

Par GBO Hervé

Tel : 97188083 / 95564920

E-mail : hervegbo@yahoo.fr

Edition provisoire
Février 2012

TABLE DES MATIERES

	Pages
Chapitre 1 : Généralités	
1. Historique et concepts de base	1
2. Algorithme et VB.net	5
3. Mon premier programme	13
4. Les structures de contrôle	14
Chapitre 2 : Présentation de l'environnement de développement	
1. Présentation	19
2. Les formulaires et les contrôles	25
Chapitre 3 : Les menus	
1. Ajouter un menu	33
2. Ajouter des éléments de menu standard	34
3. Activer ou désactiver les commandes de menu	34
4. Création de menus contextuels	35
Chapitre 4 : VB.net et les bases de données	
1. Rappel sur les bases de données relationnelles	36
2. Généralités sur ADO.Net	36
3. Les objets ADO.Net	36
4. Se connecter à une base de données sans écrire de code	42
5. Se connecter à une base de données en écrivant du code	46
Chapitre 5 : Gestion des Erreurs ou des exceptions	
1. Erreurs d'exécution	55
2. Bloc Try...Catch...Finally	55
3. L'objet Err et l'exception E	56
4. Exemple	56
Chapitre 6 : Les états ou rapport	
1. Ajouter un rapport	57
2. Interface utilisateur du concepteur de rapport	66
3. Masquer des éléments	87
4. Plan du rapport	87
5. Ajouter des graphiques à un rapport	88
6. Utilisation d'expressions	91
7. Ajout de code personnalisé	92
8. Ajout d'images	93
9. Fonctions intégrées	95
10. Définition des paramètres de rapport	97
Chapitre 7 : Empaquetage et déploiement	
1. Choix d'une stratégie de déploiement	115
2. Déploiement ClickOnce	115
3. Déploiement de Windows Installer	115

Chapitre 1 : GENERALITES

I. Historique et concepts de base

a. Historique

Microsoft .NET est le nom donné à un ensemble de produits et de technologies informatiques de l'entreprise Microsoft pour rendre des applications facilement portable sur Internet. Parmi ces produits, on peut citer VB.net. Voici la définition du .NET framework, communément appelé dotnet telle qu'elle apparait dans MSDN : .NET Framework est le modèle de programmation de la plate-forme .NET. Les principaux composants de .NET Framework sont le Common Language Runtime et la bibliothèque de classes .NET Framework, qui contient ADO.NET, ASP.NET et Windows Forms. .NET Framework fournit un environnement d'exécution managé, un développement et un déploiement simplifiés et l'intégration à une grande variété de langages de programmation.

Visual Basic .NET (VB.NET) est la nouvelle génération du langage Visual Basic. Bien que les syntaxes de Visual Basic .NET et Visual basic restent sensiblement proche mais, Visual Basic .NET constitue une rupture totale avec Visual Basic.

Il intègre de nouvelles fonctionnalités telles que l'héritage, les interfaces, la surcharge des fonctions, ce qui en fait un langage orienté objet à part entière.

Par ailleurs, Visual Basic .NET dispose avec Visual Studio d'un IDE et d'assistants générateurs de code avancés.

C'est à ce jour, un des langages les plus simples pour construire des applications .NET

AVANT, il y avait **Visual Basic.Net 2003** de Microsoft qui fonctionnait avec le Framework 1.1

Depuis 2005, il y a **Visual Basic 2005** de Microsoft qui fonctionne avec le Framework 2.0 et dans lequel les quelques points noirs de la version 2003 ont été corrigés.

Depuis 2008 il y a **Visual Basic 2008** de Microsoft et le Framework 3.5 qui permet d'utiliser les WPF.



Visual Basic fait partie de **VisualStudio** (payant) mais il existe aussi la version **Visual Basic Express** version allégée mais très bien et GRATUITE, en français.

b. Framework (Dotnet)

Un **framework** est un *kit* de composants logiciels structurels, qui définissent les fondations ainsi que les grandes lignes de l'organisation de tout ou partie d'un logiciel (architecture). En programmation orientée objet un framework est typiquement composé de classes mères qui seront dérivées et étendues par héritage en fonction des besoins spécifiques à chaque logiciel qui utilise le framework¹.

Les framework sont utilisés pour modeler l'architecture des logiciels applicatifs, des applications web, des middleware et des composants logiciels. Les framework sont achetés par les ingénieurs, puis ajoutés comme partie intégrante des logiciels applicatifs mis sur le marché, ils sont par conséquent rarement achetés et installés séparément par un utilisateur final.

Des tentatives de francisation du terme ont été faites. On trouve ainsi parfois les termes **cadre d'applications**, proposé par l'Office québécois de la langue française ou **cadriciel** - terme en usage depuis au moins 1997

Un framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des design patterns. L'ensemble forme un *squelette* de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du framework³.

Avec un framework orienté objets, le programmeur qui utilise le framework pourra personnaliser les éléments principaux du programme par extension, en utilisant le mécanisme d'héritage: créer des nouvelles classes qui contiennent toutes les fonctionnalités que met en place le framework, et en plus ses fonctionnalités propres, créées par le programmeur en fonction des besoins spécifiques à son programme. Le mécanisme d'héritage permet également de transformer des fonctionnalités existant dans les classes du framework⁴.

Un framework est conçu en vue d'aider les programmeurs dans leur travail. L'organisation du framework vise la productivité maximale du programmeur qui va l'utiliser - gage de baisse des coûts de construction du programme. Le contenu exact du framework est dicté par le type de programme et l'architecture cible pour lequel il est conçu³.

On trouve différents types de frameworks :

1. Framework d'infrastructure système : pour développer des systèmes d'exploitation, des interfaces graphiques, des outils de communication. (exemple : Framework .Net, NetBeans, Struts)
2. Framework d'intégration intergicielle (*middleware*) : pour fédérer des applications hétérogènes. Pour mettre à dispositions différentes technologies sous la forme d'une interface unique. (exemple : Ampoliros avec ses interfaces RPC, SOAP, XML)
3. Frameworks d'entreprise : pour développer des applications spécifiques au secteur d'activité de l'entreprise.
4. Frameworks orientés Système de gestion de contenu

Les principaux avantages de ces frameworks sont la réutilisation de leur code, la standardisation du cycle de vie du logiciel (Spécification, développement, maintenance, évolution), ils permettent de formaliser une architecture adaptée au besoin de l'entreprise. Ils tirent parti de l'expérience des développements antérieurs.

Le framework .NET

Le framework .NET est un ensemble de technologies servant de base pour développement d'applications. Il se compose de trois grandes parties:

-Le CLR ou runtime (Common Language Runtime) est l'équivalent de la JVM en Java. Avec les langages .NET vous ne compilez plus le code des programmes directement en code natif mais dans un langage intermédiaire nommé MSIL (Microsoft Intermediate Language) qui est l'équivalent du byte-code en Java. Ce code est ensuite pris en charge par le runtime : celui-ci se charge de la compilation finale au moment de l'installation ou de la première exécution du programme, gère la mémoire (Garbage collector) et la durée de vie des objets. Le .NET Framework n'étant pas sensible à un langage particulier, vous pouvez construire des programmes .NET dans de nombreux langages (environ 51 langages), y compris le langage c# introduit avec .NET (mélange entre c++ et Java). De nombreux sites comparent les langages Java et c#, ce n'est pas l'objet de ce site.

- Les bibliothèques de classes représentent une manne de fonctionnalités dans lesquelles les programmeurs des langages .NET peuvent puiser.

L'environnement de développement

Pour installer le Framework .Net il faut se munir d'une bonne connexion internet et d'un peu de patience. Tout d'abord, il faut télécharger la dernière version du Framework .Net SDK à l'heure d'aujourd'hui la version 3.0 est disponible. Ensuite il faut l'installer, et voilà vous disposez du tout dernier Framework de microsoft. Vous pouvez désormais utiliser votre meilleur éditeur et l'aide en ligne sur MSDN pour

développer vos propres applications .Net !

Microsoft à mis à disposition de ses développeurs un environnement de développement rapide d'application (Visual Studio), mais qui malheureusement est loin d'être gratuit. Cependant je vous conseille un éditeur comme Eclipse. Il existe de nombreux plugins pour ce formidable logiciel qui supporte le C# : Black-sun CSharp 2.0, Improve C# Plugin ou encore Emonic.

Common Language Runtime (CLR) est le nom choisi par Microsoft pour le composant de machine virtuelle du *framework .NET*. Il s'agit de l'implémentation par Microsoft du standard *Common Language Infrastructure* (CLI) qui définit l'environnement d'exécution des codes de programmes. Le CLR fait tourner une sorte de bytecode nommé *Common Intermediate Language* (CIL). Le compilateur à la volée transforme le code CIL en code natif spécifique au système d'exploitation. Le CLR fonctionne sur des systèmes d'exploitation Microsoft Windows.

c. Les objets et les classes

VB utilise la notion d'OBJETS'. Pour bien comprendre ce qu'est un objet, nous allons prendre des exemples dans la vie courante puis nous passerons à des exemples dans Visual Basic.
Mon gâteau est un objet, cet objet existe.



Pour créer un gâteau, j'utilise un 'moule à gâteau', ce moule a les caractéristiques du gâteau (forme, hauteur..), mais je ne peux pas manger le moule!! Le moule se nomme la **Classe**. Le gâteau est l'objet créé.

Avec le moule, je crée un ou plusieurs gâteaux.

De manière générale, une classe est une représentation abstraite de quelque chose. Tandis qu'un objet est un exemple utilisable de ce que représente la classe.

Autre exemple: Ma voiture est un **objet**, cet objet existe, on peut l'utiliser.



Ma voiture fait partie de "Les voitures", du type, du genre "Les voitures". "Les voitures" c'est une classe (Class) qui a ses caractéristiques : "Les voitures" ont une couleur, un moteur.., elles roulent en transportant des passagers... mais je ne peux pas utiliser "Les voitures", la Classe; pour me déplacer, il faut avoir un objet "voiture".

Avec la Classe je vais créer des Objets.

Pour fabriquer ma voiture, je prends les caractéristiques de la classe "Les voitures" (c'est comme un moule) et je fabrique une voiture, je la nomme 'MaVoiture'.

Dim Mavoiture as New Lesvoitures

MaVoiture est maintenant un nouvel objet de **type** 'Les voitures'.



Class --> Objet

Type 'Les voitures'--> Objet 'Mavoiture'

Un Objet est créé selon un 'modèle' qu'on appelle une Classe. On dit aussi qu'il faut **instancier** un objet à partir de la Classe.

'Mavoiture' est une **instance** de la classe 'Les voitures'. (On dit aussi une '**occurrence**' de la classe)

Remarque:

Avec la même classe on peut instancier plusieurs Objets.

Propriétés (Attributs):

Prenons *MaVoiture*.

Elle a des propriétés : une marque, une couleur, une puissance..

Pour indiquer la couleur de ma voiture on utilise la notation :

MaVoiture.couleur

Syntaxe : *Objet.Propriété* (Il y a un point entre les 2 mots) Pour modifier la couleur et avoir une voiture verte on écrit :

MaVoiture.couleur= "Vert"

Et la voiture devient verte !!

MaVoiture.Phares.Avant indique les phares avant de la voiture.

MaVoiture.Phares.Avant.Allumé indique l'état des phares (Allumé ou non) Si je fais :

MaVoiture.Phares.Avant.Allumé=True (Vrai) cela allume les phares.

Méthodes :

MaVoiture fait des choses : elle roule par exemple. Pour faire rouler la voiture j'appelle la méthode 'Roule'. *MaVoiture.Roule*

Syntaxe : *Objet.Méthode* (Il y a un point entre les 2 mots)

Si c'est possible pour cette méthode je peux indiquer à quelle vitesse la voiture doit rouler :

MaVoiture.Roule(100) j'ai ajouté un paramètre.

Le paramètre est un renseignement envoyé avec la méthode.

Il est possible parfois d'indiquer en plus si la voiture doit rouler en marche avant ou en marche arrière.

MaVoiture.Roule(10, Arriere)

Il y a donc 2 manières d'appeler la méthode Roule : avec 1 ou 2 paramètres, on dit que la méthode est surchargée;

chaque manière d'appeler la méthode s'appelle 'signature'. Première signature: *MaVoiture.Roule(100)*

Seconde signature: *MaVoiture.Roule(10, Arriere)*

Evènement:

Des évènements peuvent survenir sur un objet.

MaVoiture_démarre est un évènement, quand la voiture se met en route (si par exemple j'ai fait

MaVoiture.Roule(10, Arriere)), cet évènement *démarre* se déclenche automatiquement.

Interface et implémentation:

Ce que je vois de l'objet, c'est son interface exemple: la méthode *Roule* fait partie de l'interface d'une voiture. Par contre ce qui fait rouler physiquement la *voiture* se nomme implémentation, c'est le moteur, ce n'est ni visible ni accessible.

Si je crée une voiture je vais faire l'implémentation, je vais fabriquer le moteur, mais si je suis simplement utilisateur de l'objet voiture, je vais me contenter d'utiliser l'interface.

Visibilité:

Quand un objet est créé, il est visible et utilisable, uniquement dans la zone où il a été défini.

Relation entre Objets:

Héritage:

Une Classe (un moule) peut hériter d'une autre classe (d'un autre moule).

La classe Voiture hérite de la classe *Véhicule*, cela veut dire qu'une voiture est un véhicule; la classe voiture aurait les propriétés de la classe Véhicule.

Contenant-contenu:

On peut créer une Classe qui contient des Objets, une classe qui se compose d'objets. on parle de composition. L'objet *Voiture* contient 4 objets Roue.

On dit que l'objet **encapsule** le contenu.

Collections:

Les collections sont des groupes d'objets semblables qui peuvent être énumérés. Un parc automobile contient X Voitures; chaque voiture a un numéro d'item: *ParcVoiture.item(1)* pour la première Voiture *ParcVoiture.item(2)* pour la seconde Voiture

d. Espace de noms:

Un espace de nom regroupe des objets qui appartiennent au même 'domaine'. Cela sert à différencier des objets qui ont même nom mais ne font pas partie du même domaine: Si je parle de 'Porte' ce terme n'a pas la même signification si je suis dans l'espace de nom 'Composant électronique'(on y trouve des portes logiques) ou l'espace de nom 'Maison'.

Un espace de noms, c'est quelque chose qui regroupe des variables et des fonctions, des classes, tout ce que vous voulez dans un même ensemble. Un **namespace**, ou espace de nom (parfois aussi espace de nommage) est une zone de déclaration d'identificateurs permettant au compilateur de résoudre les conflits de noms.

Si, par exemple, 2 développeurs définissent des fonctions avec le même nom, il y aura un conflit lors de l'utilisation de ces fonctions dans un programme. Les espaces de nommage permettent de résoudre ce problème en ajoutant un niveau supplémentaire aux identificateurs.

II. Algorithme et VB.Net

a. Structure d'un programme

En Vb.net, un programme est appelé une procédure. On distingue deux types de procédure : les procédures **Sub** et les procédures **Function**

La différence entre les deux est qu'une procédure **Sub** est appelée pour exécuter simplement un programme alors qu'une **Function** est appelée pour exécuter un programme en vue de retourner une valeur à une instruction. Ainsi, pendant que l'appel d'une procédure **Sub** fait l'objet d'une instruction autonome, une procédure **Function** doit obligatoirement être appelée à l'intérieur d'une autre instruction (affectation, affichage etc.) à laquelle elle va passer sa valeur.

Algorithme

Algorithme <Nom de l'algorithme>

[Partie déclaration]

Début

[Corps de l'algorithme]

Fin

Un programme VB.NET n'utilisant pas de classe définie par l'utilisateur ni de fonctions autres que la fonction principale Main pourra avoir la structure suivante :

Procédure SUB

```
[Définition des options]
[Définition des espaces de nom]
Module <Nom du module>
Sub <Nom de la procédure> (Paramètres formels)
    [Corps de la procédure]
End sub
End Module
```

Procédure Function

```
[Définition des options]
[Définition des espaces de nom]
Module <Nom du module>
Function <Nom de la Fonction> (Paramètres formels) As <Type de la fonction>
    [Corps de la fonction]
    <Nom de la fonction>=<Valeur à retourner>
End function
End Module
```

- **[Définition des options]** permet de spécifier les directives **Option Explicit on** et **Option Strict on**

La directive **Option Explicit on** force la déclaration des variables. En VB.NET, celle-ci n'est pas obligatoire. Une variable non déclarée est alors de type Object.

La directive **Option Strict on** interdit toute conversion de types de données pouvant entraîner une perte de données et toute conversion entre les types numériques et les chaînes. Il faut alors explicitement utiliser des fonctions de conversion.

- **[Définition des espaces de nom]** permet au programme d'importer les espaces de nom dont il a besoin à l'aide de l'instruction **Imports**

Exemple : pour afficher une chaîne de caractère dans la fenêtre d'exécution de VB.Net, on écrira :

System.Console.Out.WriteLine(unechaîne)

où **System** est l'espace de noms contenant la classe **Console**.

