*. Si vous décompressez l'archive obtenue, vous retrouverez l'ensemble des fichiers qui constituent le projet ainsi qu'un fichier avec une extension *.vstemplate*.



◀ **Figure 24-10 :** Fichiers contenus dans le modèle de projet

Fichier vstemplate

Chaque modèle inclut un fichier avec une extension *.vstemplate*, doté de métadonnées, qui fournit à Visual Studio les informations requises pour afficher le modèle dans les boîtes de dialogue **Nouveau projet** et **Ajouter un nouvel élément** et créer un projet ou un élément à partir du modèle.

Structure d'un fichier .vstemplate

Le fichier *.vstemplate* des modèles de projets se compose de trois éléments fondamentaux :

- `VSTemplate` désigne le type de modèle (de projet ou d'élément).
- `TemplateData` définit la catégorie dans laquelle sera situé le modèle de projet et les propriétés d'affichage de la boîte de dialogue **Nouveau projet** ou **Ajouter un nouvel élément**.
- `TemplateContent` regroupe l'ensemble des fichiers inclus dans le modèle.

La structure XML d'un fichier *.vstemplate* est donc la suivante :

```
<VSTemplate Type="Project" Version="2.0.0"
    xmlns="http://schemas.microsoft.com/
        developer/vstemplate/2005">
    <TemplateData>
        …
    </TemplateData>
    <TemplateContent>
        …
    </TemplateContent>
</VSTemplate>
```

Assistants

En plus des éléments XML fondamentaux d'un modèle, vous pouvez utiliser les éléments `WizardExtension` et `WizardData` pour ajouter les fonctionnalités personnalisées à l'Assistant de modèle qui crée un projet ou un élément à partir du modèle. Pour cela, vous devez créer vous-même un assembly qui implémente l'interface IWizard, et en faire référence dans le fichier *.vstemplate*.

ProjectSubType

L'élément `ProjectSubType` peut avoir les valeurs suivantes :

- `Windows` pour les applications WinForm ;
- `Office` pour les applications Visual Studio Tools For Office ;
- `Database` pour les bases de données ;
- `Smart Devices` pour les applications destinées aux périphériques mobiles ;
- `Web` pour les sites web.

Voici un exemple de fichier *.vstemplate* pour un élément de projet :

```
<VSTemplate Version="2.0.0" Type="Item" Version="2.0.0">
    <TemplateData>
        <Name>Nom du fichier</Name>
        <Description>Description</Description>
        <Icon>icone.ico</Icon>
        <ProjectType>VBasic</ProjectType>
        <ProjectSubType>Windows</ProjectSubType>
        <DefaultName>defaultName.vb</DefaultName>
    </TemplateData>
    <TemplateContent>
        <ProjectItem>Modele.vb</ProjectItem>
    </TemplateContent>
</VSTemplate>
```

Ce tableau présente les différents éléments XML que peut contenir un fichier *.vstemplate* :

Éléments d'un fichier .vstemplate		
Élément	**Élément enfant**	**Attribut**
`Assembly`		
`BuildOnLoad`		
`CreateInPlace`		
`CreateNewFolder`		
`CustomParameter`		`Name, Value`

Éléments d'un fichier .vstemplate		
Élément	**Élément enfant**	**Attribut**
CustomParameters	CustomParameter	
DefaultName		
Description		Package | ID
EnableLocationBrowseButton		
Folder	ProjectItem | Folder	Name
FullClassName		
Hidden		
Icon		Package | ID
LocationField		
LocationFieldMRUPrefix		
Name		Package | ID
NumberOfParentCategories ToRollUp		
Project	Folder | ProjectItem	File | TargetFileName | ReplaceParameters
ProjectCollection	ProjectTemplateLink | SolutionFolder	
ProjectItem (modèle d'élément)		SubType | ReplaceParameters | TargetFileName
ProjectItem (modèle de projet)		TargetFileName | ReplaceParameters | OpenInEditor | OpenOrder | OpenInWebBrowser | OpenInHelpBrowser
ProjectSubType		
ProjectTemplateLink		ProjectName
ProjectType		
PromptForSaveOnCreation		
ProvideDefaultName		
Reference	Assembly	
References	Reference	
ShowByDefault		
SolutionFolder	ProjectTemplateLink | SolutionFolder	Name

Éléments d'un fichier .vstemplate		
Élément	**Élément enfant**	**Attribut**
SortOrder		
SupportsCodeSeparation		
SupportsLanguageDropDown		
SupportsMasterPage		
TemplateContent	ProjectCollection | Project | Références | ProjectItem | CustomParameters	
TemplateData	Name | Description | Icon | ProjectType | ProjectSubType | TemplateID | TemplateGroupID | SortOrder | CreateNewFolder | DefaultName | ProvideDefaultName | PromptForSaveOnCreation | EnableLocationBrowseButton | Hidden | DisplayInParentCategories | LocationFieldMRUPrefix | NumberOfParentCategories ToRollUp | CreateInPlace | BuildOnLoad | ShowByDefault | LocationField | SupportsMasterPage | SupportsCodeSeparation | SupportsLanguageDropDown	
TemplateGroupID		
TemplateID		
VSTemplate	TemplateData | TemplateContent | WizardExtension | WizardData	Type | Version
WizardData		Name
WizardExtension	Assembly | FullClassName	

Fichiers d'installation .vsi

Les fichiers d'installation (*.vsi*) de Visual Studio sont utilisés pour échanger facilement un contenu Visual Studio. Il s'agit en réalité d'un fichier *.zip* renommé qui contient les fichiers que vous voulez mettre à disposition, mais également un fichier spécial avec une extension *.vscontent*, qui doit répondre à un schéma XML précis. Ce fichier *.vscontent* fait de l'archive *.vsi* un véritable programme d'installation, qui permet d'installer votre contenu à un emplacement approprié détecté automatiquement par Visual Studio.

Éléments d'un fichier .vscontent

L'élément racine peut contenir un ou plusieurs éléments `Content` qui servent à définir les contenus à installer.

Éléments d'un fichier .vscontent

Élément	Description
`FileName`	Sert à définir les fichiers à copier.
`DisplayName`	Nom du contenu à afficher dans le programme d'installation de contenu de Visual Studio.
`Description`	Description du contenu qui s'affiche en tant qu'info-bulle dans le programme d'installation de contenu de Visual Studio.
`FileContentType`	Type de contenu parmi les valeurs suivantes : `VSTemplate` : si le contenu est un Starter Kit ou un modèle Visual Studio. `Code Snippet` : si le contenu est un extrait de code. `Toolbox Control` : si le contenu est un contrôle de boîte à outils. `Addin` : si le contenu est un complément. `Macro Project` : si le contenu est un projet de macro.
`ContentVersion`	Version du contenu.
`Attributes`	Peut contenir aucun ou plusieurs éléments `Attributes`.
`Attribute`	Information sur le contenu. Il utilise des attributs de nom et de valeur pour ajouter les informations de contenu personnalisées. Exemple : `<Attributes> <Attribute name="TemplateType" value="Project"/> </Attributes>` Un contenu doté d'une valeur `FileContentType` de `Code Snippet` requiert l'attribut `lang`, qui spécifie le langage de programmation de l'extrait de code : `VB`, `CSHARP`, `JSHARP` ou `XML`.

Création d'un fichier d'installation

Vous devez définir des valeurs et des attributs spécifiques dans le fichier *.vscontent* afin de déterminer comment et à quel emplacement installer le modèle.

Dans le fichier *.vscontent*, vous devez effectuer les opérations suivantes :

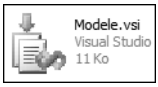
1. Spécifiez dans l'élément `FileName` le nom de fichier compressé qui contient le modèle.
2. Attribuez à l'élément `FileContentType` la valeur `VSTemplate`.
3. Ajoutez les éléments `Attribute` suivants :
   - `ProjectType` : spécifie le type de projet, à savoir Visual Basic, Visual C#, Visual J# ou Visual Web Developer.
   - `ProjectSubType` : spécifie la sous-catégorie dans laquelle sera placé le modèle dans la boîte de dialogue **Nouveau projet**.
   - `TemplateType` : spécifie le type de modèle, à savoir `Project` pour les modèles de projets, ou `Item` pour les modèles d'éléments de projet.

L'exemple de code suivant montre à quoi doit ressembler le fichier *.vscontent* qui va servir à configurer l'installation du modèle que vous venez de créer dans la catégorie racine *Visual Basic* de la boîte de dialogue **Nouveau projet** :

```
<VSContent
    xmlns="http://schemas.microsoft.com/
        developer/vscontent/2005">
    <Content>
        <FileName>Modele.zip</FileName>
        <DisplayName>Modèle de site web</DisplayName>
        <Description>Modèle de site web</Description>
        <FileContentType>VSTemplate</FileContentType>
        <ContentVersion>1.0</ContentVersion>
        <Attributes>
            <Attribute name="TemplateType"
                value="Project"/>
            <Attribute name="ProjectType"
                value="Visual Web Developer"/>
            <Attribute name="ProjectSubType"
                value="VisualBasic"/>
        </Attributes>
    </Content>
</VSContent>
```

Après avoir enregistré ce fichier au même emplacement que le modèle de projet, ajoutez-les à une archive compressée puis renommez-la avec une

extension *.vsi*. Le fichier apparaît alors avec une icône signifiant qu'il est reconnu par Visual Studio.