En important l'espace de noms **System** avec une instruction **Imports**, VB.NET l'explorera systématiquement lorsqu'il rencontrera une classe qu'il ne connaît pas. Il répétera la recherche avec tous les espaces de noms déclarés jusqu'à trouver la classe recherchée. On écrira alors :

Imports System

Et dans le corps de la procédure,

Console.Out.WriteLine(unechaîne)

- En algorithmique, la déclaration des variables et constantes se fait avant le corps du programme, alors qu'en VB, les déclarations se font dans le corps de la procédure ou de la fonction.
- La clause **As <Type de la fonction>** dans une fonction VB.net permet de spécifier le type de la valeur qui sera retournée par la fonction.
- L'instruction **<Nom de la fonction>=<Valeur à retourner>** permet de spécifier la valeur que retournera la fonction.

b. Déclaration de variables et des constantes

Une variable est un espace mémoire nommé qu'un programme réserve dans la mémoire centrale de l'ordinateur en vue de stocker une information à traiter ou résultant d'un traitement et qui pourra être

modifiée tout au long de l'exécution du programme.

La longueur maximale du nom d'une variable est de 255 caractères. Notez que ce nom doit obligatoirement commencer par une lettre. Si vous ne déclarez pas une variable, Visual Basic se chargera d'affecter par défaut un type de variable (*Variant*) à celle-ci. Une variable du type *Variant* peut aussi bien recevoir des données numériques que des chaînes de caractères. Tout dépend de ce que vous avez affecté à cette variable.

▪ Les types des variables

A présent, observons de plus près les différents types de variables :

Type de données	Mot clé	Occupe	Limite de valeurs
Octet	Byte	1 octet	0 à 255
Logique	Boolean	2 octets	True(-1) ou False(0)
Entier	Integer	2 octets	-32 768 à 32767
Entier long	Long	4 octets	-2 147 483 648 à 2 147 483 647
Décimal simple	Single	4 octets	Nombre réel avec 7 chiffres après la virgule
Décimal double	Double	8 octets	Nombre réel avec 15 chiffres après la virgule
Monétaire	Currency	8 octets	Nombre réel avec 15 chiffres avant la virgule et 4 chiffres après la virgule
Date	Date	8 octets	1er janvier 100 au 31 décembre 9999
Objet	Object	4 octets	Toute référence à des types Object
Chaîne de caractères	String	10 octets + longueur de chaîne	Chaîne de caractère dont la longueur ne doit pas excéder 2^31 caractères
Variant (avec chiffres)	Variant	16 octets	toute valeur numérique jusqu'à l'étendue d'un double
Variant (avec lettres)	Variant	22 octets+longueur de chaîne	Même étendue que pour un String de longueur variable
Défini par l'utilisateur	Type	-	L'étendue de chaque élément est la même que son type de données

Notez que les types de variables les plus utilisées sont : String, Integer, Long, Single, Double et Currency.

▪ Instruction DIM

Pour utiliser des variables, il est normalement obligatoire de les prédéfinir, soit dans la section Déclarations de la liste déroulante Objet, soit en début ou à l'intérieur de la procédure. Un programme où les variables sont bien déclarées rend un programme plus facile à comprendre, à lire et surtout à corriger en cas d'erreurs. Certes, il n'est pas obligatoire de les déclarer mais faites-le quand même, c'est un conseil. Si vous êtes prêt à déclarer une variable que vous voulez utiliser mais que vous êtes un

étourdi, alors, utilisez simplement l'instruction *Option Explicit* (à placer dans la section Déclarations de la liste déroulante Objet) qui vous oblige à chaque fois à déclarer toutes vos variables avant de pouvoir exécuter l'application.

Pour déclarer une variable, on utilise l'instruction **Dim** suivi du nom de la variable puis du type de la variable.

Exemple :

Dim DateNaissance

Dim Message, Titre As String

- Remarquez que la 1ère déclaration ne contient pas d'information sur le type de variable. Si vous déclarez une variable sans donner d'information sur le type de variable que c'est, alors, cette variable (DateNaissance) aura par défaut le type Variant.
- La 2ème déclaration est par contre explicite. Vous pouvez aussi mettre 2 variables sur une même ligne à condition que le type de variable soit le même pour les 2 et en les séparant par une virgule.

c. Portée des variables

En général, toute variable déclarée a une portée limitée. Cette même variable a une valeur nulle au départ. De plus, elle ne s'applique pas forcément à toutes les procédures d'une application. Tout dépend de 2 éléments : *la manière de déclarer* et *l'emplacement* de la variable.

- dans une procédure, si vous déclarez une variable à l'aide de l'instruction *Dim*, sa portée se trouve limitée seulement à cette procédure. On dit que la variable est locale. Elle est donc initialisée à chaque appel de la procédure et détruite lorsque celle-ci se termine (à moins que vous remplacez le mot *Dim* par *Static*). Elle n'est pas accessible en dehors de la procédure. Vous pouvez remplacer l'instruction *Dim* par *Private*, les deux termes étant équivalentes s'ils sont placés à l'intérieur d'une procédure.
- Si vous déclarez une variable dans la section Général/Déclarations d'une feuille ou d'un module en utilisant l'instruction *Dim* (ou *Private*), la variable est dite locale au module. Cette variable est disponible pour toutes les procédures de la feuille ou du module, mais pas pour les autres feuilles ou modules du projet.
- Enfin, si vous déclarez une variable dans la section Général/Déclarations d'un module (et non d'une feuille) en utilisant l'instruction *Public* au lieu de *Dim*, elle devient accessible par toutes les feuilles et tous les modules de l'application. On dit qu'elle est globale.

d. Variables indicées

On les appelle aussi des tableaux. Ils peuvent être des tableaux de nombres, de chaînes, de booléens, bref, de tout ce qu'on veut. Quant on crée un tableau, soit on sait d'avance combien d'éléments il va englober, soit on veut qu'il soit élastique (mais cela se paye bien sûr par une perte de rapidité à l'exécution).

Première méthode pour définir un tableau

Tableau à une dimension

Les tableaux de valeurs sont utiles lorsque l'on manipule des données en tout genre. Ainsi, dans un carnet d'adresses, vous pouvez stocker toutes les adresses dans un tableau structuré. Par exemple :

Dim Adresses (1 to 50) As String

Adresses(1) = "12, avenue de la Fayette, Paris"

.....

Adresses(50) = "4, rue de la Paix, Paris"

Comme vous pouvez le constater, la définition d'un tableau ressemble beaucoup à celle d'une variable à la seule différence que vous devez donner une limite aux nombres de valeurs contenues dans un tableau.

Tableaux à deux dimensions

Voici un exemple à 2 dimensions:

```
Dim Adresses (1 to 50, 1 to 50) As String
Adresses(1,0) = "Vincent"
Adresses(1,1) = "12, Rue de la paix, Paris"
```

.....

```
Adresses(50,0) = "Philip"
Adresses(50,50) = "7, Boulevard de la Villette"
```

Notez qu'avec les fonctions **LBound** et **UBound**, vous pouvez obtenir les limites inférieures et supérieures d'un tableau.

Deuxième méthode pour définir un tableau

Il existe une autre façon de définir un tableau :

```
Dim Adresses(50) As String
Adresses(0) = "12, avenue de la Fayette, Paris"
```

.....

```
Adresses(49) = "4, rue de la Paix, Paris"
```

Avec cette méthode, les limites du tableau ne sont pas définies. Notez que le nombre d'adresses est de 50 mais la numérotation va de 0 à 49 et non de 1 à 50. Il existe cependant un moyen pour que la numérotation commence à 1 : c'est en utilisant l'instruction *Option Base* (par exemple: *Option Base = 1*) qu'il faut placer dans la section Déclarations.

Troisième méthode pour définir un tableau

Une autre méthode consiste en la création de tableaux dynamiques. Pour comprendre cette méthode, prenons un exemple :

```
Dim Adresse() As String
Dim NbAdresses As Integer
'Nouvelle adresse
NbAdresses = nbAdresses + 1
Redim Adresse (NbAdresses)
Adresse (NbAdresses) = "75, Rue Lecourbe, Paris"
```

Ainsi, à chaque fois que l'on lance cette procédure, le nombre total d'adresses (NbAdresses) est incrémenté (augmenté) de 1 et la nouvelle adresse est placée à la fin du tableau. En effet, *Redim* a pour effet de reformater le tableau, et donc, tout ce qui était dans le tableau est effacé sauf la dernière valeur entrée.

Cependant, il existe un moyen de conserver les données déjà présentes dans le tableau. Pour cela, on fait appel à l'instruction *Preserve* :

```
Redim Preserve Adresse (NbAdresses)
```

Enfin, pour terminer, il est possible d'effacer toutes les données d'un tableau. Pour cela, on utilise l'instruction *Erase* suivi du nom du tableau.

e. Tester le type d'une variable

Il peut arriver qu'au cours d'un programme, vous ayez besoin de savoir quel type de données contient une variable de type Variant. Dans ce cas, VB met à votre disposition une série de mots-clés pouvant donner le type de la variable. Exemple :

If IsNumeric(Tampon) ou If IsDate(Tampon) =false etc.

Vous trouverez la liste de ces **mots-clés** dans l'Aide. Alors, de deux choses l'une : ou le type est satisfaisant : on peut ensuite employer cette variable Tampon comme prévu, ou basculer sa valeur dans une variable déclarée dans le type approprié. Dans le cas contraire, on envoie un message à l'utilisateur et on lui fait recommencer la saisie.

f. Les constantes

Contrairement aux variables dont les valeurs diffèrent souvent, les constantes ont des valeurs fixes. Mais tout comme les variables, on affecte aux constantes, un nom et une valeur qui elle est fixe. De plus, les constantes se définissent de la même façon que les variables. Tout dépend donc de l'endroit où est définie la constante. Le mot **Const** peut être précédé de l'option *Public* pour que toutes les feuilles et modules de l'application puissent y accéder.

Attention cependant à ne pas affecter à une constante des noms identiques aux constantes prédéfinies dans Visual Basic !

Prenons l'exemple du taux de TVA :

Const TVA = 20.6

TotalTTC= PrixHorsTaxe * (1 + (TVA / 100))

g. Les instructions et les opérateurs de base

▪ Les instructions de base

○ Instruction d'affichage : *MsgBox()*

MsgBox() est une instruction qui affiche une boîte de message. Une boîte de message est un genre de petit formulaire dans lequel on trouve une icône, un message, et au moins un bouton de commande qui permet de laisser le temps à l'utilisateur de lire le message, et de refermer lui même la boîte de message. Exemple d'une boîte de message créée à l'aide de l'instruction **MsgBox()**.



Syntaxe

MsgBox(Message, Boutons, Titre)

Paramètres

Message

Requis. Expression **String** affichée comme message de la boîte de dialogue. La longueur maximale de l'argument Prompt est d'environ 1 024 caractères selon la largeur des caractères utilisés. Si Prompt se compose de plusieurs lignes, vous pouvez séparer les lignes à l'aide d'un retour chariot (13 **Chr**()), d'un saut de ligne (10 **Chr**()) ou d'une combinaison des deux caractères (13 **Chr**()& 10 **Chr**()).

Boutons

Facultatif. Expression numérique qui représente la somme des valeurs spécifiant le nombre et le type de boutons à afficher, le style d'icône à utiliser, l'identité du bouton par défaut, ainsi que la modalité du message. Si l'argument Boutons est omis, la valeur par défaut est zéro.

Titre

Facultatif. Expression **String** affichée dans la barre de titre de la boîte de dialogue. Si l'argument Titre est omis, le nom de l'application est placé dans la barre de titre.

Paramètres

Les valeurs d'énumération **MsgBoxStyle** sont répertoriées dans le tableau suivant.

Membre	Valeur	Description
OKOnly	0	Affiche le bouton OK uniquement.
OKCancel	1	Affiche les boutons OK et Annuler.
AbortRetryIgnore	2	Affiche les boutons Abandonner, Réessayer et Ignorer.
YesNoCancel	3	Affiche les boutons Oui, Non et Annuler.
YesNo	4	Affiche les boutons Oui et Non.
RetryCancel	5	Affiche les boutons Réessayer et Annuler.
Critical	16	Affiche l'icône Message critique.
Question	32	Affiche l'icône Requête d'avertissement.
Exclamation	48	Affiche l'icône Message d'avertissement.
Information	64	Affiche l'icône Message d'information.
DefaultButton1	0	Le premier bouton est le bouton par défaut.
DefaultButton2	256	Le deuxième bouton est le bouton par défaut.
DefaultButton3	512	Le troisième bouton est le bouton par défaut.
ApplicationModal	0	L'application est modale. L'utilisateur doit répondre au message avant de poursuivre le travail dans l'application en cours.
SystemModal	4096	Le système est modal. Toutes les applications sont interrompues jusqu'à ce que l'utilisateur réponde au message.
MsgBoxSetForeground	65536	Spécifie la fenêtre de message comme fenêtre de premier plan.
MsgBoxRight	524288	Le texte est aligné à droite.
MsgBoxRtlReading	1048576	Spécifie que le texte doit apparaître de droite à gauche sur les systèmes hébraïques et arabes.

Le premier groupe de valeurs (0-5) décrit le nombre et le type des boutons affichés dans la boîte de dialogue. Le second groupe (16, 32, 48, 64) décrit le style de l'icône. Le troisième groupe (0, 256, 512) détermine quel bouton est la valeur par défaut. Le quatrième groupe (0, 4096) détermine la modalité du message, et le cinquième groupe indique si la fenêtre de message est la fenêtre de premier plan, avec l'alignement et l'orientation du texte. Au moment d'additionner ces nombres pour obtenir la valeur finale de l'argument Buttons, ne sélectionnez qu'un seul nombre dans chaque groupe.

Remarque

On peut vouloir récupérer la réponse de l'utilisateur c'est-à-dire savoir sur quel bouton, l'utilisateur a appuyé. Dans ce cas, MsgBox est utilisé comme une fonction c'est-à-dire :

<Variable>=MsgBox(Message, Boutons, Titre)

Les valeurs retournées sont les suivantes :

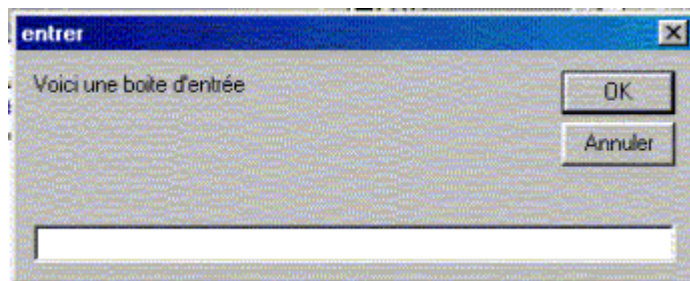
Constante	Valeur
OK	1

Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

○ **Instruction de saisie : *InputBox()***

MsgBox() est une instruction qui permet au logiciel d'informer l'utilisateur sur une action quelconque. ***InputBox()*** est une fonction qui permet à l'utilisateur d'entrer une valeur ou un texte par l'intermédiaire d'une boîte d'entrée.

Une boîte d'entrée ou boîte ***InputBox()*** est une boîte de message ***MsgBox()*** qui en plus inclut un champs que l'utilisateur doit renseigner. Une boîte d'entrée ***InputBox()*** n'affiche pas d'icône et elle ne comporte que 2 boutons, *OK* et *Annuler*.



Syntaxe

<Variable> = InputBox(<message>, <titre>, <Valeur par défaut>, <Xpos>, <YPos>)

Paramètres

<**message**> : Expression **String** requise affichée comme message de la boîte de dialogue. La longueur maximale de l'argument Prompt est d'environ 1 024 caractères selon la largeur des caractères utilisés. Si elle se compose de plusieurs lignes, vous pouvez séparer les lignes à l'aide d'un retour chariot (13 **Chr()**), d'un saut de ligne (10 **Chr()**) ou d'une combinaison des deux (13 **Chr()** & 10 **Chr()**).

<**Titre**>

Facultatif. Expression **String** affichée dans la barre de titre de la boîte de dialogue. Si elle est omise, le nom de l'application est placé dans la barre de titre.

<**Valeur par défaut**>

Facultatif. Expression **String** affichée par défaut dans la zone de texte en l'absence de toute autre valeur. Si elle est omise, la zone de texte qui s'affiche est vide.

<**XPos**>

Facultatif. Expression numérique qui spécifie, en pixels, la distance entre le bord gauche de la boîte de dialogue et le bord gauche de l'écran.

<**YPos**>

Facultatif. Expression numérique qui spécifie, en pixels, la distance entre le bord supérieur de la boîte de dialogue et le haut de l'écran. Si les arguments XPos et YPos sont omis, la boîte de dialogue est centrée par rapport à l'écran.

Notes

Si l'utilisateur clique sur **Annuler**, une chaîne de longueur nulle est retournée.

Si vous omettez certains arguments, vous devez quand même conserver la virgule de séparation correspondante.

1 Cm = 567 Twips, 1 Pouce = 1440 Twips.

○ L'affectation

Elle permet d'affecter à une variable, une valeur. Elle est représentée en VB.net par le symbole « = » à ne pas confondre avec l'opérateur de comparaison « égalité »

Exemple :

Dim I as integer

I=0

▪ *Les opérateurs de base*

Opérateur	Exemple
+(<i>Addition</i>)	6 + 2 Renvoie la Valeur 8
*(<i>Multiplie</i>)	6 * 2 Renvoie la Valeur 12
-(<i>Soustraire</i>)	6 - 2 Renvoie la Valeur 4
/ (<i>Diviser</i>)	6 / 2 Renvoie la Valeur 3 Ou 9 / 2 Renvoie la Valeur 4,5
\ (<i>Division entière</i>)	6 \ 2 Renvoie la Valeur 3 Ou 9 \ 2 Renvoie la Valeur 4
Mod (<i>Reste</i>)	9 Mod 2 Renvoie la Valeur 1
^(<i>exposant</i>)	6 ^ 2 Renvoie la Valeur 36 Ou 2 ^ 3 Renvoie la Valeur 8 (2 x 2 x 2)
+ ou &(<i>Concaténation</i>)	"Bon" + "jour" Renvoie "Bonjour" Ou Bon & "jour" Renvoie "Bonjour"

Un ordre de priorité est respecté dans les opérations complexes, avec ou sans parenthèses.

Priorité 1 Les parenthèses ()

Priorité 2 l'exposant ^

Priorité 3 *, / , \ , Mod

Priorité 4 + , -

III. Mon premier programme

Exercice

Ecrire le programme visual basic qui affiche la somme de deux nombres réels saisis au clavier

Résolution

Dans le menu projet, cliquer sur la commande Ajouter un module

Module Module1

Sub Somme()

'déclaration des variables

Dim Nb1, Nb2, S As Double

Nb1 = InputBox("Donner le premier nombre")

Nb2 = InputBox("Donner le second nombre")

S = Nb1 + Nb2

MsgBox("la somme des deux nombres est : " & CStr(S))

End Sub

'on crée la procédure principale qui va appeler la procédure Somme()

Sub Main()


Somme()

End Sub

End Module

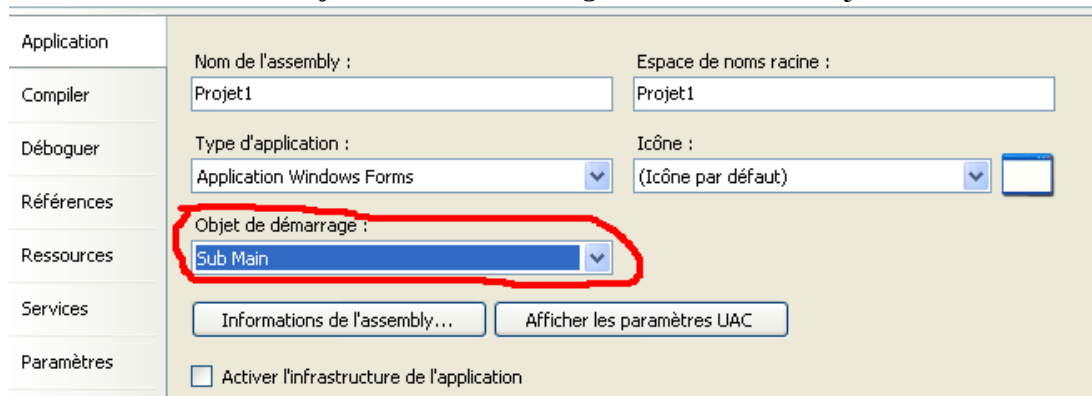
Exécuter ensuite le projet

Parmi les procédures **Sub** que Visual basic vous permet de créer, il y a une procédure nommée **MAIN** (Procédure principale). C'est à partir de cette procédure que toutes les autres procédures peuvent être exécutées. Le mot **MAIN** est donc un mot réservé en Visual basic pour nommer une procédure principale.

Pour exécuter un projet écrit en Visual basic, il faut cliquer sur le bouton **Exécuter**  dans la barre d'outils ou choisir dans le menu **Déboguer**, la commande **Démarrer le débogage**. Visual basic exécutera soit la procédure principale nommée **MAIN**, soit un des nombreux formulaires que vous aurez créés.

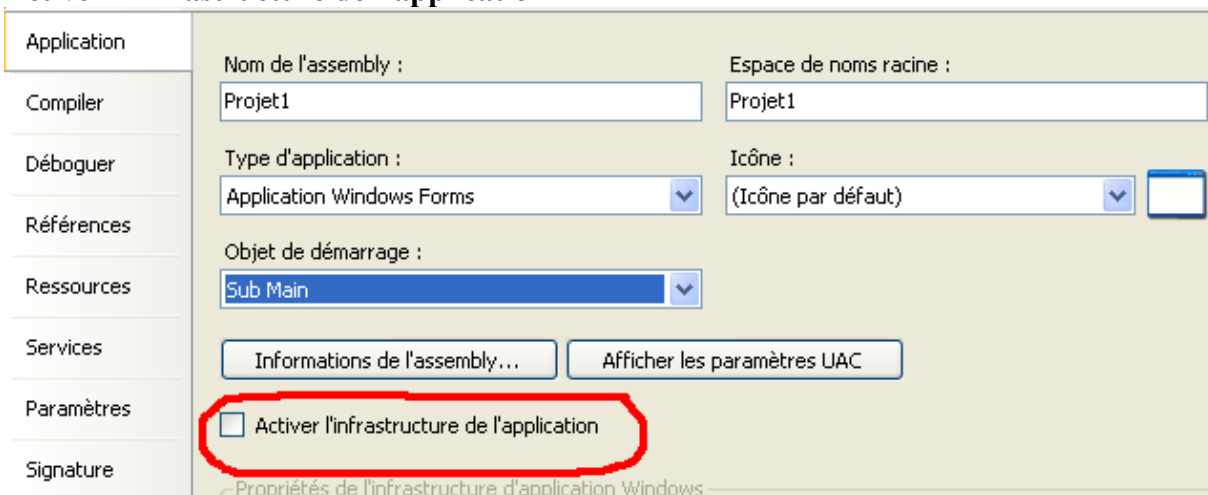
Pour spécifier à l'ordinateur l'objet à exécuter au démarrage du projet, il faut :

- Choisir dans le menu **Projet**, la commande **Propriétés de <nom du projet>**. La fenêtre **Propriété du projet** s'ouvre.
- Dans la zone **Objet de démarrage**, choisir l'objet de démarrage désiré



The screenshot shows the 'Project Properties' dialog box for 'Projet1'. The 'Objet de démarrage' (Startup Object) dropdown menu is highlighted with a red circle and contains 'Sub Main'. Other fields include 'Nom de l'assembly' (Projet1), 'Espace de noms racine' (Projet1), 'Type d'application' (Application Windows Forms), and 'Icône' (Icône par défaut). There are buttons for 'Informations de l'assembly...' and 'Afficher les paramètres UAC', and a checkbox for 'Activer l'infrastructure de l'application'.

Pour pouvoir choisir **Sub Main** comme objet de démarrage, vous devez désactiver la case à cocher **Activer l'infrastructure de l'application**



The screenshot shows the 'Project Properties' dialog box for 'Projet1'. The 'Activer l'infrastructure de l'application' checkbox is highlighted with a red circle and is unchecked. Other fields include 'Nom de l'assembly' (Projet1), 'Espace de noms racine' (Projet1), 'Type d'application' (Application Windows Forms), and 'Icône' (Icône par défaut). The 'Objet de démarrage' (Startup Object) dropdown menu contains 'Sub Main'. There are buttons for 'Informations de l'assembly...' and 'Afficher les paramètres UAC'.

IV. Les structures de contrôle

a. Les structures alternatives

- La structure : If...Then...Else...end If

Voici un exemple de structure simple avec l'instruction *If* :

```
If <condition> Then
    Traitement 1
Else
    Traitement 2
End if
```

Interprétation : Si la condition est vérifiée alors les instructions 1 sont exécutées sinon les instructions 2 sont exécutées à la place. (Notez que l'on peut utiliser des opérateurs logiques *And* (et) et *Or* (ou) pour que plusieurs conditions doivent d'abord être vérifiées avant de pouvoir exécuter les instructions suivantes.). Le mot *Else* et les instructions qui suivent ne sont pas obligatoire.

Voici un autre exemple de structure avec *If* mais un peu plus complexe:

```
If Condition 1 Then
    Traitement 1
ElseIf Condition 2 Then
    Traitement 2
Else
    Traitement X
End If
```

Interprétation : Si la condition 1 est vérifiée alors les instructions 1 sont exécutées. Sinon si la condition 2 est vérifiée alors les instructions 2 sont exécutées. Sinon, si aucune de ces deux conditions ne sont vérifiées alors les instructions X sont exécutées.

- La structure Select Case...End Select

Lorsque l'on doit effectuer toute une série de tests, il est préférable d'utiliser la structure Select Case...End Select au lieu de If...Then...End if.

Syntaxe

```
Select Case valeur à tester
    Case valeur 1
        <Traitement 1>
    ...
    [Case valeur-N
        <Traitement N>
    [Case Else
        Traitement si autrement]]
End Select
```

La syntaxe de l'instruction **Select Case** comprend les éléments suivants :

Élément	Description
<i>valeur à tester</i>	Toute expression (variable, champs, propriétés...) de type numérique ou chaîne de caractère
<i>valeur-N, valeur 1</i>	Une quelconque des valeurs que peut prendre la <i>valeur à tester</i> . Si c'est un intervalle, on écrira : <i>valeur_i To : valeur_n</i> ou <i>Is opérateur de comparaison constante</i> . Le mot clé To indique une plage de valeurs. Si vous utilisez To , la valeur la plus petite (<i>valeur_i</i>) doit figurer avant To . Utilisez le mot clé Is avec les opérateurs de

	comparaison pour indiquer une plage de valeurs. S'il n'est pas indiqué, le mot clé Is est inséré automatiquement.
<i>Traitement-n</i>	Facultatif. Une ou plusieurs instructions exécutées si <i>valeur à tester</i> correspond à l'un des éléments de <i>valeur-n</i> .
<i>Elsestatements</i>	Facultatif. Une ou plusieurs instructions exécutées si <i>valeur à tester</i> ne correspond à aucun élément de la clause Case .

Remarque

On peut utiliser à la place de la structure Select case, la structure If ... ElseIf
Ainsi, les deux exemples suivants sont équivalents :

Select Case Semaine Case 1 Jour = "Lundi" Case 2 Jour = "Mardi" Case 3 Jour = "Mercredi" Case 4 Jour = "Jeudi" Case 5 Jour = "Vendredi" Case 6 Jour = "Samedi" Case 7 Jour = "Dimanche" Case Else MsgBox "erreur", , "Note" End Select	If Semaine = 1 Then Jour = "Lundi" ElseIf Semaine = 2 Then Jour = "Mardi" ElseIf Semaine = 3 Then Jour = "Mercredi" ElseIf Semaine = 4 Then Jour = "Jeudi" ElseIf Semaine = 5 Then Jour = "Vendredi" ElseIf Semaine = 6 Then Jour = "Samedi" ElseIf Semaine = 7 Then Jour = "Dimanche" Else MsgBox "erreur", , "Note" End If
---	---

Interprétation: Si la variable Semaine vaut 1, alors la variable Jour reçoit la valeur Lundi. Si la variable Semaine vaut 2, alors la variable Jour reçoit la valeur Mardi. (Etc...). Si la variable Semaine ne reçoit aucune valeur comprise entre 1 et 7, alors un message indiquant une erreur est affiché.

b. Les structures répétitives

- **La structure For..Next**

La structure For...Next est utile lorsqu'on doit répéter plusieurs fois la même instruction.

Syntaxe

For *compteur* = *valeur initiale* **To** *valeur finale* [**Step** *pas*]

[*instructions*]

[Exit For]

[*instructions*]

Next [*compteur*]

La syntaxe de l'instruction **For...Next** comprend les éléments suivants :

Élément	Description
---------	-------------

<i>Compteur</i>	<u>Variable</u> numérique utilisée comme un compteur de boucles. Cette variable ne peut être ni une variable de type <u>Boolean</u> , ni un élément de <u>tableau</u> .
<i>Valeur initiale</i>	Valeur initiale du <i>compteur</i> .
<i>Valeur finale</i>	Valeur finale du <i>compteur</i> .
<i>Pas</i>	Facultatif. Valeur d'incrémentation du <i>compteur</i> après chaque exécution de la boucle. Si aucune valeur n'est indiquée, l'argument <i>pas</i> prend par défaut la valeur 1.
<i>instructions</i>	Facultatif. Une ou plusieurs instructions entre For et Next à exécuter le nombre de fois indiqué.

Exemple :

```
For J = 1 To 10
```

```
T(J) = J
```

```
Next
```

Interprétation : Dans cette boucle, l'instruction "T(J)=J" est répétée 10 fois c'est à dire jusqu'à ce que J soit égale à 10. A chaque passage, l'instruction "T(J)=J" affecte à T(J) les valeurs 1 à 10. Pour que l'instruction soit exécutée une fois sur deux, vous pouvez utiliser l'instruction *Step* :

```
For J = 1 To 10 Step 2
```

```
T(J) = J
```

```
Next
```

Interprétation : Dans cette boucle, l'instruction "T(J)=J" affecte à T(J) les valeurs 1,3,5,7,9.

- **Les structures Do...Loop et While...Wend**

Avec ces boucles, le nombre de fois où sont répétées les instructions est indéfini.

Syntaxe Do...Loop

```
Do [{ While | Until } condition]
```

```
[statements]
```

```
[Exit Do]
```

```
[statements]
```

```
Loop
```

Vous pouvez également utiliser la syntaxe suivante :

```
Do
```

```
[statements]
```

```
[Exit Do]
```

```
[statements]
```

```
Loop [{ While | Until } condition]
```

Syntaxe While... Wend

Exécute une série d'instructions dans une boucle aussi longtemps que la valeur d'une condition est **vrai**.

```
While condition
```

```
[instructions]
```

```
Wend
```

Élément	Description
<i>Condition</i>	Facultatif. <u>Expression numérique</u> ou <u>expression de chaîne</u> vraie (True) ou fausse (False). Si la valeur de <i>condition</i> est <u>Null</u> , elle est considérée comme fausse (False).

<i>instructions</i>	Une ou plusieurs instructions répétées tant que <i>condition</i> est True , ou jusqu'à ce qu'elle le devienne.
---------------------	---

Les deux exemples qui suivent sont équivalents :

i = 1	i = 1
Do	While i < 10
T(i) = i	T(i) = i
i = i + 1	i = i + 1
Loop Until i > 10	Wend

Interprétation : La variable i est initialisée à 1 au départ. Les instructions qui suivent sont exécutées jusqu'à ce que i soit supérieure à 10. A chaque passage, on affecte à la variable T d'indice i la valeur de i.

EXERCICES

- 1 - Ecrire le programme qui affiche dans une boîte de dialogue, le triangle de pascal d'un nombre saisi au clavier.
- 2 - Ecrire le programme qui affiche dans une boîte de message le montant d'une facture sachant que :
 - La facture peut contenir plusieurs produit
 - Les prix unitaire et quantité de produits achetés sont saisis par l'utilisateur
 - Montant de la facture = $\sum(\text{quantité achetée} * \text{prix unitaire})$
- 3 - Que deviendrait ce programme si on ajoutait :
 - Une remise de 10% est appliquée si le montant brut est supérieur à 12500
 - Net à payer = Montant brut – Remise
 - Montant brut = $\sum(\text{quantité achetée} * \text{prix unitaire})$
 - Remise = Montant brut * 10%

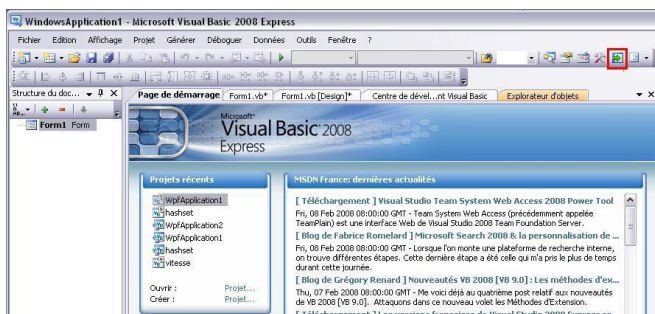
Chapitre 2 : Présentation de l'environnement de développement

I. Présentation

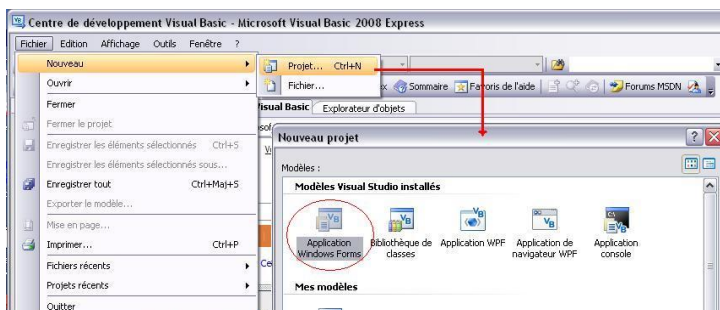
C'est l'**Integrated Development Environment (IDE)**: Environnement de développement intégré de Visual Basic Express 2008 de Microsoft. Il permet de dessiner l'interface (les fenêtres, les boutons, List, Image...) et d'écrire le code VB. Chez nous, on peut aussi dire **EDI** (Environnement de Développement Intégré). L'IDE de Visual Basic 2008 est identique à celle de VB 2005, bien meilleur que celle de VB 2003 et l'**'Edition Express'** (version légère par rapport à Visual Studio) est GRATUITE.

a. Fenêtre Projet.