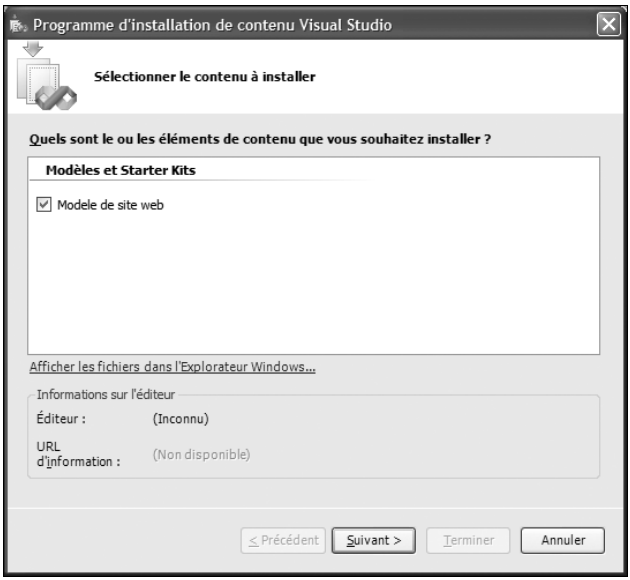


◄ **Figure 24-11 :** Fichier VSI

Installation à partir d'un fichier .vsi

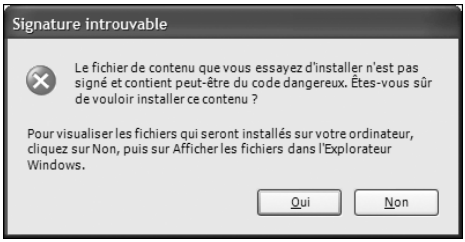
L'installation du modèle sur un autre poste est simple. Il suffit d'exécuter le fichier d'installation et de suivre les étapes.

1 La première étape affiche le ou les éléments qui vont être installés. Vous avez la possibilité de ne sélectionner que ceux qui vous intéressent.



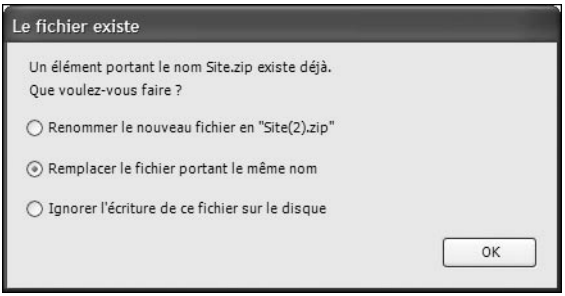
▲ **Figure 24-12 :** Sélection du contenu à installer

2 Si le projet n'a pas été signé, un message d'alerte vous le signale.



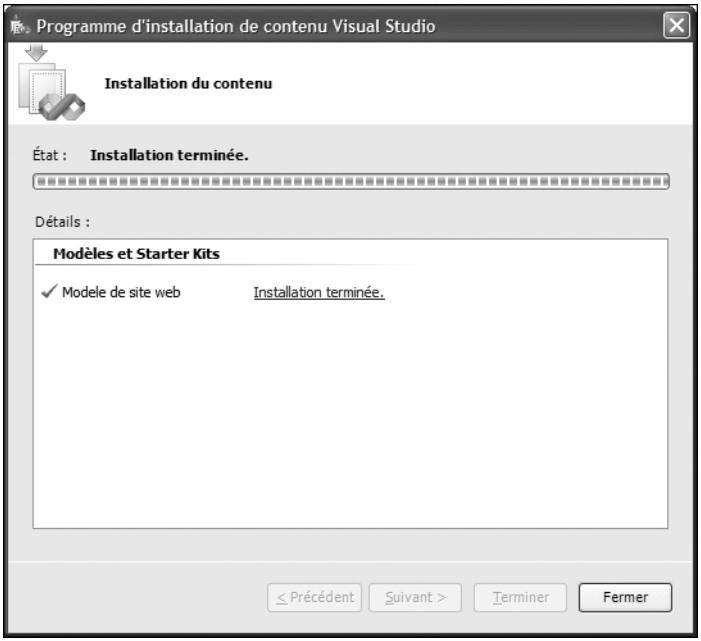
◄ **Figure 24-13 :** Message d'alerte "Signature introuvable"

3 Si vous exécutez une seconde fois cette installation ou si un fichier *.zip* du même nom que le modèle existe déjà, une boîte de dialogue vous demande de quelle façon vous voulez agir. S'il s'agit d'un remplacement, nous conseillons de choisir **Remplacer le fichier portant le même nom**.



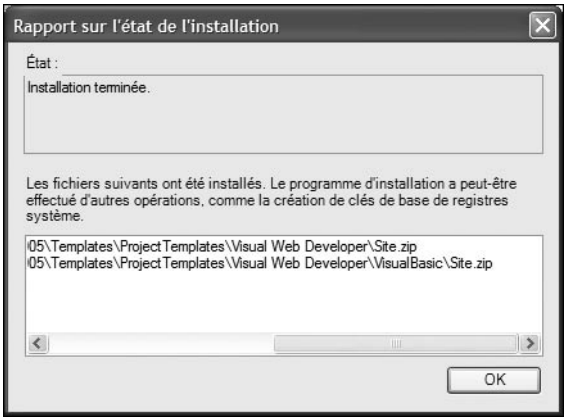
◄ **Figure 24-14 :** Boîte de dialogue Le fichier existe

4 Enfin, validez l'installation et observez le déroulement de l'opération.



▲ **Figure 24-15 :** Déroulement de l'installation

5 Si vous cliquez sur le lien *Installation terminée*, une fenêtre récapitulative s'ouvre indiquant notamment les répertoires dans lesquels le modèle a été copié.



◄ **Figure 24-16 :** Rapport sur l'état d'installation

Starter Kits

Un Starter Kit n'est autre qu'un modèle de projet un peu amélioré. La différence est notamment qu'un Starter Kit inclut souvent en complément une documentation détaillée de l'application mise à disposition.

Recherche et partage de Starter Kits

Dans la boîte de dialogue **Nouveau projet**, vous trouverez une liste de Starter Kits installés avec Visual Studio. Vous pourrez rechercher et télécharger des Starter Kits à partir de sites web qui prennent en charge les services Web Visual Studio.

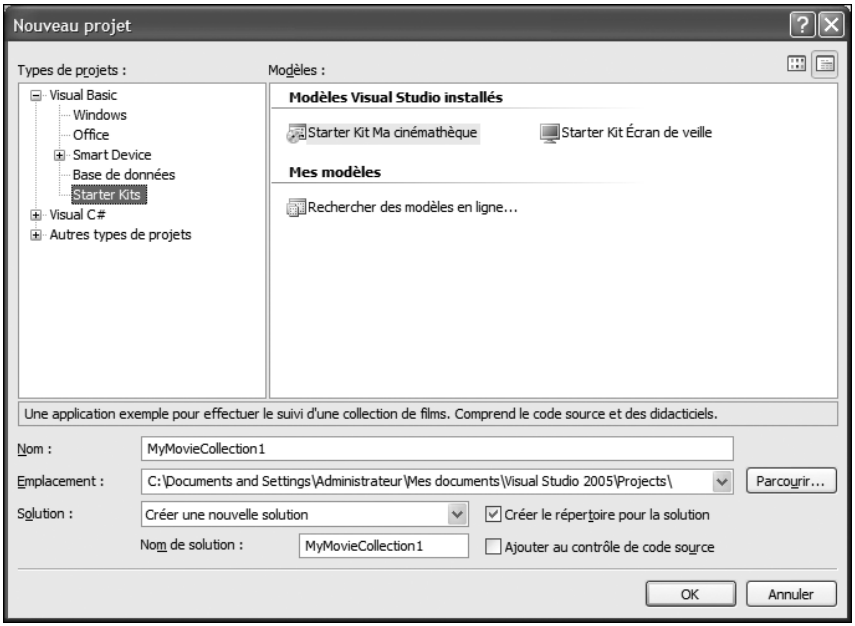
Utilisation de Starter Kits pour créer des projets

Visual Studio 2005 inclut plusieurs Starter Kits à utiliser comme exemple.

Pour créer un projet à partir d'un Starter Kit inclus dans Visual Studio, sélectionnez la commande **Nouveau/Fichier** du menu **Fichier**.

Dans la liste *Types de projet* de la boîte de dialogue **Nouveau projet**, développez les nœuds correspondant au langage de programmation souhaité, puis cliquez sur *Starter Kits*.

Sélectionnez un Starter Kit, entrez le nom du nouveau projet et cliquez sur OK.



▲ **Figure 24-17 :** Ajouter un nouveau projet Starter Kit

24.4 Check-list

Le Framework 1.1 ne permettait pas de compresser et de décompresser des fichiers. Désormais, cela est possible grâce aux nouvelles classes de l'espace de noms `System.IO.Compression`.

L'utilisation des fichiers compressés est couramment répandue pour répondre à différentes problématiques d'optimisation. Même Visual Studio a recours à certains fichiers compressés pour stocker ces modèles de projets.

Vous avez découvert les différentes façons de créer vos propres modèles de projets et appris à les accompagner de programme d'installation ou à personnaliser leur ouverture.

Plus précisément, les fonctionnalités abordées ont été les suivantes :

- la création d'un modèle de projet ou d'élément de projet (*.vstemplate*) ;
- la création d'un fichier *.vscontent* ;
- la création d'un programme d'installation *.vsi* ;
- l'exécution d'un programme d'installation ;
- la création d'un projet à partir d'un Starter Kit.

Toutes ces fonctionnalités rentrent dans une démarche de partage de l'information et d'empaquetage de solutions prêtes à l'emploi. Cela vous oblige notamment à faire le point sur vos méthodologies de développement et à vous questionner sur les différentes façons de réutiliser des éléments intéressants que vous avez mis en place afin de gagner du temps par la suite.

Création d'un client FTP

Le protocole de transfert de fichiers FTP est couramment utilisé pour télécharger des fichiers. Pour vous connecter à un serveur FTP, vous avez recours à un logiciel appelé "client FTP", qui permet de gérer facilement les fichiers qui s'y trouvent.

Dans ce chapitre, vous apprendrez tout d'abord à installer un serveur FTP et à le configurer puis à utiliser un client FTP. Enfin, vous créerez une classe d'aide pour la connexion à un serveur FTP, et des opérations de base au moyen des classes de l'espace de noms `System.Net`.

25.1 Protocole FTP

Le protocole FTP répond à certaines normes et les opérations que vous pouvez effectuer lorsque vous vous connectez à un serveur FTP sont codifiées. En d'autres termes, FTP répond à une syntaxe de commandes et de codes de réponse précise.