Quand on lance VB.net 2008, on ouvre l'IDE dans laquelle la fenêtre centrale charge la page du centre de développement Visual Basic de MSDN (site Microsoft);



On constate que les diverses fenêtres sont accessibles par des onglets. Pour créer un nouveau projet Visual Basic, il faut choisir 'Créer' à gauche ou passer par le menu '**Fichier**' puis '**Nouveau**' puis '**Projet**' ou **Nouveau Projet**. La fenêtre suivante s'ouvre:



On a le choix à partir de VB 2008 de créer l'interface utilisateur: En Windowsforms (basé sur GDI+), interface habituelle, bien connue ou en WPF interface vectorielle élaborée n'existant pas avant VB 2008.

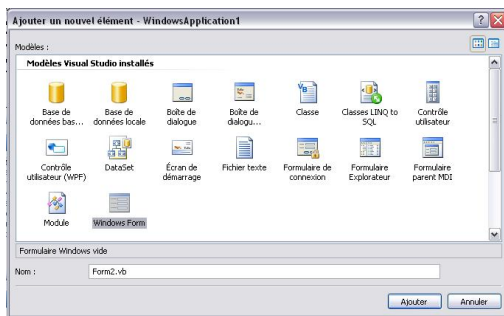
b. Interface 'Windows Forms'

Choisir l'icône '**Application Windows forms**', puis donner un nom au projet, enfin cliquer sur '**Ok**'. (Le chemin de l'emplacement du projet n'est pas modifiable ici, il est par défaut 'C:\Documents and Settings\Nom Utilisateur\Mes documents\Visual Studio 2008\ Projects\MonProjet')

Dans un nouveau projet, créer ou ajouter une fenêtre 'Windows Form':

Pour ajouter une fenêtre (un formulaire) **Menu Project, Ajouter un formulaire Windows** ('Add a

WindowsForms' en version anglaise):

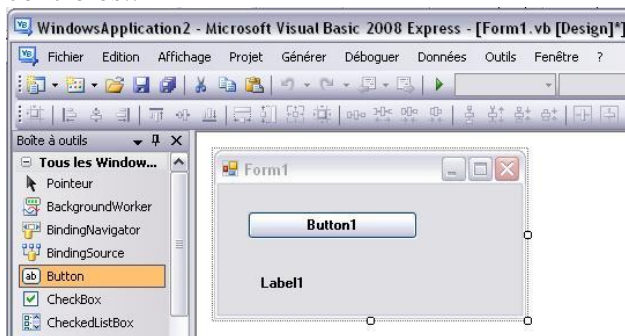


Cliquer sur Windows Form, une fenêtre (un formulaire) Form2 vide apparaît (Form1 était le nom du premier formulaire).

Il y a des fenêtres toutes faites pour accélérer le travail (les templates) comme les 'Ecran de démarrage' les 'FormulaireExplorateur'...

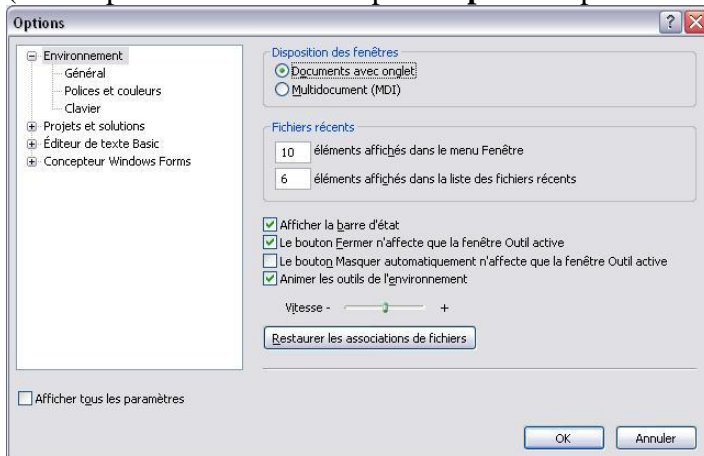
c. Designer:

La zone de travail se trouve au centre de l'écran: C'est l'onglet Form1.vb[Design] ci-dessous qui donne donc accès au dessin de la feuille (du formulaire); on peut ajouter des contrôles, modifier la taille de ces contrôles..

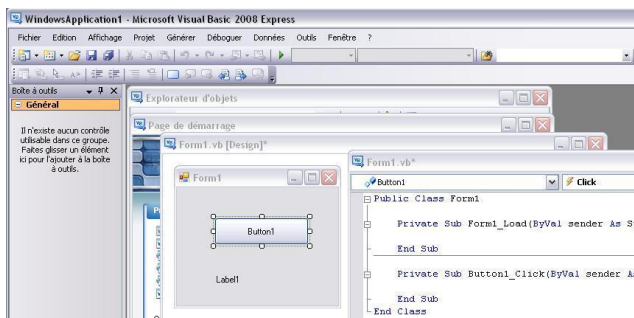


On peut passer en mode '**Multidocument Mdi**' (comme en VB6) au lieu du mode 'Onglet':

(Passer par le menu '**Outils**' puis '**Options..**' puis bouton '**Multidocument (Mdi)**').



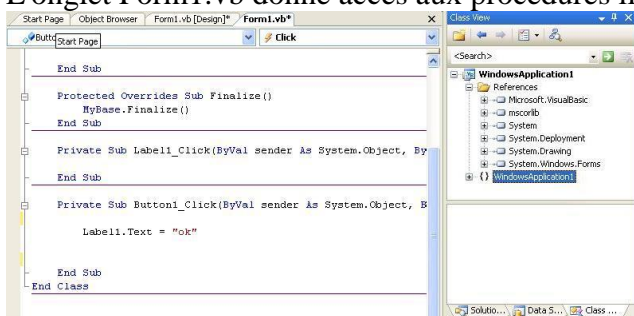
On obtient un mode multidocument avec plusieurs fenêtres.



A noter que si on utilise le menu 'Projet' puis 'Ajouter..' cela permet d'ajouter un formulaire, un module standard, un module de Classe.

d. Voir les procédures:

L'onglet Form1.vb donne accès aux procédures liées à Form1.



On peut saisir du code dans les procédures.

La liste déroulante de gauche donne la liste des objets, celle de droite, les évènements correspondants à cet objet.

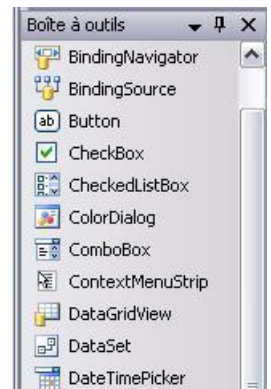
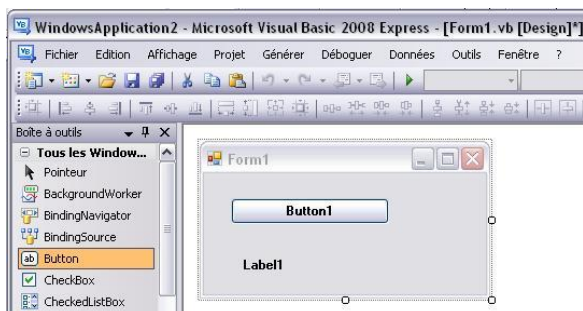
Il est possible en double-cliquant dans le formulaire ou un contrôle de se retrouver directement dans le code de la procédure correspondant à cet objet.

e. Ajouter des contrôles au formulaire 'Winform'

Ajouter un bouton par exemple:

Cliquer sur Boite à outils (Toolbox) à gauche, les contrôles apparaissent tous ou classés par ordre alphabétique.

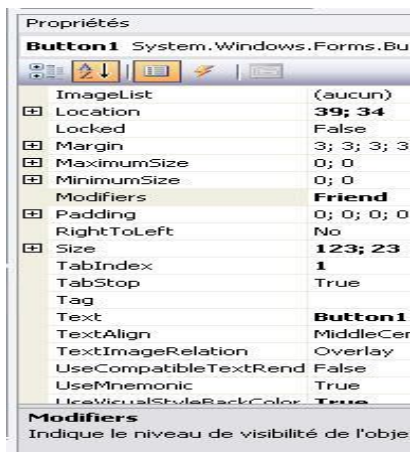
Cliquer sur 'Button' dans la boite à outils, cliquer dans la Form, déplacer le curseur sans lâcher le bouton, puis lâcher: Un bouton apparaît.



f. Modifier les propriétés d'un contrôle ou du formulaire.

Quand un formulaire ou un contrôle est sélectionné dans la fenêtre Design, ses propriétés sont

accessibles dans la fenêtre de 'Propriétés' (Properties) à droite en bas: Ici ce sont les propriétés du contrôle 'Button1' qui sont visibles (Text, Location..) on peut modifier directement les valeurs.



En bas de la fenêtre propriétés, il y a une explication succincte de la propriété sélectionnée (Si elle n'apparaît pas, click droit sur la propriété puis cocher la commande '**Description**').

Exemple:

Si au niveau de la ligne 'Text' des propriétés du bouton, j'efface 'Button1' et que je tape 'Ok', dans le designer (l'aperçu), le texte écrit sur le bouton deviendra 'OK'.

g. Le déplacement des contrôles et l'accès aux principales tâches:

La croix à gauche permet de déplacer le contrôle, la petite flèche à droite permet d'ouvrir un menu qui donne accès aux tâches les plus fréquentes.



h. L'alignement automatique des contrôles:

Si on modifie la taille ou l'emplacement d'un contrôle, VB signale par un trait bleu que le contrôle modifié et le contrôle voisin sont alignés:

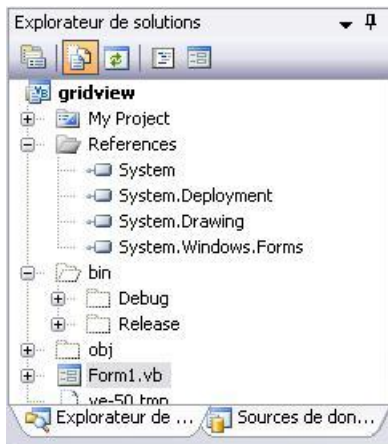


Renommer un nom: modification automatique.

On nomme cela '**Refactoring**': Cliquer sur une variable, puis bouton droit, dans le menu cliquer sur 'Renommer'. Modifier le nom de la variable, valider. Dans toute la Classe la variable est renommée.

i. Voir tous les composants d'un projet:

Pour cela il faut utiliser la fenêtre **Explorateur de solutions** en haut à droite, elle permet de voir et d'avoir accès au contenu du projet (Pour voir tous les fichiers, il faut cliquer sur le deuxième bouton en haut) : gridview est le nom du programme.




MyProject: double-cliquer dessus, vous ouvrirez la fenêtre 'propriétés du projet'.

Références qui contient les dll chargées. Pour atteindre les références, on peut aussi passer par le menu 'Projet' puis 'Propriétés' ou double cliquer sur 'MyProject' puis choisir l'onglet 'Références'.

Form1.vb est un formulaire (une fenêtre). Les formulaires, modules de classe ou standard sont tous des '.vb' Il suffit de double-cliquer dessus pour les ouvrir. Si on ouvre la sous-liste de Form1.vb (en cliquant sur le '+'), on voit: **Form1.Designer.vb** (qui montre le code qui crée le formulaire, on n'a pas à y toucher). **Form1.resx** (le fichier de ressources). Il suffit de cliquer sur la ligne Form1 dans l'explorateur de solution pour voir apparaître la Form1 dans la fenêtre principale.

Si on clique sur un espace de noms dans la liste Références, cela montre l'arborescence des Classes.

j. Tester son logiciel:

On peut tester le projet grâce à :  lancer l'exécution avec le premier bouton (mode 'Run', le second servant à arrêter temporairement l'exécution (mode 'Debug'), le troisième à terminer l'exécution (Retour au mode 'Design' ou 'Conception'). Quand on est en arrêt temporaire en mode 'Debug', la ligne courante, celle qui va être effectuée, est en jaune:

```
For i=0 To 100
  Label1.Text=i.ToString
Next i
```

```
For i=0 To 100
  Label1.Text=i.ToString
Next i
```

Si on tape la touche F10 (exécution pas à pas), la ligne 'Label1.Text=i.ToString' est traitée et la position courante passe à la ligne en dessous. En mode Debug, on peut modifier une ligne et poursuivre le programme qui tiendra compte de la modification (Sauf pour les déclarations). On parle d'Edit and continue'.

La **sauvegarde du projet** se fait comme dans tous les logiciels en cliquant sur l'icône du paquet de disquettes. On peut **compiler** le programme pour créer un exécutable par le menu Générer ('Build'). Le code présent dans l'IDE est le code **source**, après compilation le fichier exécutable contient du code **exécutable**.

k. Projet.

Dans la terminologie VB, un **projet** est une application en cours de développement.

Une '**solution**' (Team Project) regroupe un ou plusieurs projets (C'est un groupe de projets). Il n'y en a pas dans la version express.

En VB express on parle donc uniquement de projet, en fait ,VB crée aussi une solution de même nom.

l. Fichiers, Chemins des sources.

Si vous regardez dans ' C:\Documents and Settings\Nom Utilisateur\Mes documents\Visual Studio 2008\ Projects\MonProjet' les fichiers correspondant à un projet VB:

MonProjet.sln est le fichier solution.

MonProjet.psess est le fichier de performance (pas toujours présent).

MonProjet.suo est le fichier de User solution. Dessous existe un répertoire nommé aussi MonProjet qui contient:

MonProjet.vbProj le fichier de projet.

Form1.vb contient un formulaire et ses procédures.

MyClasse.vb contient par exemple des classes.

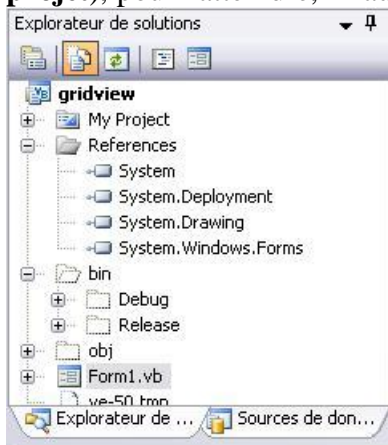
Form1.Designer.vb contient le code qui créer la fenêtre et les contrôles.

Il y a encore les sous répertoires \Bin, il y a aussi un répertoire Obj et un répertoire \MyProjet

Si on compile le projet l'exécutable est dans un sous répertoire \Bin,

m. Propriétés du projet:

Toutes les propriétés de l'application peuvent être modifiées dans le 'Projet Designer' (**Propriétés du projet**), pour l'atteindre, il faut double-cliquer sur 'My Project' dans l'explorateur de solution:



Une autre manière d'ouvrir le 'Projet Designer' est de passer par les menus 'Projet' puis 'Propriétés de..'. On retrouve dans le projet designer:



Le nom de l'application, son icône, la fenêtre de démarrage, celle de fin. (Application) Les Option Strict, Explicit compare et la nouvelle Option Infer.(Compiler).

Les références (dll liées au projet).

Les paramètres (valeurs liées à l'application).

Les ressources (texte, images, sons utilisées dans le programme). La signature et la sécurité.
 Les Extension My (nouveau 2008).
 Les paramètres relatifs à la publication (distribution et installation).

n. Erreur dans l'écriture du code.

S'il existe une erreur dans le code au cours de la conception, celle-ci est soulignée en bleu ondulé. Un carré donne la cause de l'erreur si le curseur passe sur la zone où se trouve l'erreur.

```
Label1.Texte() = "12"
```

'Texte' n'est pas un membre de 'System.Windows.Forms.Label'.

Ici la propriété 'Text' a été mal orthographiée.

Si je lance le programme en mode 'Run' et qu'il y a des erreurs, Vb me le signale et répertorie les erreurs dans la liste des tâches en bas. Vb propose des solutions pour corriger les erreurs de code. (Voir plus haut)

o. Mode débogage (mode BREAK):

Une fois lancée l'exécution (F5), puis stoppée (par Ctrl +Alt +Pause ou sur un point d'arrêt), on peut: Voir la valeur d'une propriété d'un objet en le pointant avec la souris:

```
Label1.Text = i.ToString
```

Label1.Text "0"

Il s'affiche un petit cadre donnant la valeur de la propriété d'un objet.

Voir la valeur d'une variable, simplement en positionnant le curseur sur cette variable.

F11 permet l'exécution pas à pas (y compris des procédures appelées). **F10** permet le pas à pas (sans détailler les procédures appelées). **Maj+F11** exécute jusqu'à la fin de la procédure en cours.


En cliquant sur le bouton droit de la souris, on peut exécuter jusqu'au curseur (Run To Cursor), voir la définition, la déclaration de ce qui est sous le curseur (Atteinte la définition:Go To Definition)...


II. Les formulaires et les contrôles


L'interface utilisateur est la partie de votre programme que les utilisateurs voient lorsqu'ils exécutent le programme. Une interface utilisateur se compose habituellement d'une fenêtre principale ou d'un formulaire et de plusieurs contrôles, tels que des boutons, des champs de saisie, etc. Ces types de programmes Visual Basic sont appelés applications *Windows Forms* et l'interface utilisateur est créée à l'aide de contrôles *Windows Forms*.

Les leçons de cette section vous montreront comment créer une interface utilisateur à l'aide de certains des contrôles *Windows Forms* les plus répandus.

A. Les contrôles Zone de texte (Textbox), Etiquette (Label) et Bouton de commande(Button)

Une **Zone de texte**  sert à afficher du texte ou à permettre à l'utilisateur de saisir du texte. L'utilisateur peut modifier le contenu de la zone de texte.

Un contrôle **Label**  permet d'afficher du texte que l'utilisateur ne peut pas modifier directement.

Un bouton de commande **Button**  permet de traiter l'événement utilisateur le plus courant. Comme dans la vie courante lorsque vous appuyez sur un interrupteur dans votre salon, la lampe s'allume. Vous avez provoqué un événement, vous attendiez une réponse à celui-ci, en l'occurrence que la lampe s'allume. Sous Visual Basic, il en va de même, l'utilisateur provoque un événement (lorsqu'il appuie sur le bouton) et attend en réponse l'exécution immédiate d'une action spécifique.

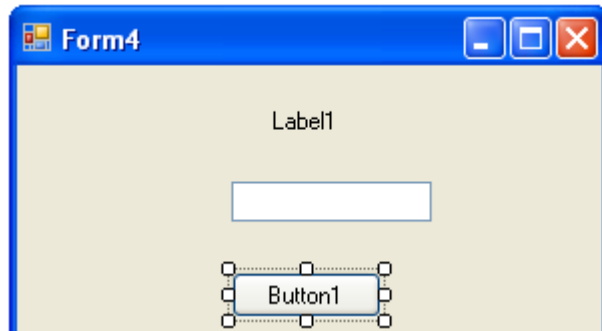
Exemple :

On désire créer un formulaire qui demande à l'utilisateur de saisir son nom. Lorsque l'utilisateur clique sur un bouton de commande, le nom saisi s'affiche dans une boîte de message.

Résolution

Démarrez Visual Basic et créer une nouvelle application "Application Windows Form".

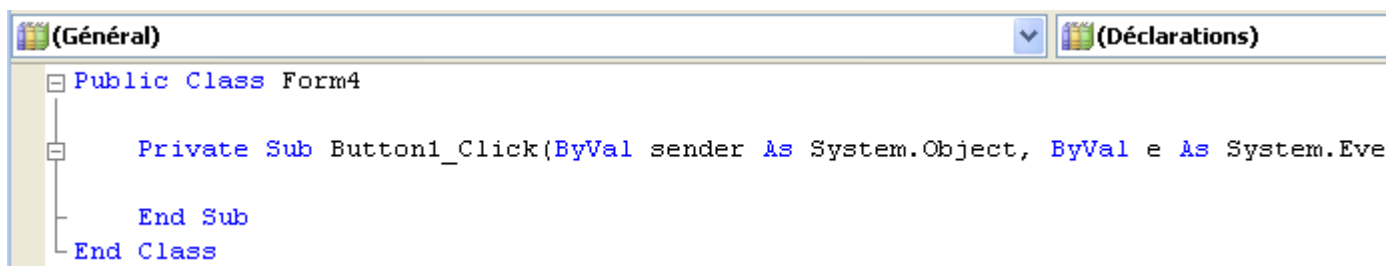
1. Positionnez sur la feuille(Form1) nouvellement créée un contrôle Label. Pour le faire, cliquer sur le contrôle Textbox dans la boîte à outils et cliquer sur le formulaire à l'endroit où vous désirez le placer
2. Faire la même chose pour le contrôle Textbox et le contrôle Button. On obtient la figure ci-dessous



3. Dans la fenêtre des propriétés, définir pour chacun des contrôles, les propriétés suivantes :

Contrôle	Propriété	Valeur
Label	Text	Veuillez saisir vos nom et prénoms
Button	Text	Afficher

Pour écriture le code devant générer l'événement souhaité, il faut effectuer un double-clic sur le bouton de commande. Une fenêtre s'ouvre, c'est l'éditeur de code:



Cette fenêtre contient dans son en-tête, deux zones de liste déroulante. La première contient la liste des contrôles présents sur le formulaire et la deuxième contient la liste des événements auxquels peut réagir le contrôle sélectionné.

Public Class Form4 permet de spécifier que toutes les procédures qui seront créées appartiennent à la classe Form4 (du nom du formulaire Form4).

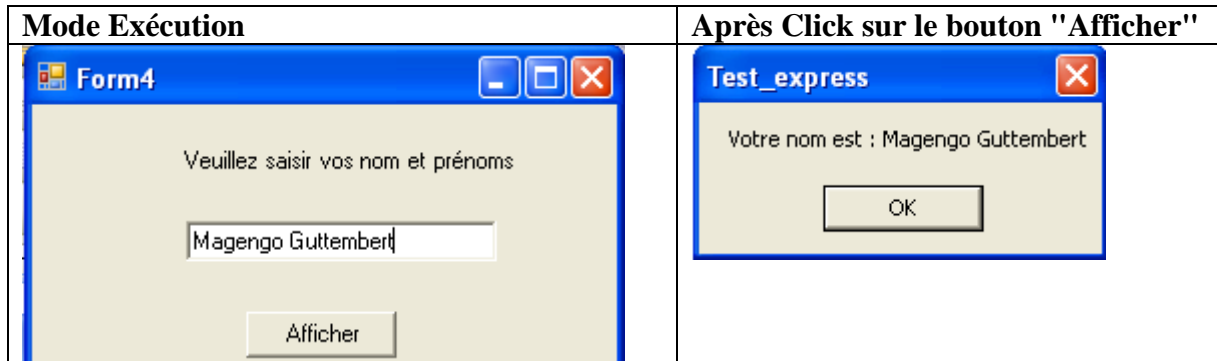
Private Sub Button1_Click est une procédure dite événementielle qui correspond à un programme qui va s'exécuter chaque fois que l'utilisateur va cliquer sur le bouton de commande. Le nom de la procédure est formé du nom du contrôle et du nom de l'événement les deux séparés par un trait de soulignement(_). Si vous n'écrivez rien dans cette procédure, il ne se passera rien.

4. Pour afficher le nom saisi dans une boîte de message, On écrira :

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

```
MsgBox("Votre nom est : " & TextBox1.Text)
```

```
End Sub
```



Exercice 2

On désire faire la même chose mais cette fois ci, le message doit s'afficher sur validation après que l'utilisateur ait saisi les Nom et prénom

Exercice 3

Créer un formulaire permettant de saisir dans deux zones de texte deux nombres et d'afficher dans une troisième zone de texte, la somme de ces deux nombres après avoir cliqué sur un bouton de commande

Remarque

Le mot clé Handles permet de partager la procédure événementielle entre plusieurs événements du même contrôle ou de contrôles différents. Pour partager un gestionnaire d'événements entre plusieurs contrôles, vous devez simplement ajouter les noms des contrôles supplémentaires et le nom de l'événement que vous souhaitez gérer. Le gestionnaire d'événements est alors notifié lorsque l'événement se produit pour un de ces contrôles. Par exemple, si vous disposez de deux contrôles Button, et si vous souhaitez utiliser le même gestionnaire d'événements pour les deux, la clause **Handles** ressemblera à ce qui suit :

Handles Button1.click, Button2. click.





Vous disposez à présent d'une seule méthode qui gère l'événement **Click** pour les deux contrôles. Mais comment fait le gestionnaire d'événements pour savoir quel contrôle a déclenché cet événement ? c'est grâce à la clause **ByVal sender As Object** : le mot clé **Sender** informe le gestionnaire d'événements de l'identité de l'objet (dans le cas présent, du contrôle) qui a déclenché l'événement.

Exemple :

On ajoute un second bouton sur la feuille et on modifie la procédure événementielle Button1_Click comme suit :

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click, Button2.Click
    If sender.Equals(Button1) Then
        MsgBox("Votre nom est : " & TextBox1.Text & " et vous avez appuyé sur le premier bouton")
    ElseIf sender.Equals(Button2) Then
        MsgBox("Votre nom est : " & TextBox1.Text & " et vous avez appuyé sur le deuxième bouton")
    End If
End sub
```



B. Les boutons d'option(Radiobutton), les cases à cocher(Checkbox), les zones de liste(Listbox) et les zones de liste déroulante (Combobox)

Les boutons d'option  (Radiobutton), les cases à cocher  (Checkbox), les zones de liste  (Listbox) et les zones de liste déroulante  (Combobox) sont des contrôles utilisés pour permettre à l'utilisateur d'effectuer des choix parmi plusieurs éléments.

a. Les boutons d'option(Radiobutton) et les cases à cocher(Checkbox),

Les boutons d'option(Radiobutton), les cases à cocher(Checkbox), fonctionnent de la même façon. Ils permettent de spécifier l'élément choisi en le cochant. La différence entre ces deux contrôles est que, dans un groupe de cases à cocher, on peut choisir plusieurs éléments alors que dans un groupe de boutons d'option, on ne peut choisir qu'un seul élément.

Remarque

Les boutons d'option doivent toujours être regroupées dans des contrôles conteneurs(Panel  ou Groupbox ) pour permettre à VB de différencier sur une même feuille, les différents Groupes d'options.

b. les zones de liste(Listbox) et les zones de liste déroulante (Combobox)

Un contrôle **ListBox** affiche une liste dans laquelle l'utilisateur peut sélectionner un ou plusieurs éléments. Si le nombre d'éléments est trop grand pour tenir dans la zone affichée, une barre de défilement lui est automatiquement ajoutée.

La propriété **Items** permet d'accéder à la collection des index des éléments d'un listbox. L'index du premier élément est 0.

La propriété **SelectedIndex** permet d'obtenir l'index de l'élément actuellement sélectionné

La propriété **SelectedItem** permet d'obtenir le texte de l'élément actuellement sélectionné

Remarques

Pour ajouter ou supprimer des éléments dans la zone de liste, utilisez la méthode **Add** de la collection **Items** de l'objet **Listbox** ou **Remove**. Vous pouvez également ajouter des éléments à la liste en utilisant la propriété **Items** au moment de la création.

Un contrôle **ComboBox** réunit les fonctionnalités des contrôles **TextBox** et **ListBox**. Il permet aux utilisateurs d'entrer des informations dans la partie zone de texte (TextBox) ou de sélectionner un élément dans la partie zone de liste (ListBox).

Pour ajouter des éléments dans un contrôle **ComboBox** ou pour en supprimer, utilisez les méthodes **Add** de la collection **Items** ou **Remove**. Vous pouvez également ajouter des éléments à la liste en utilisant la propriété **Items** au moment de la création.

Remarque



L'événement **DropDown** se produit lorsque la liste d'un contrôle **ComboBox** est sur le point de se dérouler. Cet événement ne se produit pas si la propriété **Style** d'un contrôle **ComboBox** a la valeur 1 (Simple Combo).

Utiliser une procédure d'événement **DropDown** pour procéder à la mise à jour finale d'une liste **ComboBox** avant que l'utilisateur effectue une sélection.

Exercice1

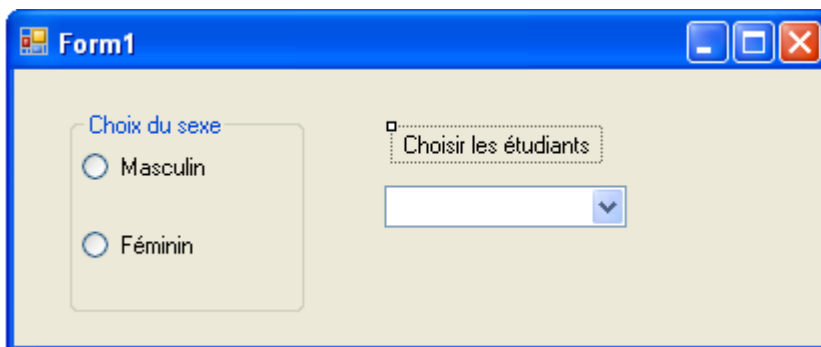
On désire créer un formulaire qui permet de remplir un combobox des noms des étudiants suivant le sexe choisi à partir d'un groupe de boutons d'option. Si l'utilisateur coche la case Masculin, sur déroulement du combobox, la liste des étudiants suivant s'affiche : Magengo, Guttembert. Si l'utilisateur coche la case Féminin, sur déroulement du combobox, la liste des étudiantes suivante s'affiche : Djangoni, Jolie, Joliette.

Résolution

1. Ajouter un formulaire à votre projet.
2. Sur ce formulaire, ajouter un contrôle Groupbox 
3. A l'intérieur du contrôle groupbox, dessiner deux contrôles bouton d'option  (Radiobutton)
4. A droite du contrôle groupbox ajouter un contrôle Combobox et au dessus du contrôle combobox, ajouter un contrôle Label.
5. Définir pour chacun des contrôles, les propriétés suivantes :

Contrôles	Propriétés	Valeur
Groupbox	Text	Choix du sexe
RadioButton1	Name	Masculin
RadioButton1	Text	Masculin
RadioButton2	Name	Féminin
RadioButton2	Text	Féminin
Label1	Text	Choisir les étudiants

On obtient en mode création, le formulaire suivant :



6. Double cliquer sur le contrôle combobox
7. Choisir l'évènement DropDown et taper le code suivant :

```
Private Sub ComboBox1_DropDown(ByVal sender As Object, ByVal e As System.EventArgs) Handles
ComboBox1.DropDown
```

```
    If Masculin.Checked = True Then
        ComboBox1.Items.Clear()
        ComboBox1.Items.Add("Magengo")
        ComboBox1.Items.Add("Guttembert")
        ComboBox1.Items.Add("Gros")
        ComboBox1.Items.Add("Séchard")
    End If
```

```
    If Féminin.Checked = True Then
        ComboBox1.Items.Clear()
        ComboBox1.Items.Add("Djangoni")
        ComboBox1.Items.Add("Mandelai")
        ComboBox1.Items.Add("Joliette")
    End If
End Sub
```

Exercice 2

On désire créer un formulaire qui permet de remplir un Listbox des noms des loisirs suivants les catégories de loisir sélectionnés à partir d'un groupe de cases à cocher. Les catégories des loisirs sont les suivants

Sport (Boxe, Lutte, Football)

Lecture(Poésie, Policier)

Cinéma(Horreur, Aventure, Action, amour)

TAF

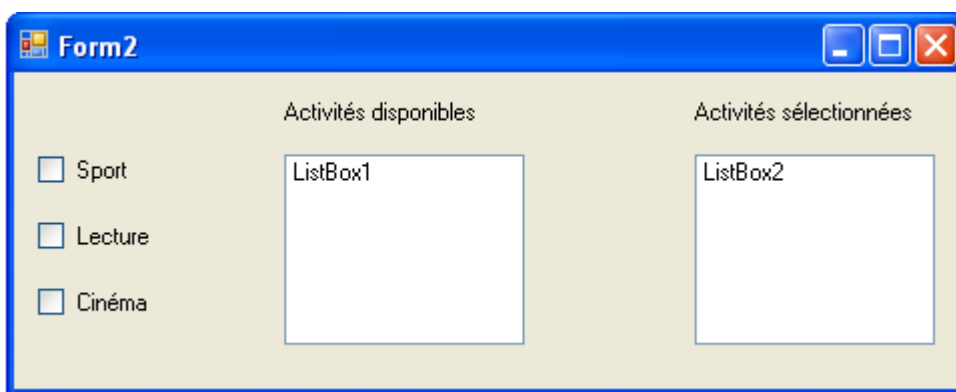
Chaque fois que l'utilisateur coche une catégorie, la liste des éléments de cette catégorie s'ajoute aux éléments déjà présents dans la zone de liste (listbox1). Chaque fois qu'une catégorie est décochée, la liste des éléments de cette catégorie est soustraite des éléments présents dans la zone de liste (listbox1). Un double clic sur un élément de la zone de liste listbox1 ajoute cet élément à la zone de liste listbox2. Un double clic sur un élément de la zone de liste listbox2 supprime cet élément à la zone de liste listbox2.

Résolution

1. Ajouter un formulaire à votre projet.
2. Sur ce formulaire, dessiner trois cases à cocher ☒ (Checkbox)
3. A droite du groupe de cases à cocher, ajouter deux contrôles Listbox et au dessus de chaque contrôle Listbox, ajouter un contrôle Label.
4. Définir pour chacun des contrôles, les propriétés suivantes :

Contrôles	Propriétés	Valeur
CheckBox1	Name	Sport
CheckBox1	Text	Sport
CheckBox2	Name	Lecture
CheckBox2	Text	Lecture
CheckBox3	Name	Cinéma
CheckBox3	Text	Cinéma
Label1	Text	Activités disponibles
Label2	Text	Activités sélectionnées

On obtient en mode création, le formulaire suivant :



5. Double cliquer sur le contrôle CheckBox1 (Sport)
6. Choisir l'évènement CheckedChanged et taper le code suivant :

```
Private Sub Sport_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Sport.CheckedChanged
    'On efface le contenu de listbox1
```



```

ListBox1.Items.Clear()
'Si la case à cocher Sport est cochée
If Sport.Checked = True Then
    'on ajoute à listBox1 les activités sportives
    ListBox1.Items.Add("Boxe")
    ListBox1.Items.Add("Lutte")
    ListBox1.Items.Add("Football")
End If
If Cinéma.Checked = True Then
    ListBox1.Items.Add("Horreur")
    ListBox1.Items.Add("Aventure")
    ListBox1.Items.Add("Action")
    ListBox1.Items.Add("Amour")
End If
If Lecture.Checked = True Then
    ListBox1.Items.Add("Poésie")
    ListBox1.Items.Add("Policier")
End If
End Sub

```

Cette procédure ne s'exécute que lorsqu'on coche ou qu'on décoche la case à cocher Sport. Elle devra aussi s'exécuter lorsqu'on coche ou qu'on décoche les cases à cocher Lecture et Cinéma. Pour cela, on a trois possibilités :

1^{ère} possibilité :

On crée les procédures événementielles **Lecture_CheckedChanged** et **Cinéma_CheckedChanged** dans lesquels on recopie le même code que celui de **Sport_CheckedChanged**

2^{ème} possibilité :

On crée une procédure (non événementielle) qu'on va nommer **Ajouter()** et qui contiendra le code le même code que celui de **Sport_CheckedChanged**. On crée ensuite les procédures événementielles **Lecture_CheckedChanged**, **Cinéma_CheckedChanged** et **Sport_CheckedChanged** (on efface le code qui y était au préalable) dans lesquelles on appelle la procédure **Ajouter()**