Les différentes opérations possibles sont :

- la connexion au serveur FTP ;
- le téléchargement de fichiers du serveur vers la machine locale (download) ;
- le téléchargement de fichiers de la machine locale vers le serveur (upload) ;
- la création et la suppression de répertoires ;
- le listage de répertoires ;
- la navigation entre les répertoires (affectation du répertoire courant).

25.2 Serveur FTP

Installation d'un serveur FTP

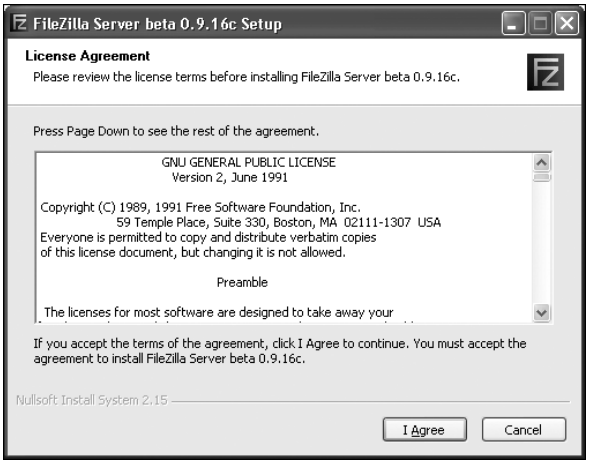
Pour effectuer des tests, il faut un serveur FTP à disposition sur lequel effectuer les requêtes. Nous allons tout d'abord vous présenter un exemple de serveur FTP qui possède toutes les fonctionnalités de base de ce genre de logiciel.

Le logiciel FileZillaServer est un serveur FTP gratuit, téléchargeable à partir du site **http://sourceforge.net/projects/filezilla**. **http://sourceforge.net** regroupe de nombreux projets open source. Après avoir récupéré l'exécutable, réalisez les étapes d'installation et de configuration suivantes :

1. Lancez l'exécutable d'installation.

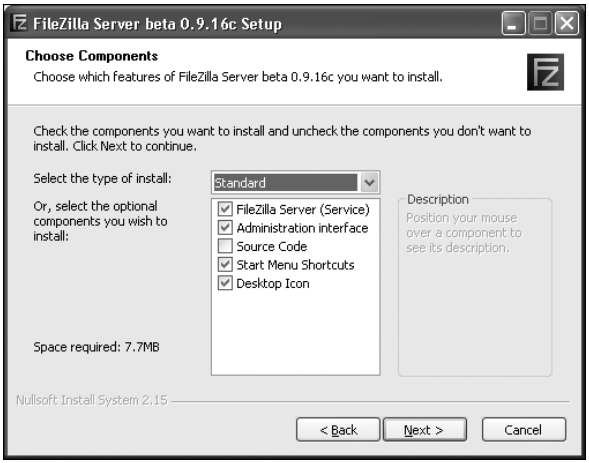
◄ **Figure 25-1 :** Exécutable d'installation de FileZillaServer

2 Le premier écran vous demande d'accepter les termes du contrat de licence utilisateur.



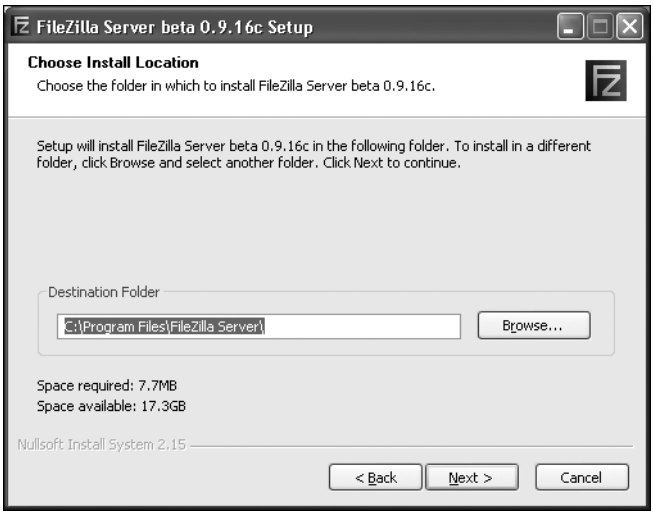
▲ **Figure 25-2 :** Contrat de licence utilisateur

3 L'écran suivant vous permet de sélectionner les composants qui vont être installés, notamment le serveur FTP et une interface d'administration.



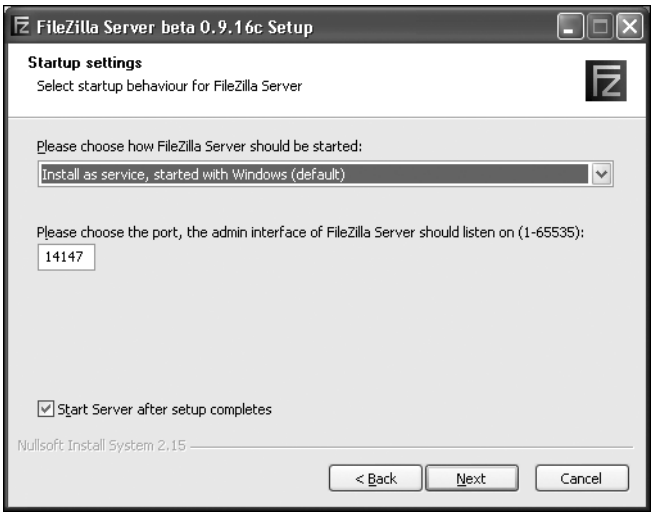
▲ **Figure 25-3 :** Sélection des composants à installer

4 L'Assistant vous demande ensuite de spécifier le répertoire d'installation.



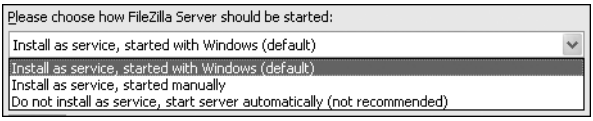
▲ **Figure 25-4 :** Sélection du répertoire d'installation

5 Sélectionnez le mode de démarrage du serveur FTP et son port.



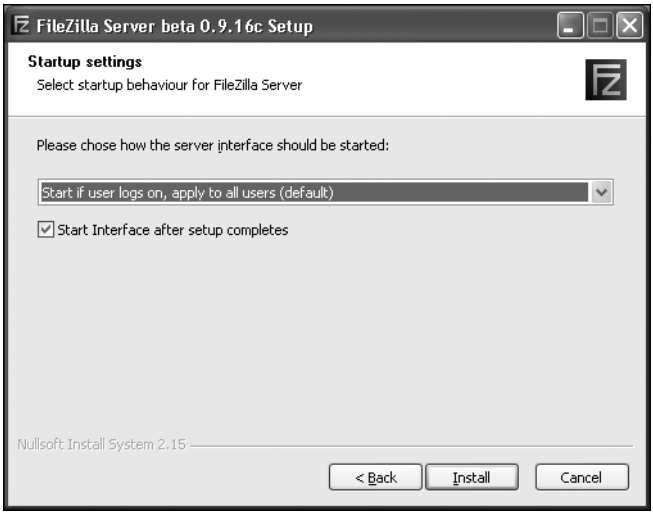
▲ **Figure 25-5 :** Configuration des paramètres de démarrage du serveur

Vous avez le choix entre installer un service (c'est l'option préférable) qui démarre automatiquement avec Windows ou manuellement ou de ne pas installer de service.



▲ **Figure 25-6 :** Sélection du mode de démarrage du serveur

6 Indiquez si l'interface d'administration doit être démarrée manuellement ou quand un utilisateur se connecte et si elle démarre pour tous les utilisateurs ou seulement pour celui en cours.



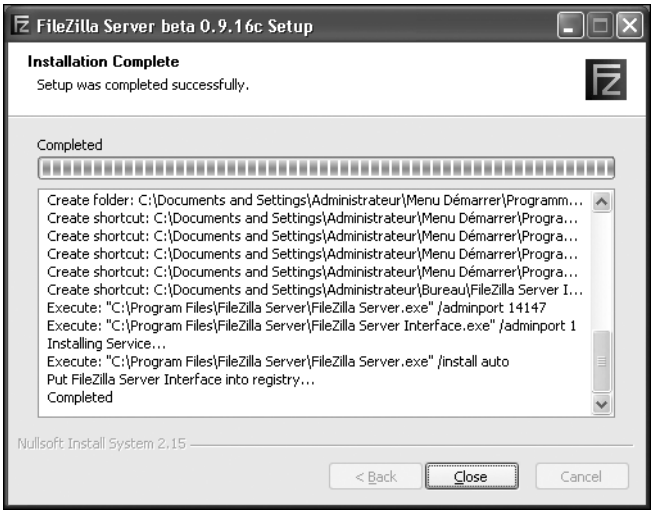
▲ **Figure 25-7 :** Sélection du mode de démarrage : à la connexion de l'utilisateur

Remarque

Démarrage du serveur après l'installation

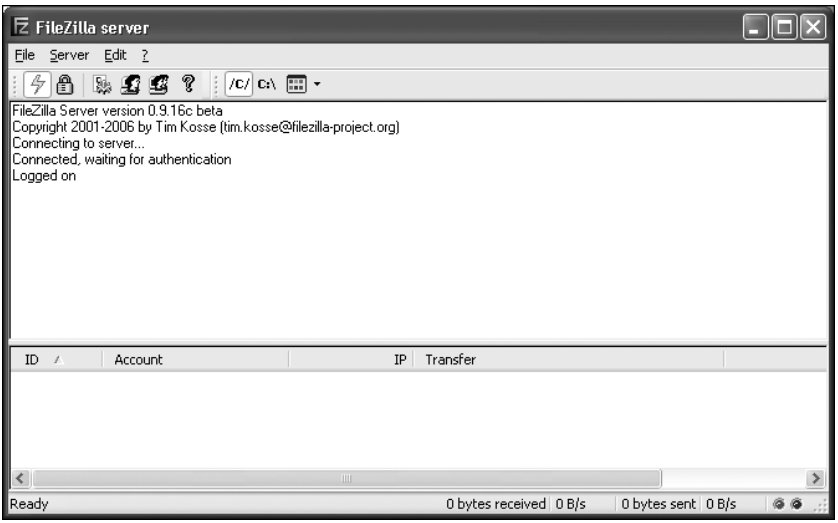
Laissez les cases *Starts Server after setup completes* et *Starts Interface after setup completes* **cochées afin que le serveur et l'interface d'administration démarrent à la fin de l'installation.**

7 Le dernier écran vous montre la progression de l'installation et les composants qui sont installés.



▲ **Figure 25-8 :** Écran de progression de l'installation

8 L'interface d'administration se lance alors comme prévu. Il s'agit en réalité d'une fenêtre à partir de laquelle vous pouvez voir l'activité de votre serveur FTP, c'est-à-dire les commandes qui lui sont demandées et les réponses qu'il renvoie. Elle affiche ce qu'on appelle les logs (journaux) du serveur.



▲ **Figure 25-9 :** Interface d'administration du serveur FTP FileZilla

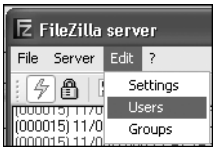
Cette interface indique notamment qu'elle est connectée au serveur et qu'elle est dans l'attente de l'authentification d'un utilisateur.

Création d'un utilisateur

Vous allez maintenant créer un utilisateur qui pourra accéder à votre serveur FTP. Vous indiquerez son identifiant (login), son mot de passe et définirez les répertoires de la machine qu'il peut visiter, en lecture et/ou en écriture, ou dans lesquels il ne peut naviguer. Vous spécifierez au besoin le taux de transfert maximum à ne pas dépasser. En outre, vous êtes libre d'autoriser les connexions provenant de certaines adresses ou plages d'adresses IP, qu'il vous faudra préciser.

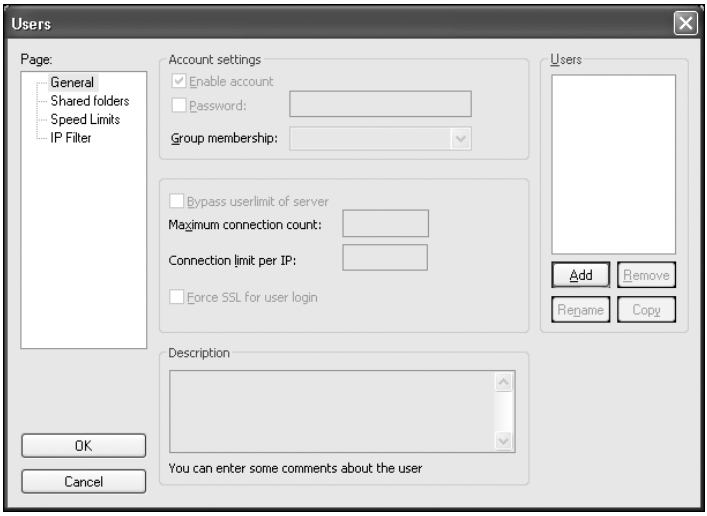
Voici les étapes de création d'un utilisateur :

1. Dans le menu **Edit**, sélectionnez la commande **Users** pour gérer les utilisateurs du serveur.



◄ **Figure 25-10** : Commande Edit/Users

2. La fenêtre de gestion des utilisateurs s'ouvre. Aucun utilisateur n'est présent pour le moment dans la liste située à droite.



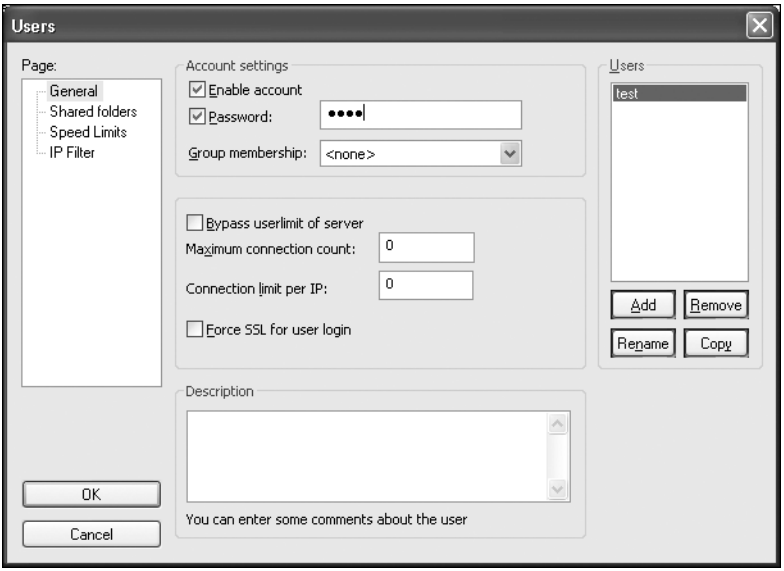
▲ **Figure 25-11** : Interface d'administration du serveur FTP FileZilla

3 Add Cliquer sur le bouton **Add**.

4 La boîte de dialogue d'ajout d'un utilisateur s'ouvre. Saisissez le nom d'utilisateur `test`.

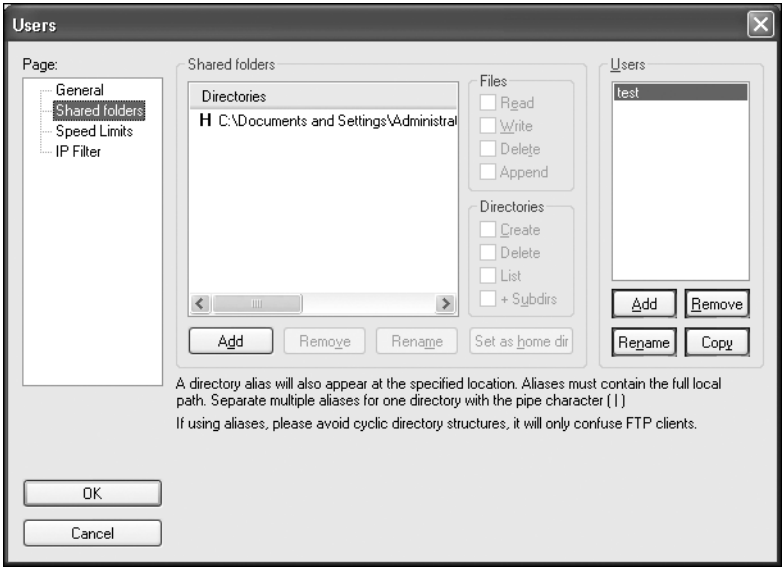
◄ **Figure 25-12 :** Boîte de dialogue d'ajout d'un utilisateur

5 `test` est ajouté dans la liste des utilisateurs. Vous pouvez l'activer en cochant la case *Enable account* et spécifier son mot de passe en cochant la case *Password* et en le saisissant.



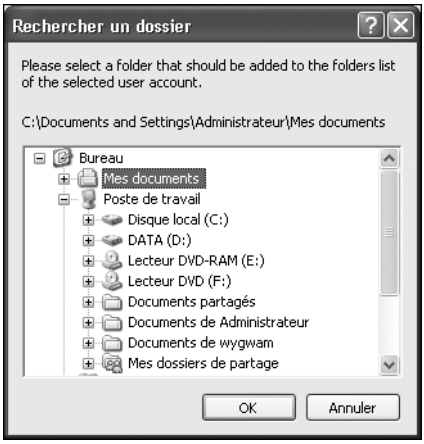
▲ **Figure 25-13 :** Activation du compte et affectation du mot de passe

6 Sélectionnez la page *Shared folders* pour lui attribuer des répertoires qu'il peut visiter.



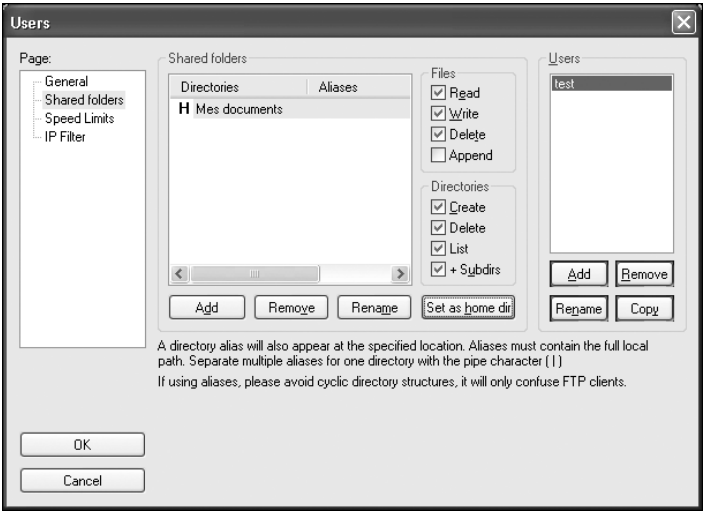
▲ **Figure 25-14 :** Page Shared folders

7 Pour ajouter un répertoire, cliquez sur le bouton **Add**. Une boîte de dialogue s'ouvre.



◄ **Figure 25-15 :** Sélection d'un répertoire que l'utilisateur peut visiter

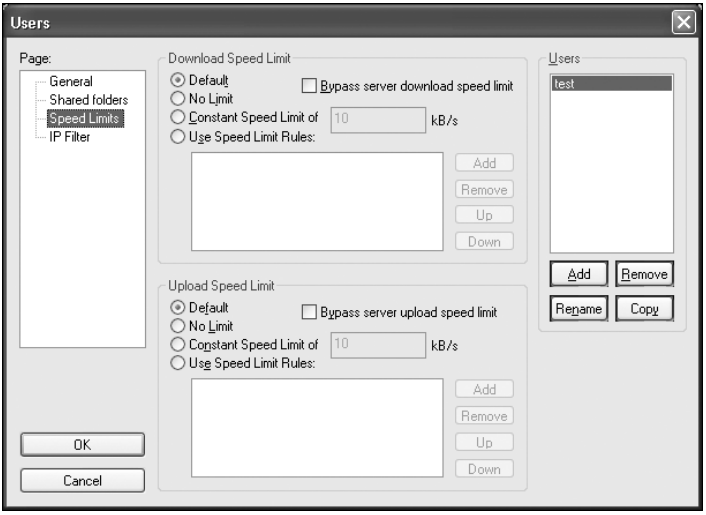
8 Sélectionnez le répertoire et spécifiez les droits de l'utilisateur, comme la création ou la suppression de répertoires ou la lecture ou l'écriture de fichiers.