```

Sub Ajouter()
    'On efface le contenu de listBox1
    ListBox1.Items.Clear()
    'Si la case à cocher Sport est cochée
    If Sport.Checked = True Then
        'on ajoute à listBox1 les activités sportives
        ListBox1.Items.Add("Boxe")
        ListBox1.Items.Add("Lutte")
        ListBox1.Items.Add("Football")
    End If
    If Cinéma.Checked = True Then
        ListBox1.Items.Add("Horreur")
        ListBox1.Items.Add("Aventure")
        ListBox1.Items.Add("Action")
        ListBox1.Items.Add("Amour")
    End If
    If Lecture.Checked = True Then
        ListBox1.Items.Add("Poésie")
        ListBox1.Items.Add("Policier")
    End If
End Sub

```

End If
End Sub

Private Sub Sport_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles Sport.CheckedChanged
Ajouter()
End Sub

Private Sub Lecture_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles Lecture.CheckedChanged
Ajouter()
End Sub

Private Sub Cinéma_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Cinéma.Click
Ajouter()
End Sub

Remarque

```
If ListBox1.SelectedItem IsNot Nothing Then
    If Not ComboBox1.Items.Contains(Me.ListBox1.SelectedItem) Then
        Me.ComboBox1.Items.Add(Me.ListBox1.SelectedItem)
    End If


    Me.ComboBox1.SelectedItem = Me.ListBox1.SelectedItem
End If
```

Chapitre 3 : Les menus

I. Ajouter un menu

Dans cette leçon, vous apprendrez comment créer des menus et comment écrire un code qui s'exécute lorsque des éléments de menu sont sélectionnés. Vous apprendrez également à ajouter un jeu d'éléments de menu standard en une seule étape.

Les menus constituent pour l'utilisateur un moyen facile et familier de faire des choix quant à votre programme. Les menus sont généralement utilisés pour présenter les options d'un programme, pour ajouter des raccourcis pour les tâches les plus répandues, notamment couper et coller, ou pour charger et enregistrer des fichiers.

Visual Basic facilite l'implémentation des menus. Le contrôle **MenuStrip**  vous permet de créer des menus graphiquement. Une fois placé dans un formulaire, le contrôle **MenuStrip** se présente sous la forme d'une zone située dans la partie supérieure du formulaire et qui contient la mention « Tapez ici ». Vous pouvez cliquer dans la zone et y entrer un texte pour créer le titre du menu correspondant.

Lorsque le titre d'un élément de menu est défini, les éléments de menu supplémentaires peuvent être créés en dessous à droite de celui-ci. Cela vous permet d'étendre le menu avec autant d'éléments ou de sous-éléments supplémentaires que vous le souhaitez. Lorsque l'aspect de votre menu vous convient, créez les gestionnaires d'événements qui régiront les événements Click associés à chaque élément.

Exemple

1. Dans le menu **Fichier**, cliquez sur **Nouveau Projet**.
2. Dans la boîte de dialogue **Nouveau projet**, dans le volet **Modèles**, cliquez sur **Application Windows**.
3. Dans la zone **Nom**, tapez **Menus**, puis cliquez sur **OK**. Un nouveau projet Windows Forms s'ouvre.
4. À partir de la **Boîte à outils**, faites glisser un contrôle **MenuStrip** dans le formulaire. Quel que soit l'emplacement où vous le placez, le contrôle **MenuStrip** se fixe automatiquement au sommet du formulaire. Vous avez pu également remarquer qu'une icône **MenuStrip1** est ajoutée dans une zone grise située sous le formulaire. Cette zone est appelée barre d'état des composants. Si vous cliquez hors du contrôle **MenuStrip**, la zone disparaît. Vous pouvez la faire réapparaître en cliquant sur l'icône **MenuStrip1**.
5. Dans le formulaire, cliquez sur le contrôle **MenuStrip**, tapez **File**, puis appuyez sur **Entrée**. De nouvelles zones destinées aux entrées de menu supplémentaires s'affichent sous le premier élément de menu, à sa droite. Il s'agit d'espaces destinés à des éléments de menu supplémentaires. Vous pouvez continuer d'ajouter des éléments de menu dans les deux sens, jusqu'à ce que votre menu soit complet.
6. Dans la zone située sous la première zone, tapez **Exit**, puis appuyez sur **Entrée**.
7. Double-cliquez sur le menu **Quitter** pour ouvrir l'éditeur de code.
8. Dans le gestionnaire d'événements **ExitToolStripMenuItem_Click**, tapez le code suivant.

```
Application.Exit()
```

9. Appuyez sur F5 pour exécuter le programme. À l'aide de la souris, cliquez sur le menu **Fichier**, puis sur **Quitter**. Votre application se ferme.

II. Ajouter des éléments de menu standard

Vous pouvez ajouter plusieurs menus standard et éléments de menu en une seule étape. Le contrôle MenuStrip possède un volet **Tâches MenuStrip** qui permet d'insérer plusieurs éléments de menu standard dans le contrôle MenuStrip.

Pour ajouter un jeu d'éléments de menu standard à une bande de menu

1. Sélectionnez le contrôle **MenuStrip**, cliquez sur la flèche de tâche guidée dans l'angle supérieur droit du contrôle puis sur **Insérer des éléments standard**. Plusieurs menus standard et éléments de menu sont ajoutés au contrôle MenuStrip.
2. Appuyez sur F5 pour exécuter le programme.
3. Au démarrage de l'application, passez en revue les éléments de menu sur les nouveaux menus pour vous familiariser avec les éléments standard.
4. Fermez l'application.

Remarque :

Vous pouvez supprimer des éléments de menu qui ne s'appliquent pas à votre application. Vous devrez écrire du code dans le gestionnaire d'événements Click pour chaque élément de menu que vous utilisez.

III. Activer ou désactiver les commandes de menu

La plupart des programmes désactivent les commandes de menu lorsqu'elles sont indisponibles, au lieu de les masquer. Lorsqu'un élément de menu est désactivé, le texte de menu devient gris et cliquer sur l'élément de menu ne produit aucun effet. Lors de l'utilisation d'un contrôle MenuStrip, vous pouvez activer et désactiver des éléments de menu à l'aide de la propriété Enabled de l'élément MenuItem.

Exemple

1. Dans le menu **Fichier**, cliquez sur **Nouveau Projet**.
2. Dans la boîte de dialogue **Nouveau projet**, dans le volet **Modèles**, cliquez sur **Application Windows Forms**.
3. Dans la zone **Nom**, tapez Menus2, puis cliquez sur **OK**. Un nouveau projet Windows Forms s'ouvre.
4. À partir de la **Boîte à outils**, faites glisser un contrôle MenuStrip et un contrôle TextBox dans le formulaire.
5. Dans le formulaire, cliquez sur le contrôle MenuStrip et tapez Edit, puis appuyez sur ENTRÉE.
6. Dans la zone située sous la première zone, tapez Copy,, puis appuyez sur Entrée.
7. Dans la fenêtre **Propriétés**, attribuez à la propriété **Enabled** de **CopyToolStripMenuItem** la valeur **False**.
8. Double-cliquez sur le contrôle TextBox pour ouvrir l'éditeur de code.
9. Dans le gestionnaire d'événements **TextBox1_TextChanged**, tapez le code suivant.

```
If Textbox1.Text <> "" Then
    CopyToolStripMenuItem.Enabled = True
Else
    CopyToolStripMenuItem.Enabled = False
End If
```

- Appuyez sur F5 pour exécuter le programme. Cliquez sur le menu **Edition**. L'élément de menu **Copier** est désactivé. Tapez quelque chose dans le contrôle TextBox, puis cliquez à nouveau sur le menu **Edition**. L'élément de menu **Copier** est à présent activé.

IV. Création de menus contextuels

Beaucoup de programmes utilisent des *menus contextuels* pour faciliter l'accès aux commandes les plus utilisées. Pour accéder à un menu contextuel, cliquez avec le bouton droit sur un formulaire ou un contrôle au moment de l'exécution. Vous pouvez créer vos propres menus contextuels dans Visual Basic à l'aide d'un contrôle ContextMenuStrip.

Comme avec le contrôle MenuStrip, lorsque vous faites glisser un contrôle ContextMenuStrip dans un formulaire, le contrôle ContextMenuStrip s'affiche dans la partie supérieure du formulaire sous la forme d'une zone qui présente le texte « Tapez ici », et une icône est ajoutée à la barre d'état des composants. Contrairement à MenuStrip, les éléments supplémentaires s'ajoutent uniquement sous le premier élément de menu, créant ainsi un menu vertical.

De plus, un ContextMenuStrip doit être associé au formulaire ou au contrôle dans lequel vous souhaitez qu'il s'affiche. Pour ce faire, attribuez pour valeur à la propriété ContextMenuStrip du formulaire ou du contrôle le nom du contrôle ContextMenuStrip. Vous pouvez associer un seul ContextMenuStrip avec autant de contrôles que vous le souhaitez.

Exemple

- Dans le menu **Fichier**, sélectionnez **Nouveau Projet**.
- Dans la boîte de dialogue **Nouveau projet**, dans le volet **Modèles**, cliquez sur **Application Windows**.
- Dans la zone **Nom**, tapez ContextMenus, puis cliquez sur **OK**.
Un nouveau projet Windows Forms s'ouvre.
- À partir de la **Boîte à outils**, faites glisser un contrôle ContextMenuStrip dans le formulaire.
- Dans la fenêtre **Propriétés**, sélectionnez la propriété **ContextMenuStrip** du formulaire, puis sélectionnez **ContextMenuStrip1** dans la liste déroulante.
- Dans le formulaire, cliquez sur le contrôle ContextMenuStrip et tapez Option1, puis appuyez sur ENTRÉE.
- Dans la zone située sous la première zone, tapez Option2,, puis appuyez sur Entrée.
- Double-cliquez sur l'élément de menu **Option1** pour ouvrir l'éditeur de code.
- Dans le gestionnaire d'événements **Option1ToolStripMenuItem_Click**, tapez le code suivant.

```
MsgBox("You chose Option 1")
```

- Dans l'**éditeur de code**, sélectionnez **Option2ToolStripMenuItem** dans la liste déroulante de gauche, puis **Click** dans celle de droite. Un nouveau gestionnaire d'événements appelé **Option2ToolStripMenuItem_Click** s'affiche dans l'éditeur de code.
- Dans le gestionnaire d'événements **Option2ToolStripMenuItem_Click**, tapez le code suivant.