▲ **Figure 25-16 :** Affectation des droits de l'utilisateur dans un répertoire

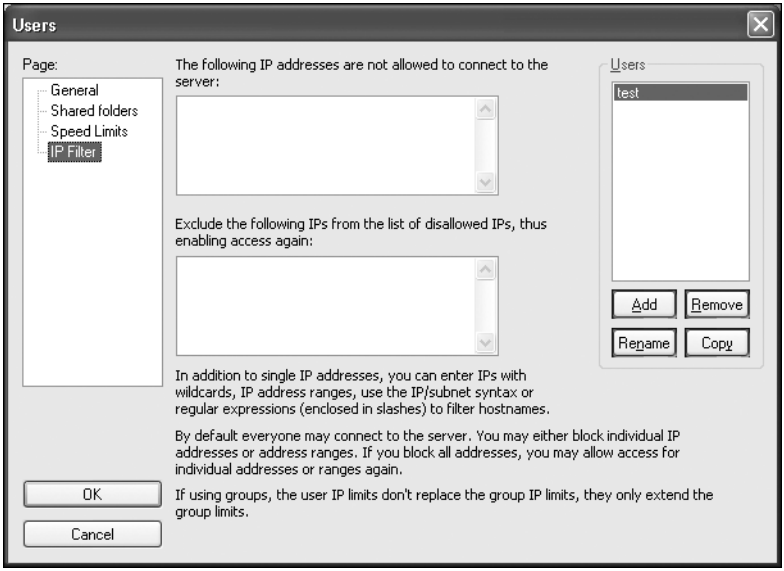
9 Set as home dir Vous pouvez également choisir ce répertoire comme étant celui par défaut grâce au bouton **Set as home dir**.

10 La page *Speed Limits* permet de limiter le taux de transfert sur le téléchargement de fichiers.



▲ **Figure 25-17 :** Page Speed Limits

11 La page *IP Filter* permet d'indiquer des adresses ou des plages d'adresses IP pour lesquelles l'accès est autorisé.



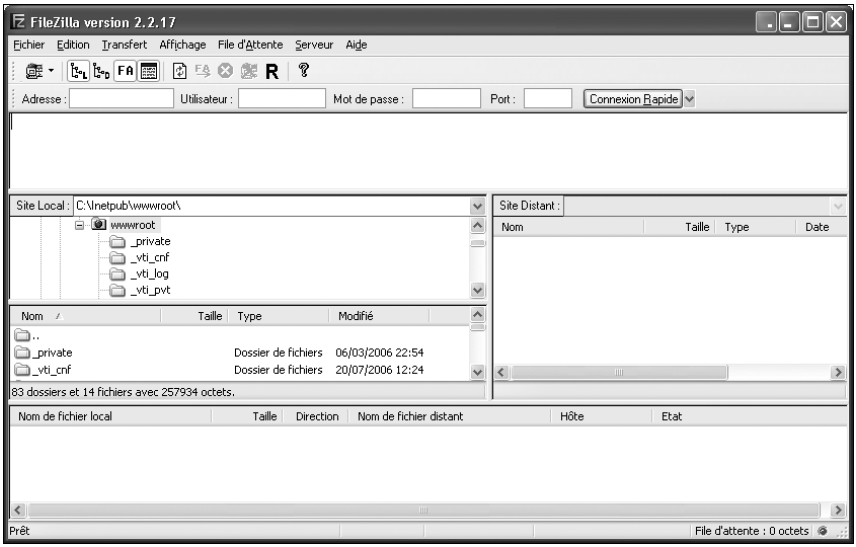
▲ **Figure 25-18 :** Page IP Filter

25.3 Client FTP

Voici un aperçu, côté client, du logiciel de gestion de transfert FTP FileZilla. Son interface est un peu plus évoluée que la précédente et permet de télécharger des fichiers sur un serveur vers votre machine locale, ou inversement d'uploader des fichiers dans un répertoire du serveur. Ce logiciel présente deux avantages : il est traduit en français et il est simple d'utilisation grâce à une interface graphique efficace composée d'Explorateurs, de fenêtres de logs, de progression des transferts, etc. (voir Figure 25-19).

Vous pouvez facilement naviguer dans les répertoires de votre machine et du serveur, et glisser-déplacer des fichiers entre l'un et l'autre.

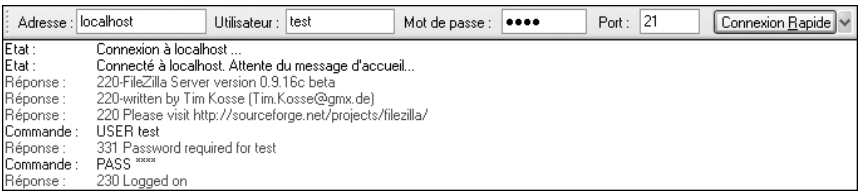
Voici quelques fonctionnalités intéressantes de FileZilla. On les retrouve habituellement dans de ce genre de logiciel.



▲ **Figure 25-19 :** Interface graphique de FileZilla

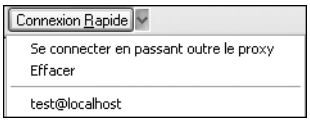
Connexion rapide

Grâce à une barre composée de plusieurs zones de saisie, vous pouvez vous connecter facilement à un serveur FTP en fournissant vos informations de connexion, c'est-à-dire essentiellement l'adresse du serveur, votre nom d'utilisateur et votre mot de passe. Dans ce cas, l'adresse du serveur FTP est **ftp://localhost**.



▲ **Figure 25-20 :** Barre de connexion rapide

Le bouton **Connexion Rapide** enregistre les dernières adresses auxquelles vous avez essayé de vous connecter.



◄ **Figure 25-21 :** Bouton de connexion rapide

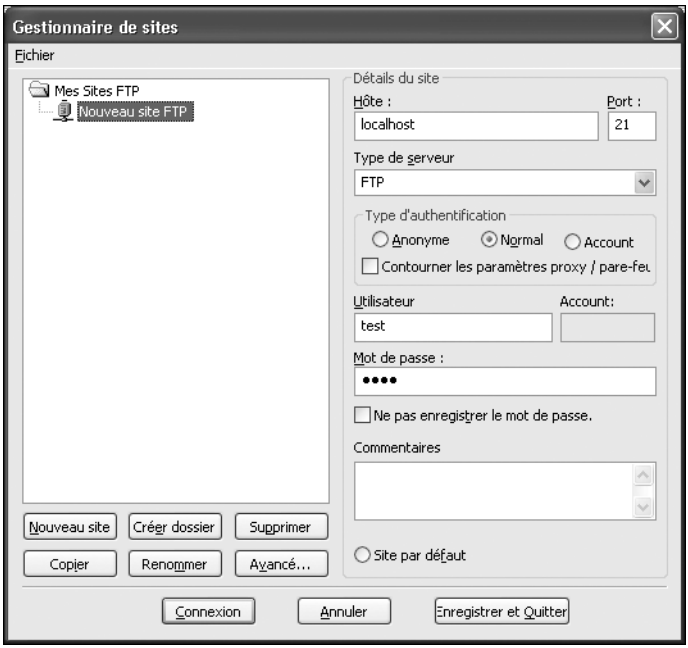
Enregistrer une connexion

Si vous vous connectez souvent à une adresse FTP précise, vous pouvez choisir de l'enregistrer dans le gestionnaire de sites qui stocke plusieurs connexions et vous permet de les gérer et de vous connecter facilement aux serveurs FTP correspondants.

Pour enregistrer une connexion, procédez aux étapes suivantes :

1.  Cliquez sur le bouton d'affichage du gestionnaire de sites.

2. La fenêtre de gestion des sites s'affiche, avec la liste des sites enregistrés d'un côté et un formulaire de configuration des connexions permettant de spécifier notamment les informations de connexion.



▲ **Figure 25-22 :** Gestionnaire de sites

3.  Le bouton **Nouveau Site** permet d'ajouter une nouvelle connexion que vous devez ensuite configurer.
4.  Vous pouvez la désigner comme la connexion par défaut en activant le bouton radio *site par défaut*.
5. Cliquez enfin sur **Enregistrer** et quitter.

6 Votre connexion est désormais enregistrée. Vous y accéderez facilement en cliquant sur la flèche située à côté du bouton du gestionnaire de sites.

◀ **Figure 25-23 :** Connexion à un site enregistré dans le gestionnaire de sites

Fenêtre d'affichage du journal des messages

La fenêtre d'affichage du journal des messages permet de visualiser les messages échangés entre le client et le serveur. On distingue trois types de messages :

- `Etat` indique l'état du serveur ou l'opération qu'il est en train de réaliser.
- `Réponse` indique une réponse du serveur, avec comme préfixe un code de trois chiffres.
- `Commande` indique une commande du client FTP.