```
MsgBox("You chose Option 2")
```

- Appuyez sur F5 pour exécuter le programme. Cliquez avec le bouton droit sur le formulaire, puis sur l'un des éléments du menu contextuel. Un message s'affiche et indique l'option choisie.

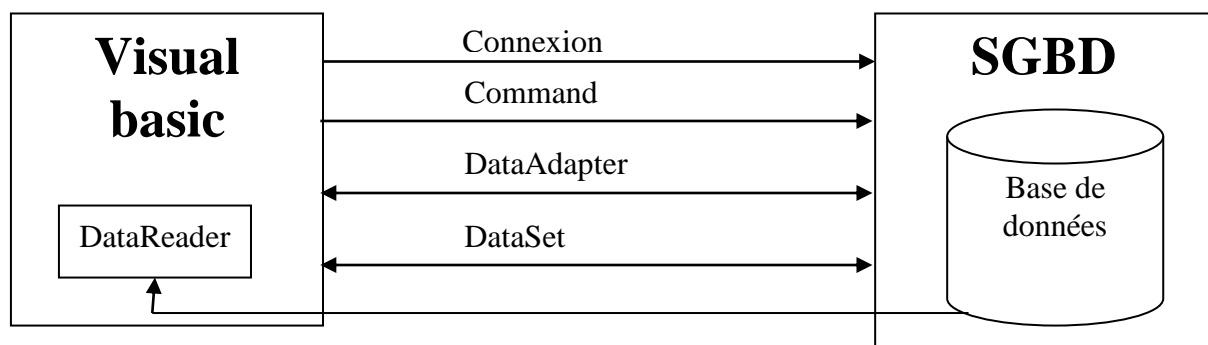
Chapitre 4 : VB.net et les bases de données

I. Rappel sur les bases de données (Voir le cours de base de données relationnelles)

II. Généralités sur ADO.NET

Pour avoir accès à partir de VB.NET aux bases de données il faut utiliser ADO.NET. ADO veut dire **Activex Database Objet**. C'est la couche d'accès aux bases de données, le SGBD (Système de Gestion de Base de Données) de VB. ADO.NET est ".NET" donc **managé** et géré par le CLR.

Il est indépendant de la base de donnée alors que initialement chaque type de gestionnaire de base de données avait ses instructions, sa manière de fonctionner, ADO.NET à un langage unique pour ouvrir, interroger, modifier une base de données quelle que soit la base de données.



Le langage de requête est le SQL.

III. Les Objets ADO.NET

a. Les Managed Providers

Pour avoir accès aux données il faut charger les **DRIVERS** (ou providers). Comme d'habitude, il faut:

- Charger les références des drivers (les Dll)
- Importer les espaces de nom. Ainsi on a accès aux objets.

Exemples :

- **OLE DB** Managed Provider est fourni dans 'System'; après avoir importé le NameSpace System.Data.OLEDB, on peut travailler sur des bases Access par exemple.
- **SQL Server** Managed Provider est fourni dans 'System'; après avoir importé le NameSpace System.Data.SqlClient, on peut travailler sur des base SqlServeur.

Ainsi, pour travailler sur une base Access, il faudra taper: `Imports System.Data.OLEDB`

b. L'objet Connection








Il permet de définir une connexion. Pour définir une connexion, Il faut donner les paramètres de cette connexion. Ces paramètres sont différents d'un SGBD à un autre. Ils peuvent être définis grace à la propriété `ConnectionString` de l'objet `Connexion`. Par exemple, pour une base Access, il faut définir :

- Le Provider qui indique le moteur de BD. Provider = OLEDB Jet 4.0
- Le Data source qui indique le chemin et le nom de la BD.







Et dans certains cas, Il peut être nécessaire de donner aussi :

- Le Password qui indique le mot de passe de la base si celle-ci était protégée
- Le User ID qui indique le nom d'utilisateur.




Méthodes, propriétés et évènements de l'objet Connection**Méthodes**

	Nom	Description
	ChangeDatabase	Modifie la base de données en cours d'un OleDbConnection ouvert.
	Close	Ferme la connexion à la source de données.
	CreateCommand	Crée et retourne un objet OleDbCommand associé à OleDbConnection.
	Dispose	Surchargé. Libère toutes les ressources utilisées
	Equals	Détermine si l'objet Object spécifié est égal à l'objet Object en cours.
	Open	Ouvre une connexion de base de données avec les paramètres de propriété spécifiés par ConnectionString.
	ResetState	Met à jour la propriété State de l'objet OleDbConnection .

Propriétés

	Nom	Description
	ConnectionString	Obtient ou définit la chaîne utilisée pour ouvrir une base de données.
	ConnectionTimeout	Obtient la durée d'attente préalable à l'établissement d'une connexion avant que la tentative ne soit abandonnée et qu'une erreur ne soit générée.
	Database	Obtient le nom de la base de données en cours ou de la base de données à utiliser une fois la connexion ouverte.
	DataSource	Obtient le nom de serveur ou le nom de fichier de la source de données.
	Provider	Obtient le nom du fournisseur OLE DB tel qu'il est spécifié dans la clause "Provider=" de la chaîne de connexion.
	State	Obtient l'état actuel de la connexion. Les changements d'état autorisés sont les suivants : <ul style="list-style-type: none"> • De Closed à Open, à l'aide de la méthode Open de l'objet Connection. • De Open à Closed, à l'aide de la méthode Close ou de la méthode Dispose de l'objet

Événements






	Nom	Description
	Disposed	Se produit lorsque le composant est supprimé par un appel à la méthode Dispose.
	InfoMessage	Se produit lorsque le fournisseur envoie un avertissement ou un message d'information.
	StateChange	Se produit lorsque l'état de l'événement change.

c. L'objet Command



Pour chaque provider il y a un objet Command spécifique: **SqlCommand**, **OleDbCommand**, mais tous les objets 'Command' ont la même interface, les mêmes membres.



Méthodes, propriétés et événements de l'objet Command

Méthodes

	Nom	Description
	Cancel	Tente d'annuler l'exécution de la commande
	ExecuteNonQuery	Exécute la requête SQL de CommandText (généralement une mise à jour par UPDATE, INSERT, DELETE ou un ajout de table) sans retourner de données.
	ExecuteReader	Surchargé. Envoie le texte de la commande SQL à exécuter (CommandText) à l'objet Connection et génère un objet DataReader .
	ExecuteScalar	Exécute la requête et retourne la première colonne de la première ligne du jeu de résultats retourné par la requête. Les colonnes ou lignes supplémentaires sont ignorées.
	ResetCommandTimeout	Rétablit la valeur par défaut de la propriété CommandTimeout .

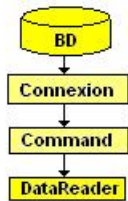
Propriétés

	Nom	Description
	CommandText	Obtient ou définit l'instruction SQL ou la procédure stockée à exécuter au niveau de la source de données.
	CommandTimeout	<ul style="list-style-type: none"> Obtient ou indique la durée en secondes avant qu'une requête qui plante soit abandonnée.

	CommandType	Obtient ou définit une valeur qui indique la manière dont la propriété CommandText doit être interprétée. <ul style="list-style-type: none"> ○ Text ; par défaut, exécution direct des instructions SQL contenues dans CommandText ○ StoredProcedure: exécution de procédure stockée dont le nom est dans CommandText. ○ Tabledirect : spécifie que les données proviennent d'une table.
	Connection	Obtient ou définit l'objet Connection utilisé par cette instance de l'objet Command .

d. L'objet DataReader

Un objet **DataReader** fournit des données en lecture seule en un temps record. La seule possibilité est de se déplacer en avant. En contrepartie de sa rapidité il monopolise la connexion. Il faut créer un objet **Connexion** puis un objet **Command**, ensuite on exécute la propriété **ExecuteReader** pour créer l'objet **DataReader**; enfin on parcourt les enregistrements avec la méthode **Read**.



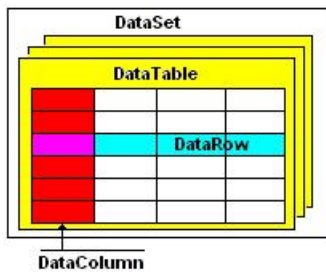
Pour chaque provider il y a un objet 'DataReader' spécifique : **SqlDataReader**, **OleDbDataReader**

- **Read** avance la lecture des données à l'enregistrement suivant, etl retourne la valeur **True** s'il y a encore des enregistrements et **False** quand il est en fin de fichier.
- **GetDateTime**, **GetDouble**, **GetGuid**, **GetInt32**, **GetBoolean**, **GetChar**, **GetFloat**, **GetByte**, **GetDecimal** permettent de récupérer les valeurs typées des champs.
- **GetName** retourne le nom du champ (numéro du champ en paramètre)
- **GetOrdinal** fait l'inverse: retourne le numéro du champ (nom en paramètre)
- **FieldCount** retourne le nombre de colonne.
- **GetDataTypeName** retourne le nom du type de donnés.
- **IsDBNull** retourne True si le champ est vide

e. L'objet DataSet

Le **DataSet** a la structure d'une base de données mais en local; il contient des **DataTable** qui contiennent des **DataRow** et des **DataColumn**. Pour utiliser **DataSet**, **DataTable**, **DataRow**, il faut importer l'espace de nom Data: **Imports System.Data**. On peut créer un **Dataset** de toute pièce en écrivant du code, mais la plupart du temps, on charge le **DataSet** à partir d'une base de données.

Une requête SQL charge le **DataSet**, on travaille sur les lignes et colonnes du **DataSet** en local (sur des enregistrements ou des champs), en mode déconnecté (une fois que le DataSet est chargé, la connexion à la base de données peut être libérée).



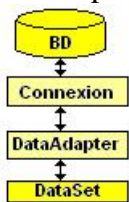
La structure de données du **DataSet** reflétera automatiquement et exactement celle des données extraites. Si j'extrais 2 colonnes de données avec l'instruction SQL fournis à l'objet Command, le **DataSet** aura une table de 2 colonnes avec les données extraites.

Exemple :

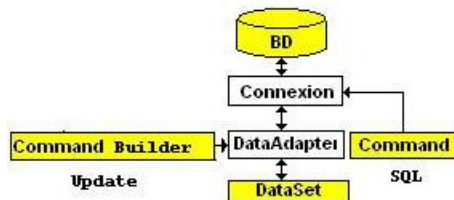
Avec ADO.NET je lance une requête SQL demandant tous les enregistrements de la table Etudiant dont le champ 'prénom' est 'Magengo', je récupère un DataSet local contenant tous ces enregistrements (Le DataColumn "Prénom" ne contient que des 'Magengo'). Je peux modifier en local le DataSet, (modifier le sexe ou la date de naissance ou la filière par exemple) et mettre à jour automatiquement la base de données distante.

Le **DataSet** est une représentation en mémoire des données. On charge le DataSet à partir de la base de données. Une fois chargé on peut travailler en mode déconnecté. Pour effectuer une modification, on modifie le DataSet puis on met à jour la base de données à partir du DataSet.

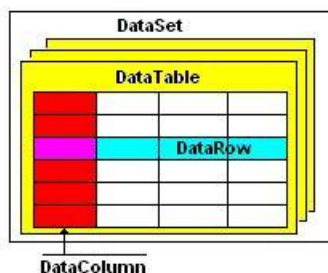
Pour remplir un DataSet il faut un objet **Connexion** puis un objet **DataAdapter**.



Il faut créer un objet **Connexion** puis un objet **DataAdapter** qui par sa méthode **Fill** charge le **DataSet**.

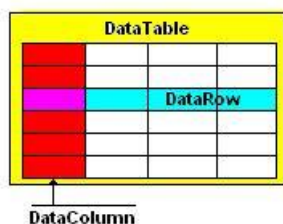


Les données sont extraites à l'aide de requête SQL sur l'objet **Command**, et on utilise un **CommandBuilder** pour mettre à jour la base de données à partir du DataSet.

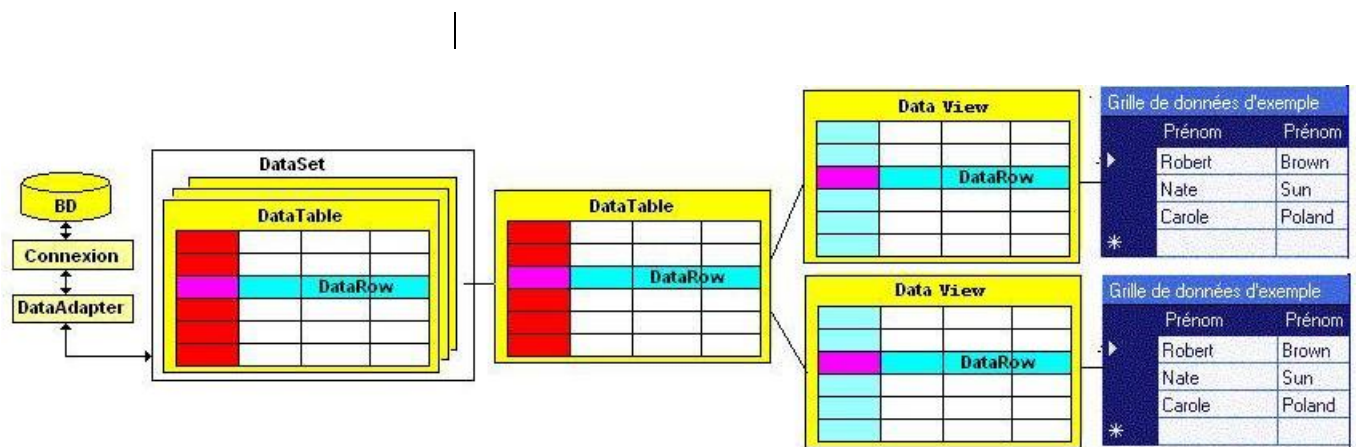


Le **DataSet** est organisé comme une base de données en mémoire, il possède:

- Une propriété **Tables** qui contient des **DataTable** (Comme les tables d'une BD). Chaque **DataTable** contient une propriété **Columns** qui contient les **DataColumn** (les colonnes ou champs des BD) et une propriété **Rows** qui contient des **DataRow** (Les lignes ou enregistrements des BD)
- **DataColumn** contient des informations sur le type du champ.
- **DataRow** permet d'accéder aux données.



On peut créer des **DataTable** 'autonomes' et y mettre une table provenant d'un **DataSet**. On peut créer des **DataView** qui sont des représentations d'une table. A partir d'une table, on peut créer plusieurs **DataView** avec des représentations différentes: **DataView** trié, **DataView** dont les lignes ont été filtrées (**RowFilter**) Enfin une **DataTable** ou un **DataView** peut être affichés dans un contrôle (**Grid** par exemple).



Un **DataSet** possède aussi la propriété **Constraints** qui contient les **Constraint** (Clé primaire ou clé étrangère), et la propriété **Relations** qui contient les **DataRelations** (Relation entre les tables).

Ce qu'il faut retenir

Il y a 2 manières de travailler avec une base de données

1^{ère} méthode : On envoie une requête Sql 'SELECT' à la base, on récupère le résultat dans un objet.

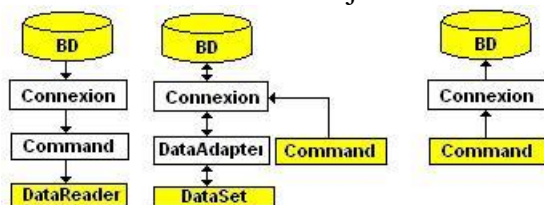
- Avec un objet **DataReader** on extrait les données en lecture seule: une requête SQL charge le **DataReader**. C'est rapide; on peut lire uniquement les données et aller à l'enregistrement suivant. Il travaille en mode connecté. Pour gérer un **DataReader** on a besoin d'un objet **Command**.
- Avec un objet **DataSet**, on manipule les données extraites: une requête SQL charge le DataSet avec des enregistrements et des champs, on travaille sur les lignes et colonnes du DataSet en local, en mode déconnecté (une fois que le **DataSet** est chargé, la connexion à la base de données est libérée). Pour alimenter un DataSet on a besoin d'un objet **DataAdapter** qui fait l'intermédiaire entre la BD et le **DataSet**.

2^{ème} méthode : On manipule directement la base. (Sans retour de résultats)

- Avec un objet **Command** on peut manipuler directement la BD (UPDATE, INSERT, DELETE CREATE, DROP...), on utilise la propriété **ExecuteNonQuery** pour cela.

Avec la méthode **Close** on ferme la base.

Résumons les différents objets nécessaires pour travailler sur une BD:



Noter bien le sens des flèches:

- le **DataReader** est en lecture seule, les données lues dans la BD sont accessibles dans le **DataReader**.
- le **DataSet** peut lire et écrire des données dans la BD, il faut un **DataAdapter** en plus de la connexion.
- l'objet **Command** peut modifier la BD.

Ce schéma souligne aussi les objets intermédiaires nécessaires:

- un objet **connexion** dans tous les cas,
- un objet **Command** pour le **DataReader**,
- un objet **DataAdapter** plus un objet **Command** pour le **DataSet**.

L'objet **Command** permet d'envoyer des ordres en SQL à la BD et de la modifier, il permet aussi, quand on utilise un **DataSet**, d'envoyer une requête SELECT en SQL afin de remplir le **DataSet** avec le résultat de la requête. Enfin certains contrôles comme les **DataGrid**, les **ListBox** par exemple peuvent afficher des données à partir d'un **DataSet**.

Pour mettre à jour la base après modification du **DataSet** ou de la **Grid** il faut un objet **CommandBuilder**. Mode connecté ou déconnecté:

- le **DataReader** fonctionne en mode connecté, la connexion entre la BD et le DataReader est ouverte tant que le DataReader fonctionne.
- le **DataSet** peut travailler en mode déconnecté: on ouvre la connexion, on charge le DataSet, on ferme la connexion (il faut le faire, ce n'est pas automatique), on travaille sur le DataSet, on peut le rouvrir plus tard pour les mises à jour.

Remarque: En fonction du provider, le nom des objets change: Avec le provider OleDb, après Imports System.Data.OleDb on utilisera OleDbConnexion, OleDbAdapter... Avec le provider SQL, après Imports System.Data.SqlClient on utilisera SqlConnection, SqlDataAdapter...

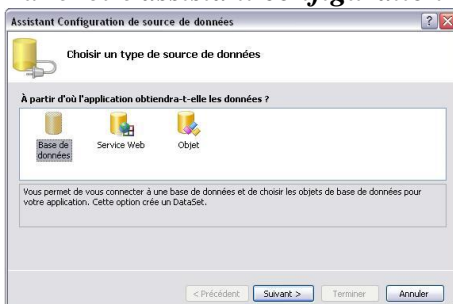
IV. Se connecter à une base de données sans écrire de code VB

1. Création de la source de données

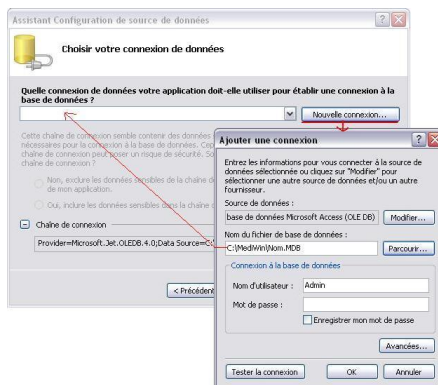
Dans la barre de Menu, cliquer sur le menu **Données** et choisir **Ajouter une nouvelle source de données**



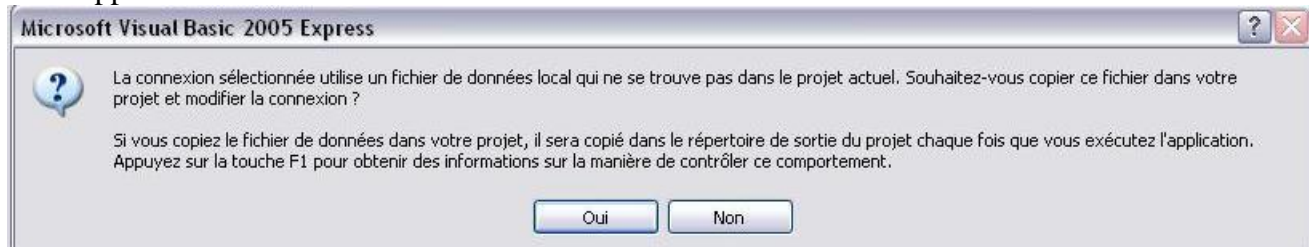
La fenêtre *assistant configuration de source de données* s'ouvre.



Sélectionner *Base de données* et cliquer sur le bouton *Suivant*. La fenêtre *Choisir votre connexion de données* s'ouvre.



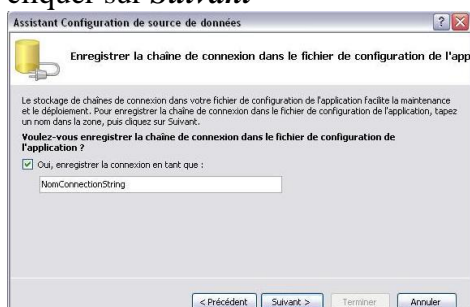
Cliquer sur **Nouvelle connexion**. Une nouvelle fenêtre nommée **Ajouter une connexion** s'ouvre : Indiquer le type de source de données en cliquant sur le bouton **Modifier** (ici Microsoft Access (OLEDB)), puis cliquer sur le bouton **Parcourir** pour rechercher et sélectionner la base de données (ici Gesco.mdb) et cliquer sur **Ok**. On retrouve le nom de la base de données dans la zone connexion. Le bouton **Tester la connexion** permet de vérifier que la connexion est effective. Cliquer sur **Suivant**. Une boîte de message apparaît et vous demande s'il faut copier la base de données dans le projet : Si oui la base de données sera copiée dans le répertoire de travail quand vous installerez votre application.



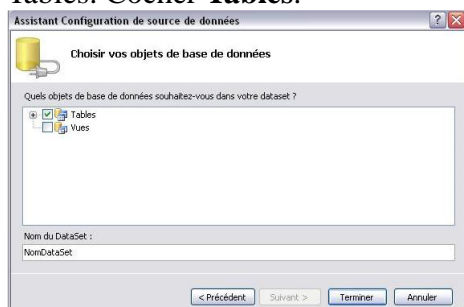
On va cliquer sur **Oui**.

VB vous propose ensuite d'enregistrer la chaîne de connexion (nom de la base...) dans le fichier de configuration afin qu'elles soient stockées et lues lors d'utilisation ultérieure (Automatiquement bien sur)

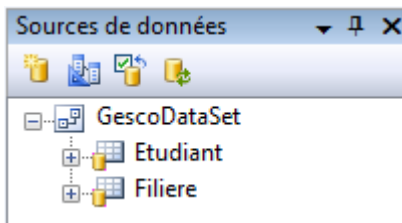
Cocher la case **Oui, enregistrer la connexion en tant que** (le nom ici es GescoConnectionString) et cliquer sur **Suivant**



Ensuite il faut choisir dans la base les objets qui seront chargés dans le Dataset. C'est habituellement les Tables. Cocher **Tables**.



Cliquer sur **Terminer**. Dans l'explorateur de projet, apparaît la source de données.



2. Créer un formulaire permettant d'ajouter des enregistrements à une table

Pour créer un formulaire d'accès aux données il faut

- Ouvrir le projet si ce n'est déjà fait et ajouter un formulaire
- Dans l'**Explorateur de solutions**, cliquez sur l'onglet *Sources de données*. Dans la fenêtre *Sources de données*, vous pouvez développer le nœud *Filiere* pour afficher les différents champs de la table *Filiere*.
- Faites glisser le nœud *Filiere* depuis la fenêtre *Sources de données* jusqu'au formulaire.

Remarque

Le nœud *Filiere* est une liste déroulante contenant les éléments *Détails* et *Tableau*.

- Choisir *Détails* pour afficher sur le formulaire la table en mode colonne simple (c'est-à-dire on ne peut visualiser qu'un seul enregistrement à la fois). L'affichage en mode *Détails* se fait à l'aide de contrôles zone de texte (*Textbox*) où chaque zone de texte représente un champ de la table.

- Choisir *Tableau* pour afficher sur le formulaire la table en mode tabulaire ou tableau (c'est-à-dire on peut visualiser plusieurs enregistrements à la fois). L'affichage en mode tableau se fait à l'aide d'un contrôle nommé *DataGridView* qui affiche les lignes et les colonnes de la table.

	Codfil	Libfil	Effmax
*			

Certains contrôles sont ajoutés automatiquement au formulaire. Plusieurs composants sont créés et ajoutés à la barre d'état des composants située sous le formulaire :

- FiliereBindingNavigator** il est créé à l'aide du contrôle *BindingNavigator* qui sert à la navigation dans la table.
- FiliereBindingSource** permet de se connecter à la base de données
- FiliereTableAdapter** gère la récupération et la mise à jour des données
- GescoDataSet** enregistre les données récupérées dans un **DataSet** local

La propriété *Dock* du contrôle *DataGridView* permet de spécifier la position du contrôle sur le formulaire.

Vous pouvez également apporter des modifications aux enregistrements en modifiant les données affichées dans la grille. Toutefois, ces modifications ne seront pas enregistrées, à moins que vous cliquiez sur l'icône *Enregistrer les données*. Dans la rubrique suivante, vous apprendrez comment enregistrer automatiquement les modifications apportées à vos données.

3. Mettre à jour le fichier de base de données local

Comme vous avez pu le remarquer, vous pouvez modifier les données des filières et même ajouter de nouvelles filières. Toutefois, si vous fermez et redémarrez le programme, ces modifications sont perdues.

Ces données étaient, en fait, des copies des données présentes dans la base de données, enregistrées dans le **DataSet** local. À chaque démarrage du programme, le **DataSet** récupère ces données de la base de données. Lorsque des modifications sont apportées au **DataSet**, elles ne sont pas répercutées dans la base de données.

Si vous cliquez sur le bouton **Enregistrer** du contrôle **FiliereBindingNavigator**, toutes les modifications sont copiées en retour du **DataSet** dans la base de données. Sachant qu'un utilisateur ne pense pas systématiquement à enregistrer son travail, ajoutez un code pour enregistrer automatiquement les modifications effectuées dans la base de données lors de la fermeture du programme. Tant que vous y êtes, modifiez également l'interface utilisateur pour faciliter la saisie des données.

- Dans l'**Explorateur de solutions**, sélectionner la base de données et assurez-vous que la propriété **Copier dans le répertoire de sortie** a la valeur **Copier si plus récent**.
- Dans l'**Explorateur de solutions**, sélectionner le formulaire contenant le contrôle **datagridview** créé précédemment, puis s'assurer qu'il est en mode conception (dans le menu **Affichage**, cliquer sur **Concepteur**).
- Sélectionner le contrôle **DatGridView** et le supprimer.
- Dans l'**Explorateur de solutions**, cliquer sur l'onglet **Sources de données**.
- Dans la fenêtre **Sources de données**, sélectionner la table **Filiere**, puis cliquer sur **Détails** dans la liste déroulante.
- Faire glisser le nœud **Filiere** de la fenêtre **Sources de données** au nouveau formulaire.
Des contrôles **TextBox** sont ajoutés pour chaque champ de la table, ainsi que des contrôles **Label** qui décrivent ces champs.
- Double-cliquer sur le formulaire pour ouvrir l'éditeur de code.
- Dans la liste déroulante **Événements**, cliquez sur **FormClosing**.
- Dans le gestionnaire d'événements **Form1_FormClosing**, tapez le code suivant :