Voici une liste d'instructions échangées entre le client et le serveur FTP au moment de la connexion d'un utilisateur et du listage d'un répertoire :

```
Etat :    Connexion à localhost…
Etat :    Connecté à localhost. Attente du message d'accueil.
Réponse :    220-FileZilla Server version 0.9.16c beta
Réponse :    220-written by Tim Kosse (Tim.Kosse@gmx.de)
Réponse :    220 Please visit
http://sourceforge.net/projects/filezilla/
Commande :    USER test
Réponse :    331 Password required for test
Commande :    PASS ****
Réponse :    230 Logged on
Commande :    FEAT
Réponse :    211-Features:
Réponse :     MDTM
Réponse :     REST STREAM
Réponse :     SIZE
Réponse :     MLST type*;size*;modify*;
Réponse :     UTF8
Réponse :     CLNT
Réponse :    211 End
Commande :    CLNT FileZilla
Réponse :    200 Don't care
Commande :    OPTS UTF8 ON
Réponse :    200 UTF8 mode enabled
Commande :    SYST
Réponse :    215 UNIX emulated by FileZilla
Etat :    Connecté
Etat :    Récupération de la liste de répertoires…
```

```
Commande :    PWD
Réponse :    257 "/" is current directory.
Commande :    TYPE A
Réponse :    200 Type set to A
Commande :    PASV
Réponse :    227 Entering Passive Mode (127,0,0,1,5,109)
Commande :    LIST
Réponse :    150 Connection accepted
Réponse :    226 Transfer OK
Etat :    Succès du listage du répertoire
Commande :    PWD
Réponse :    257 "/" is current directory.
Commande :    TYPE A
Réponse :    200 Type set to A
```

Du côté du serveur, vous avez des messages de logs équivalents au moment de la connexion :

```
(000001) 11/08/2006 21:16:39 - (not logged in) (127.0.0.1)>
 Connected, sending welcome message…
(000001) 11/08/2006 21:16:39 - (not logged in) (127.0.0.1)>
 USER test
(000001) 11/08/2006 21:16:39 - (not logged in) (127.0.0.1)>
 331 Password required for test
(000001) 11/08/2006 21:16:39 - (not logged in) (127.0.0.1)>
 PASS ****
(000001) 11/08/2006 21:16:39 - test (127.0.0.1)>
 230 Logged on
```

Classe FTPClient

Vous pouvez à présent réaliser la classe `FTPClient` d'aide à la connexion à un serveur, au passage de commandes et à la récupération des réponses. Vous pourrez ensuite faire une interface utilisateur adéquate et utiliser cette classe et ses méthodes.

L'utilisation de cette classe pour réaliser des opérations de base telles que la connexion à un serveur, la récupération et l'upload d'un fichier, devra être simple :

```
Dim ftp As New FTPclient("ftp://localhost", "test", "test")
ftp.Download("/MonFichier.txt", "C:\MonFichier.txt")
ftp.Upload("C:\MonFichier.txt", "/MonFichier.txt")
```

Pour développer cette classe, vous utiliserez en majeure partie l'espace de noms `System.Net`. Voici une explication de ces différents membres et méthodes.

Constructeur

Le constructeur prend en paramètre les informations de connexion :

```
Private _host As String
Private _username As String
Private _password As String

Sub New(ByVal Host As String, ByVal Username As String, _
ByVal Password As String)
        _host = Host
        _username = User
        _password = Password
End Sub
```

Méthode GetRequest

Pour effectuer une commande FTP, vous devez utiliser un objet `FtpWebRequest`. Pour récupérer facilement un tel objet, vous allez créer la méthode `GetRequest`, qui prend une adresse FTP en paramètre. Vous utiliserez cette méthode de cette façon :

```
Dim ftp As Net.FtpWebRequest =
 GetRequest(Hostname & sourceFilename)
```

Le code de la fonction est le suivant :

```
Private Function GetRequest(ByVal URI As String) _
   As FtpWebRequest
    Dim result As FtpWebRequest =
    CType(FtpWebRequest.Create(URI), FtpWebRequest)
    result.Credentials =
    New Net.NetworkCredential(Username, Password)
    result.KeepAlive = False
    Return result
End Function
```

L'objet `NetworkCredential` permet de passer les informations de connexion de l'utilisateur.

Méthode Download

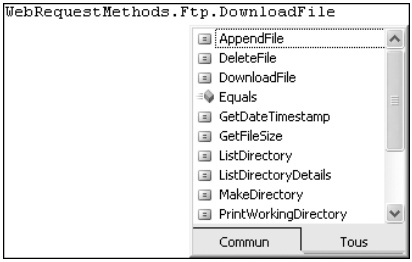
La méthode prend en paramètre le chemin relatif du fichier source sur le serveur et le chemin physique :

```
Public Function Download(ByVal sourceFilename As String, _
ByVal localFilename As String) As Boolean
```

Vous devez envoyer la commande de téléchargement au serveur. Il faut donc spécifier à la requête le type de commande, en affectant la propriété `Method` qui correspond au type de commande à effectuer :

```
ftp.Method = Net.WebRequestMethods.Ftp.DownloadFile
```

La structure `WebRequestMethods` contient l'ensemble des types de commandes possibles.



◄ **Figure 25-24 :** Gestionnaire de sites

Après avoir créé un objet `FileInfo` pour le fichier de destination, vous pouvez récupérer la réponse de la requête dans un flux et remplir le fichier :

```
Using response As FtpWebResponse = _
     CType(ftp.GetResponse, FtpWebResponse)
    Using responseStream As Stream = _
      response.GetResponseStream
        Using fs As FileStream = fi.OpenWrite
            Try
                Dim buffer(2047) As Byte
                Dim read As Integer = 0
                Do
                  read = _
                  responseStream.Read(buffer, 0, buffer.Length)
                  fs.Write(buffer, 0, read)
                Loop Until read = 0
                responseStream.Close()
                fs.Flush()
                fs.Close()
            Catch ex As Exception
                fs.Close()
                fi.Delete()
                Throw
            End Try
        End Using
        responseStream.Close()
    End Using
    response.Close()
End Using
```

Méthode Upload

D'une manière similaire, développez la méthode `Upload`, qui prend en paramètre le chemin du fichier à uploader et le chemin relatif du fichier de destination sur le serveur :

```
Public Function Upload(ByVal localFilename As String, _
Optional ByVal targetFilename As String = "") As Boolean
```

Le type de requête est cette fois `UploadFile` :

```
ftp.Method = Net.WebRequestMethods.Ftp.UploadFile
```

En outre, vous n'écrivez plus, mais vous lisez le fichier source et vous envoyez son contenu dans un flux :

```
ftp.ContentLength = fi.Length
Const BufferSize As Integer = 2048
Dim content(BufferSize - 1) As Byte
Dim dataRead As Integer
Using fs As FileStream = fi.OpenRead()
    Try
        Using rs As Stream = ftp.GetRequestStream
            Do
              dataRead = fs.Read(content, 0, BufferSize)
              rs.Write(content, 0, dataRead)
            Loop Until dataRead < BufferSize
            rs.Close()
        End Using
    Catch ex As Exception

    Finally
        fs.Close()
    End Try

End Using
```

Méthode GetResponse

Pour la plupart des commandes, le serveur renvoie un message. La méthode `GetResponse` permet de récupérer facilement ce message :

```
Private Function GetResponse(ByVal ftp As FtpWebRequest) _
   As String
    Dim result As String = ""
    Using response As FtpWebResponse = _
    CType(ftp.GetResponse, FtpWebResponse)
        Dim size As Long = response.ContentLength
```

```
        Using datastream As Stream = _
        response.GetResponseStream
            Using sr As New StreamReader(datastream)
                result = sr.ReadToEnd()
                sr.Close()
            End Using
            datastream.Close()
        End Using
        response.Close()
    End Using
    Return result
End Function
```

Vous pourrez notamment l'utiliser de la manière suivante :

```
ftp.Method = Net.WebRequestMethods.Ftp.DeleteFile
Dim strResponse As String = GetResponse(ftp)
```

25.4 Check-list

Au cours de ce chapitre, vous avez utilisé différentes classes de l'espace de noms `System.Net` pour créer une classe d'aide qui permet de réaliser les opérations FTP de base. Voici ce que vous avez également découvert :

- l'installation d'un serveur FTP ;
- la création d'un compte utilisateur FTP et la configuration de ses droits d'accès ;
- l'utilisation d'un client FTP et de ses fonctionnalités de base ;
- la connexion à un serveur FTP ;
- le téléchargement de fichiers (upload et download) ;
- la gestion des répertoires ;
- le listage des fichiers.

Annexes

26.1 Glossaire

.NET

Plateforme de développement multilangage créée par Microsoft.

.NET Compact Framework

Environnement indépendant du matériel pour l'exécution de programmes sur des périphériques informatiques soumis à des contraintes de ressources. Il hérite de l'architecture .NET Framework complète du Common Language Runtime, prend en charge un sous-ensemble de la bibliothèque de classes .NET Framework et contient des classes exclusivement conçues pour le .NET Compact Framework. Les périphériques pris en charge incluent les Assistants numériques personnels (PDA), tels que les PC de poche, les téléphones mobiles, les décodeurs, les périphériques informatiques automobiles et les périphériques incorporés personnalisés créés avec le système d'exploitation Microsoft Windows CE .NET.

Accès aux données

Partie du développement permettant le stockage et la restitution de l'information saisie.

ADO .NET

Technologie du Framework .NET permettant la gestion complète de l'accès aux données.

Appartenance (membership)

Fonctionnalité de gestion des utilisateurs d'une application web.

Application d'interface multidocument (MDI)

Application qui comporte un formulaire principal pouvant accueillir d'autres formulaires. Ces derniers, considérés comme des formulaires enfants, ne peuvent s'afficher que dans le formulaire parent. Si l'application contient des menus ou des barres d'outils, ces éléments sont généralement partagés entre le formulaire parent et ses enfants.

ASP .NET

Technologie de développement de sites et de services web du Framework .NET.

Assembly

Résultat de la compilation d'une bibliothèque de classes en un fichier avec une extension *.dll*.

Authentification

Notion de sécurité. Étape par laquelle un utilisateur doit passer avant d'accéder à une zone sécurisée. L'authentification peut se faire grâce à la saisie d'informations de connexion, comme un identifiant et un mot de passe, ou en fonction des droits d'accès attribués sur un ordinateur ou un réseau.

BCL (Base Class Library)

Librairies de classes de base, qui distribuent les blocs fondamentaux à tout développement d'application .NET

CAB

Fichier compressé contenant une application à installer.

Cache ou Mise en cache

Mise en mémoire d'informations ou de pages, qui vise à améliorer les performances d'une application.

Chaîne de connexion (connection string)

Élément à fournir pour spécifier les informations de connexion à une base de données.

Conforme CLS (CLS-compliant)

Code exposant publiquement les fonctionnalités de langue (et elles seules) figurant dans la Common Language Specification. La conformité CLS peut s'appliquer aux classes, interfaces, composants et outils. Voir aussi : Common Language Specification (CLS).

Classes unifiées

Bibliothèque de classes, d'interfaces et de types de valeurs, incluse dans le kit de développement .NET Framework SDK de Microsoft. Cette bibliothèque, qui permet d'accéder aux fonctionnalités du système, est le fondement des applications, composants et contrôles du Framework .NET. Voir aussi : Common Language Specification, Conformité CLS.

Clé étrangère

Champ d'une table indiquant une liaison entre deux tables.

Clé primaire

Champ d'une table permettant d'identifier de façon unique un enregistrement.

ClickOnce

Technologie .NET de déploiement et de mise à jour intégrée dans les applications Windows.

CLR (Common Language Runtime)

Moteur d'exécution interne du Framework Microsoft .NET. Il prend en charge le code managé en fournissant des services comme l'intégration du multilangage, la gestion du cycle de vie des objets, la sécurité d'accès au code, la gestion des ressources, le débogage.

CLS ou Common Language Specification

Spécification de langage commune. Ensemble des spécifications qu'un langage doit respecter, notamment en ce qui concerne la programmation objet, pour pouvoir être intégré à la plateforme .NET.

Code Behind

Code d'une page web dans un fichier de classe séparé du fichier correspondant à l'apparence.

Code managé (managed code)

Code exécuté par l'environnement du Common Language Runtime, et non directement par le système d'exploitation. Les applications de code managé bénéficient des services Common Language Runtime tels que les opérations de ramasse-miettes automatiques, la vérification du type au moment de l'exécution, la prise en charge de la sécurité, etc. Grâce à ces services, les applications de code managé ont un comportement uniforme, indépendant de la plateforme (et du langage).

Code non managé (unmanaged code)

Code directement exécuté par le système d'exploitation, hors de l'environnement du Common Language Runtime. Le code non managé doit fournir ses propres opérations de ramasse-miettes, de vérification du type, de prise en charge de la sécurité, etc., contrairement au code managé, qui reçoit ces services du Common Language Runtime. Voir aussi : Code managé.

Composant

Objet qui n'a pas de représentation graphique et qui n'est donc pas visible sur le rendu final du formulaire lors de son exécution.

Concepteur de projet

Interface de l'environnement Microsoft Visual Basic 2005 Express, permettant de paramétrer divers éléments associés à un projet .NET, le tout organisé par onglets.

Concepteur de vues

Espace de l'environnement Microsoft Visual Basic 2005 Express, donnant un aperçu graphique du formulaire.

Conteneur de données

Objet de l'espace de noms `System.Data`, permettant une représentation fidèle de la base de données en mode déconnecté.

Contrôle

Objet qui s'affiche, directement visible sur le formulaire en mode Design. Il existe différents types de contrôles.

Contrôle d'événement

Contrôle qui génère un événement dans le cadre d'une interaction avec l'utilisateur.

Contrôle d'affichage

Contrôle qui permet d'afficher du texte à titre d'information.

Contrôle de saisie

Contrôle qui permet à l'utilisateur de saisir des valeurs.

Conteneur

Contrôle qui peut contenir d'autres contrôles

Contrôle personnalisé

Contrôle héritant de la classe `System.Windows.Form.Control`, dont on récrit entièrement le comportement (logique métier) et le rendu graphique (il devra surcharger la méthode `Paint`).

Contrôle utilisateur ou composite

Contrôle héritant de la classe `System.Windows.Form.UserControl`, qui est composé d'un ensemble de contrôles préexistants et dont on peut personnaliser le comportement métier.

CSS (Cascading Style Sheet)

Feuille de style en cascade. Fichier de déclaration de style pour des balises HTML.

CTS (Common Type System)

Système de type commun. Spécification déterminant la façon dont le Common Language Runtime définit, utilise et gère les types.

Débogueur (debugger)

Fonctionnalité qui permet de dérouler pas à pas l'exécution d'un programme pour vérifier que le comportement du code correspond bien aux attentes du développeur. Les valeurs des variables peuvent également être inspectées. Un débogueur sert notamment à placer les points d'arrêt au niveau des sections de code par lesquelles il faut passer.

EDI (environnement de développement intégré)

Logiciel qui prend en charge une technologie de développement. Il peut proposer des fonctionnalités qui permettent de programmer des applications plus facilement, comme des éditeurs de code, des boîtes de dialogue, des Assistants, des pages d'aide, des fenêtres de configuration et différents autres outils associés (compilateur, débogueur).

Entrepôts de données

Méthodes de stockage de l'information, système de gestion de bases de données, fichiers plats, fichiers XML, etc.

Explorateur de serveurs

Fenêtre de l'environnement Microsoft Visual Basic 2005 Express affichant l'arborescence d'une base de données.

Explorateur de solutions

Fenêtre de l'environnement Microsoft Visual Basic 2005 Express affichant l'arborescence de la solution .NET ouverte.

Fournisseur (provider)

Ensemble de classes encapsulant toute la logique qui permet de mettre en place une fonctionnalité avancée pour un site web. Le modèle de base de fournisseur et les fournisseurs par défaut peuvent être étendus pour répondre à vos besoins.

Fournisseurs managés

Objets managés permettant la communication avec différents systèmes de gestion de bases de données. Ils permettent l'accès à SQL Server, Oracle ou à toute source de données de type ODBC ou OLEDB.

Framework

Ensemble de bibliothèques fournissant des briques logicielles qui permettent le développement rapide d'applications.

Gestionnaire d'événements

Code au sein d'un programme, qui répond à une action provoquée par l'utilisateur. On peut déléguer la gestion des événements à Microsoft Visual Basic 2005 Express (mots-clés `Handles` et `WithEvents`) ou le gérer soi-même par code (mots-clés `AddHandler`, `RemoveHandler`, `AddressOf`).

HTML (HyperText Markup Language)

Langage à balises hypertextes, qui permet d'écrire des pages web affichables dans un navigateur Internet.

HTTP (HyperText Transfer Protocol)

Protocole de transfert de données utilisé notamment pour accéder à des pages web ou à des services web.

IDE (Integrated Development Environment)

Voir EDI.

IL

Voir MSIL.

Intellisense

Fonctionnalité de complétion de code. L'éditeur peut ainsi proposer tour à tour les espaces de noms, puis les classes, puis les membres et enfin les paramètres disponibles en fonction du texte que l'on a commencé à saisir.

JIT (Just in Time)

Technique de compilation de code qui vise à améliorer les performances.

Librairie de classes

Ensemble de classes.

Localisation ou Globalisation

Traduction des différents composants d'une application pour l'adapter à plusieurs cultures.

Microsoft SQL Server Express 2005

Version gratuite du système de gestion de bases de données de Microsoft.

Mono

Le projet Mono vise à fournir une implémentation libre de la plateforme de développement Microsoft .NET.

MSI (Microsoft Windows Installer)

Exécutable d'installation d'un logiciel.

MSIL (Microsoft Intermediate Language)

Langage bas niveau des instructions processeur d'exécution d'un code .NET compilé, quel que soit le langage de développement utilisé.

Page maître (master page)

Page web qui sert de modèle à d'autres.

Paramètre d'application

Variables persistantes de programme qui ont une portée globale au sein de l'application. Ces variables sont dans la plupart des cas en lecture seule.

Paramètre utilisateur

Variable persistante spécifique à l'utilisateur, généralement utilisée pour stocker les préférences de l'utilisateur.

Plan de site (sitemap)

Fichier dans lequel on définit la hiérarchie des pages d'un site web (URL, titre, description…).

PostBack

Publication d'une page web, rafraîchissement.

Procédure stockée

Ordre SQL stocké et exécuté au sein du système de gestion de bases de données.

Profil

Stockage de données personnalisées relatives à un utilisateur d'un site web.

Projet Bibliothèque de classes

Type de projet .NET regroupant un ensemble de classes, généralement ordonnées par thème ou fonctionnalité, générant au final un assembly exploitable.

Projet WinForm

Projet .NET permettant de concevoir des applications Windows Forms.

Ramasse-miettes (garbage collector)

Système de gestion automatique de la mémoire. Il est responsable du recyclage des ressources mémoire préalablement allouées puis inutilisées.

Service web

Programme qui fournit des informations par le biais d'un flux XML via un protocole web.

SQL (Stuctured Query Language)

Langage utilisé pour la communication avec un système de gestion de bases de données.

Thème ASP .NET

Ensemble de fichiers spécifiant l'apparence graphique des éléments d'une page web. Il peut contenir des feuilles de style CSS, des images et des fichiers d'apparence (skin). Un site web ASP .NET peut avoir plusieurs thèmes différents.

Runtime

Moteur d'exécution de code.

Smart tag

Élément qui permet d'accéder aux tâches courantes d'un contrôle en mode Édition.

SMTP (Simple Mail Transfer Protocol)

Protocole de transfert d'e-mails.

ViewState

Fonctionnalité qui permet de stocker l'état des contrôles d'une page web entre deux publications.

Visual Web Developper

EDI de la gamme Visual Studio permettant de développer des sites web ASP .NET.

Web.config

Fichier de configuration XML d'un site web ASP .NET.

WebPart

Contrôle de page web qui peut être personnalisé par l'utilisateur.

XHTML (eXtended Markup Language)

Langage à balises étendu.

26.2 Références web

Coach VB2005

www.microsoft.com/france/msdn/vbasic/decouvrez/coach.mspx

Votre coach Visual Basic vous guide à la découverte du développement Windows sur la plateforme .NET de Microsoft. Étape par étape, suivez votre coach. Il vous accompagne au travers d'un cursus d'initiation et de prise en main des concepts, des technologies et des outils.

Blogs

Blogs des auteurs

- Patrice Lamarche : **http://blogs.developpeur.org/patrice/**.
- Antoine Griffard : **http://blogs.developpeur.org/tonio/**.
- Mauricio Díaz Orlich : **www.madd0.com/**.

▲ **Figure 26-1 :** Les blogs du réseau CodeS-SourceS

Blogs anglais

- MSDN Blogs US : **http://blogs.msdn.com/default.aspx**.
- MSDN-ASP .NET US : **http://weblogs.asp.net/**.
- Microsoft Belgique : **www.microsoft.com/belux/fr/msdn/community/blogs.mspx**.

Blogs francophones

- Blogs du réseau CodeS-SourceS : **http://blogs.developpeur.org/**.
- Microsoft France : **http://blogs.microsoft.fr/**.

Communautés

Vous trouverez dans ce qui suit, la liste des communautés, avec un bref descriptif de leurs créateurs.

ASP-PHP .NET

www.asp-php.net/

Plateforme ouverte aux rédacteurs de tous horizons, la communauté ASP-PHP .NET aborde les différents langages majeurs d'Internet, tout en restant "ni catégoriquement pour l'un, ni farouchement contre l'autre !". Regroupant une large gamme de tutoriaux, de scripts et d'articles, ASP-PHP .NET se veut pédagogique, orientée vers la formation des visiteurs débutants, et objective quant aux comparaisons technologiques. Son forum reste une aide précieuse pour tous les visiteurs du fait de la qualité des intervenants issus de technologies diverses et variées.

▲ **Figure 26-2 :** Communauté ASP-PHP .NET

C2i.fr

www.c2i.fr/

C2i.fr fut tout d'abord un site consacré à Visual Basic (4, 5 et 6). Dès juin 2000, des articles sur la plateforme .NET de Microsoft sont publiés, consacrés à toutes les technologies .NET (WinForm, WebForm, Web Service, Compact Framework…), aussi bien en VB .NET qu'en C#. Ce fut le premier site francophone développé en ASP .NET. Vous trouverez sur C2i.fr un fil quotidien d'informations sur .NET, des articles, des forums actifs où poser vos questions.

CodePPC

www.codeppc.com

La communauté CodePPC est composée de développeurs qui programment ou qui veulent programmer pour la plateforme Pocket PC. La majorité des membres est francophone, même si des contacts sont noués avec des personnes des États-Unis, du Japon et de certains pays d'Europe. Le but est de proposer des articles clairs, simples et pratiques pour aider au maximum les développeurs à créer des applications pour cette plateforme. Un forum permet aux membres de poser leurs questions et d'aider les autres. Des programmes avec leurs sources sont aussi disponibles ; leur analyse est l'un des meilleurs moyens de progresser.

CodeS-SourceS

www.codes-sources.com

La communauté de référence avec plus de quatorze langages représentés : VB, VB .NET, ASP, ASP .NET, SQL, C/C++, C#, assembleur, Flash, Delphi, ColdFusion, PHP, JavaScript, Java, Scripting Graphique, Scripting IRC, Pocket PC & Palm. Le réseau CodeS-SourceS est une communauté de développeurs francophones de cultures informatiques variées. Cette communauté regroupe plus de six cent mille membres à travers une idée simple : "partager ses connaissances".

▲ **Figure 26-3 :** Réseau de communautés CodeS-SourceS

Developpez.com

www.developpez.com

Developpez.com apporte aux développeurs une multitude de news, de cours, de tutorats, d'articles, de FAQ, d'outils, de composants et d'exemples de codes rédigés par des rédacteurs bénévoles passionnés, qui désirent partager leurs connaissances. Developpez.com a pour vocation d'aider n'importe quel développeur, quels que soient son niveau et le langage de programmation qu'il utilise, mais focalise cependant ses efforts sur la formation et l'aide au professionnels.

Dotnet-Project

www.Dotnet-Project.com

Dotnet-Project est le nouveau site de la communauté Dotnet en France. Il est le fruit de la collaboration entre Nix de CodeS-SourceS, Rédo de ASP-PHP, Kevin de Dotnet-fr, Richard de C2i et Aleksandar Lukic. Vous y trouverez des projets complets réalisés principalement avec la technologie .NET. Le plus souvent possible, leurs sources sont disponibles. Vous trouverez actuellement les starters kits traduits en français. Le but est principalement de promouvoir le travail de chacun. Pour cela, des espaces sont mis en place pour mettre en valeur les projets les plus méritants, mais également leurs auteurs. Les développeurs seront intéressés, mais aussi les chefs de projets, qui trouveront ici des solutions toutes faites ou à modifier légèrement pour les intégrer à leurs projets et passer à la vitesse supérieure !

Labo-dotnet

www.labo-dotnet.com

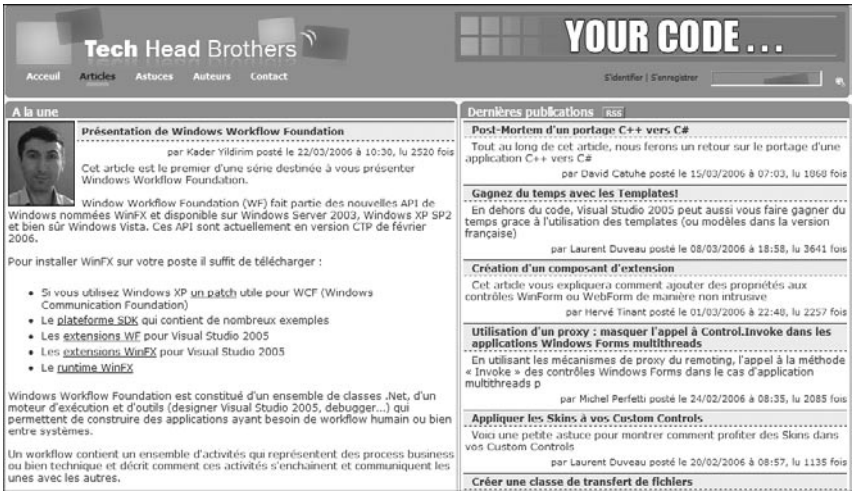
Le laboratoire des technologies de développement Microsoft .NET a été créé par des élèves ingénieurs de Sup-Info Paris (**www.supinfo.com**) dans le but de répondre à la question suivante : "Quelles compétences sont nécessaires pour faire d'un développeur un véritable expert des technologies de développement .NET ?".

◄ **Figure 26-4 :** Site du Laboratoire .NET

Tech Head Brothers

www.TechHeadBrothers.com

Tech Head Brothers est un portail francophone d'informations, d'articles et d'astuces techniques haut de gamme sur les technologies orientées développeur de Microsoft. Fondé et animé depuis 2001 par Laurent Kempé (MVP .NET) et Mathieu Kempé (MVP .NET).

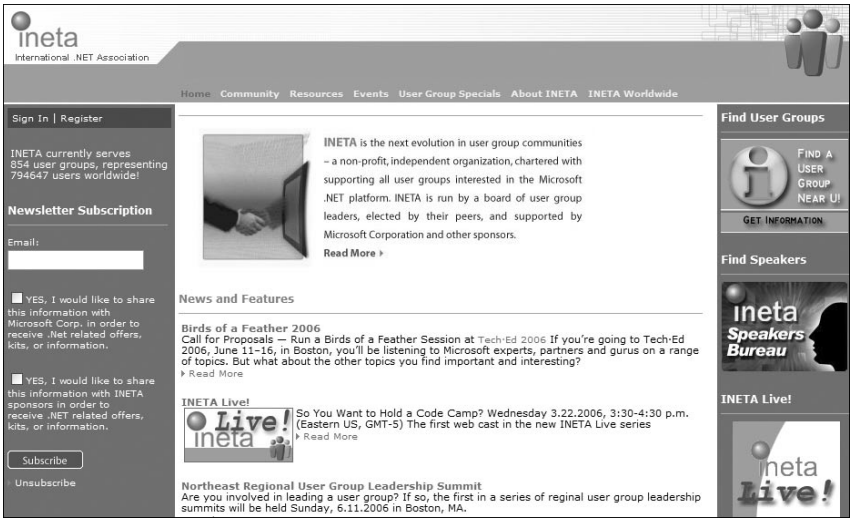


▲ **Figure 26-5 :** Communauté Tech Head Brothers

INETA (International .NET Association)

www.ineta.org

Cette association à but non lucratif a pour vocation de fédérer l'ensemble des groupes d'utilisateurs intéressés par la plateforme de Microsoft .NET. Sa mission est de proposer une assistance et des ressources utiles aux communautés ou aux groupes d'utilisateurs qui promeuvent et informent techniquement leurs membres sur les technologies .NET. L'INETA est formée d'un bureau de responsables de groupes d'utilisateurs, élus par leurs pairs. Elle est soutenue par Microsoft Corporation et autres sponsors.



▲ **Figure 26-6 :** INETA

Communautés des experts Microsoft

RD (Microsoft Regional Director)

msdn.microsoft.com/isv/rd/

Experts et partenaires privilégiés Microsoft à l'échelle mondiale, les cent vingt-neuf Regional Directors sont des experts sur les technologies de développement Microsoft, désignés par Microsoft pour intervenir lors des événements techniques en partenariat.

◄ **Figure 26-7 :** Microsoft Regional Director

MVP (Microsoft Most Valuable Professional)

https://mvp.support.microsoft.com/Default.aspx

Ce programme récompense les utilisateurs les plus actifs et les plus enthousiastes, désireux de partager leurs compétences techniques et leurs savoir-faire avec leurs pairs. Il vise à reconnaître l'apport des membres de la communauté les plus entreprenants, dont l'expertise, diffusée en ligne et par ailleurs, enrichit l'expérience de tous et permet de distinguer les communautés techniques concernant les produits Microsoft.

◄ **Figure 26-8 :** Microsoft Most Valuable Professional

MVS (Microsoft Most Valuable Student)

www.microsoft.com/france/education/sup/etudiants/mvs/default.asp

Le titre de Most Valuable Student (MVS) récompense les étudiants faisant preuve d'une expertise et d'une implication exceptionnelles autour des produits ou technologies Microsoft.

◄ **Figure 26-9 :** Microsoft Most Valuable Student

Liens Microsoft

Visual Studio

Retrouvez l'ensemble de la gamme Visual Studio 2005 à l'adresse **www.microsoft.com/france/msdn/vstudio/default.mspx**.

Microsoft Developper Network

- Librairie MSDN2 – version française : **http://msdn2.microsoft.com/fr-fr/library/default.aspx**.
- Librairie MSDN2 – version américaine : **http://msdn2.microsoft.com/en-us/library/default.aspx**.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

U

V

W

X

Z

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Notes

Composé en France par Jouve
11, bd de Sébastopol - 75001 Paris