```
Me.FiliereBindingSource.EndEdit()  
Me.FiliereTableAdapter.Update(Me.GescoDataSet.Filiere)
```
- Ce code contraint le composant **FiliereTableAdapter** à copier toute modification apportée au groupe de données dans la base de données locale.

4. Afficher les données provenant de tables différentes et liées

Dans cette leçon, vous apprendrez à afficher les données liées dans deux tables séparées sur un Windows Form.

Les quatre leçons précédentes décrivent comment créer une base de données, modifier et afficher des données de celle-ci. Lorsque vous installez SQL Server Compact 3.5 avec Visual Basic Express, vous avez accès à un exemple de base de données nommé Northwind.sdf, qui contient plusieurs tables.

• Etablir d'abord le lien entre ces deux tables

- Dans l'**Explorateur de solutions**, faire un clic droit sur **GescoDataSet** et choisir **Modifier le dataset à l'aide du concepteur**. Une fenêtre s'affiche contenant les différentes tables du dataset.
- Faire un clic droit n'importe où dans la fenêtre et choisir dans le menu **Ajouter**, la commande **Relation**. La fenêtre Relation s'ouvre.

- c. Choisir la table parente (Filiere - cardinalité maximum égale à N) et la table enfant (Etudiant - cardinalité maximum égale à 1)
- d. Dans le tableau **Colonnes**, choisir le ou les champs à utiliser pour mettre les deux tables en relation. Dans notre cas, choisir **codfil** comme **colonne clé** et comme **colonne clé étrangère**.

- **Afficher les données liées**

Lorsque deux tables sont liées, leur relation est visible dans la fenêtre **Sources de données**. Par exemple, si vous développez la table **Filiere**, vous remarquerez que la table **Etudiant** est affichée comme faisant partie de la table **Filiere**. Si vous faites glisser cette table **Etudiant** ou l'un de ses champs vers un **Windows Form**, vous pouvez afficher la relation entre la table ou le champ et la table **Filiere** sur le formulaire. Par exemple, vous pouvez vous déplacer à travers les filières dans la table **Filiere** et afficher les étudiants inscrits dans chaque filière.

Pour afficher les données liées

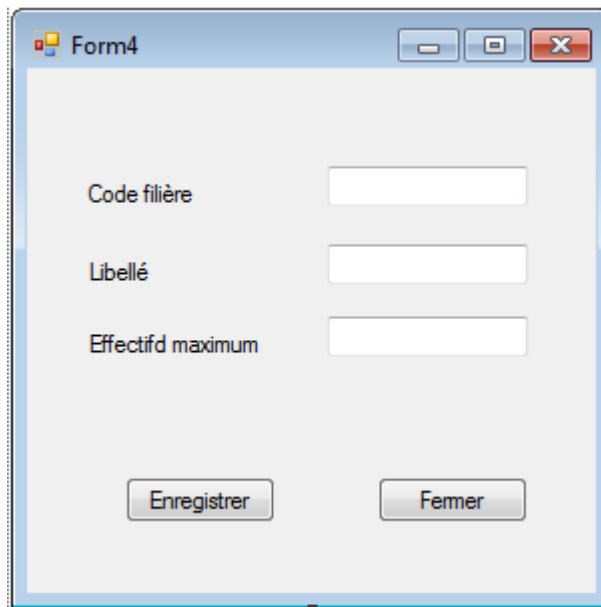
- a. Ajouter un nouveau formulaire au projet.
- b. Dans l'**Explorateur de solutions**, cliquez sur l'onglet **Sources de données**.
- c. Développer la table **Filiere**, sélectionner **Libfil**, cliquez sur la flèche de déroulement à côté de **Libfil**, puis cliquer sur **Étiquette**.
- d. Faire glisser le champ **Libfil** vers le **Windows Form**.
- e. Faire glisser la table **Etudiant** (qui se trouve dans la table **Filiere**) vers le formulaire et positionnez-la sous l'étiquette. Un contrôle **DataGridView** est ajouté au formulaire.
- f. Appuyez sur **F5** pour exécuter le programme.
- g. Lorsque l'application démarre, cliquez sur le bouton **Déplacer vers le bas** sur la barre d'outils située en haut du formulaire.
- h. Vérifiez que les étudiants qui figurent dans le contrôle **DataGridView** sont actualisés à chaque changement de filière.

V. Se connecter à une base de données en écrivant du code VB

- a. **Se connecter et ajouter des enregistrements à une table directement dans la base**
 - i. Ajouter un nouveau formulaire au projet
 - ii. Dessiner sur ce formulaire, trois zones d'étiquette (**Label**) et trois zones de texte (**Textbox**) ainsi que deux boutons de commande.
 - iii. Définir pour chacun des contrôles, les propriétés suivantes :

Contrôle	Propriété	Valeur
Label1	Text	Code filière
Label2	Text	Libellé
Label3	Text	Effectif Maximum
Textbox1	Name	Codfil
Textbox2	Name	Libfil
Textbox3	Name	Effmax
Button1	Text	Enregistrer
Button2	Text	Fermer

Ce qui nous donne la figure ci-dessous



- iv. Double cliquer sur le bouton **Enregistrer** pour accéder à la page de code et écrire le code suivant :

‘Importation des espaces de nom ‘des classes

Imports System

Imports System.Data

Imports System.Data.OleDb

Imports Microsoft.VisualBasic

Public Class Form4

‘Procédure événementielle exécutée lorsqu’on clique sur le bouton de commande enregistrer

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

‘On déclare un objet connexion

Dim VConnexion As OleDbConnection

‘On crée un nouvel objet connexion ou une nouvelle instance de l’objet connexion

VConnexion = New OleDbConnection

‘On définit la chaîne de connexion de notre objet connexion

Vconnexion.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb"

Vconnexion.open

‘On déclare objet command

Dim Vcommand As OleDbCommand

‘On crée une nouvelle instance de l’objet command

Vcommand=New OleDbCommand

‘On définit la connexion à utiliser pour exécuter la commande

Vcommand.Connection=Vconnexion

‘on spécifie le texte de la commande SQL qui sera exécutée par l’objet Command

Vcommand.CommandText = "Insert into filiere Values(' & Codfil.Text & ',' & Libfil.Text & ',' & Effmax.Text & ')"

‘On ouvre la connexion

VConnexion.Open()

‘on exécute la requête SQL de l’objet Command

Vcommand.ExecuteNonQuery()

'On vide les zones de texte pour les préparer à un nouvel enregistrement

```
Codfil.Text = ""
```

```
Libfil.Text = ""
```

```
Effmax.Text = ""
```

'On ferme la connexion

```
VConnexion.close()
```

```
End Sub
```

```
End Class
```

Remarque

Les phrases en italique sont des commentaires

Explication du code

- La première chose faite a été de créer un objet **Connection** et de lui affecter une chaîne de connexion. Cette opération a été effectuée à l'aide des trois instructions suivantes :

```
Dim VConnexion As
```

```
VConnexion = New OleDbConnection
```

```
Vconnexion.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb"
```

Ces trois instructions auraient pu être regroupées en une seule; c'est à dire, définir la propriété ConnectionString pendant qu'on crée une nouvelle instance de Connection et au moment de la déclaration. Ce qui donnera :

```
Dim VConnexion As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb")
```

- La deuxième opération a consisté à créer un objet command et à lui affecter ses propriétés à travers les instructions :

```
Dim Vcommand As OleDbCommand
```

```
Vcommand=New OleDbCommand
```

```
Vcommand.Connection=Vconnexion
```

```
Vcommand.CommandText = "Insert into filiere Values(" & Codfil.Text & "," & Libfil.Text & "," & Effmax.Text & ")"
```

Comme pour l'objet connexion, ces quatre instructions auraient pu s'écrire en une seule. Ce qui donnera :

```
Dim Vcommand As New OleDbCommand("Insert into filiere Values(" & Codfil.Text & "," & Libfil.Text & "," & Effmax.Text & ")", Vconnexion)
```

On aurait aussi pu aussi utiliser la méthode CreateCommand de l'objet connexion pour créer l'objet Command. Vcommand=Vconnexion.CreateCommand()

Ainsi, on aurait plus besoin de spécifier la connexion à utiliser pour exécuter l'objet Command

- Enfin, la dernière opération a été d'ouvrir la connexion, d'utiliser la méthode ExecuteNonQuery pour exécuter la requête SQL et de fermer la connexion.

b. Se connecter et ajouter des enregistrements à une table en important la table en VB à l'aide d'un Dataset

- i. Créer un nouveau formulaire identique au formulaire précédent.
- ii. Dans la partie déclaration du formulaire, c'est-à-dire juste après **Public Class** <Nom du formulaire>, saisir le code suivant qui permet de déclarer des variables publiques qui pourront être utilisées dans toutes les procédures relatives à ce formulaire ou à ses contrôles.

```

Dim VConnexion As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb")
Dim Vcommand As OleDbCommand = VConnexion.CreateCommand()
' Déclaration de l'Objet DataAdapter
Dim VDataAdapter As OleDbDataAdapter
' Déclaration de l'Objet DataSet. Attention au New
Dim VDataSet As New DataSet()
' Déclaration de l'Objet DataTable
Dim VDataTable As DataTable
' Déclaration de l'Objet DataRow c'est à dire ligne
Dim VDataRow As DataRow
' Numéro de la ligne en cours ou de l'enregistrement courant
Dim VRowNumber As Integer
' Pour recompiler les données modifiées avant de les remettre dans le DataAdapter
Dim VObjetCommandBuilder As OleDbCommandBuilder

```

Dans la procédure événementielle **Form_Load** du formulaire, Saisir le code suivant qui permettra de créer et de définir de nouvelles instances des objets déclarés plus haut.

```

Private Sub Form4_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
' on spécifie le texte de la commande SQL qui sera exécutée par l'objet Command
Vcommand.CommandText = "Select * from filiere"
' On ouvre la connexion
VConnexion.Open()
' Instancier un objet Adapter
VDataAdapter = New OleDbDataAdapter(Vcommand)
' Avec l'aide de la propriété Fill du DataAdapter charger le DataSet
VDataAdapter.Fill(VDataSet, "Filiere")
' Mettre dans un Objet DataTable une table du DataSet
VDataTable = VDataSet.Tables("Filiere")
' On initialise le numéro de l'enregistrement actif à -1 c'est à dire, pas d'enregistrement actif au départ
VRowNumber = -1
If VDataSet.Tables("Filiere").Rows.Count > 0 Then
' Si le nombre d'enregistrement de la table Filiere est supérieur à 0
' alors, le numéro de l'enregistrement actif est le numéro de la première ligne
VRowNumber = 0
' on charge les valeurs des champs du premier enregistrement dans les zones de texte
Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
End If

```

End Sub

Dans la procédure événementielle Button_Click (**Private Sub** Button3_Click(**ByVal** sender **As** System.Object, **ByVal** e **As** System.EventArgs) **Handles** Button3.Click) du bouton enregistrer, taper le code suivant :

```

        'On crée un nouvel enregistrement (une nouvelle ligne) vide
        VDataRow = VDataSet.Tables("Filiere").NewRow()
        'on le remplit avec les données des zones de texte
        VDataRow("Codfil") = Me.Codfil.Text
        VDataRow("Libfil") = Me.Libfil.Text
        VDataRow("effmax") = Me.Effmax.Text
        'on ajoute la ligne au dataset
        VDataSet.Tables("Filiere").Rows.Add(VDataRow)
        'Pour modifier les valeurs changées dans le DataAdapter
        VObjetCommandBuilder = New OleDbCommandBuilder(VDataAdapter)
        'Mise à jour
        VDataAdapter.Update(VDataSet, "Filiere")
        'On vide le DataSet et on le recharge de nouveau.
        VDataSet.Clear()
        VDataAdapter.Fill(VDataSet, "Filiere")
        VDataTable = VDataSet.Tables("Filiere")
        'On vide les zones de texte pour permettre la saisie d'un nouvel enregistrement
        Codfil.Text = ""
        Libfil.Text = ""
        Effmax.Text = ""

```

c. Consulter un enregistrement**1. Se positionner sur le premier enregistrement et l'afficher**

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Premier**

Dans la procédure événementielle **Premier_Click**, saisir le code suivant:

```

If VDataSet.Tables("Filiere").Rows.Count > 0 Then
    'Si le nombre d'enregistrement de la table Filiere est supérieur à 0
    'alors, le numéro de l'enregistrement actif est le numéro de la première ligne
    VRowNumber = 0
    'on charge les valeurs des champs du premier enregistrement dans les zones de texte
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
Else
    'Si le nombre d'enregistrement de la table Filiere est égal à 0
    'on affiche un message
    MsgBox("la table ne contient pas d'enregistrements")
    Exit Sub
End If

```

2. Se positionner sur le dernier enregistrement et l'afficher

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Dernier**

Dans la procédure événementielle **Dernier_Click**, saisir le code suivant :

```
If VDataTable.Rows.Count > 0 Then
    'Si le nombre d'enregistrement de la table Filiere est supérieur à 0
    'alors, le numéro de l'enregistrement actif est le numéro de la dernière ligne
    'c'est-à-dire le nombre total de ligne moins 1
    VRowNumber = VDataTable.Rows.Count - 1
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
Else
    MsgBox("La table ne contient pas d'enregistrements")
End If
```

3. Se positionner sur l'enregistrement suivant et l'afficher

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Suivant**

Dans la procédure événementielle **Suivant_Click**, saisir le code suivant:

```
If VRowNumber + 1 < VDataTable.Rows.Count Then
    'Si l'enregistrement actif n'est pas le dernier alors on lui ajoute 1
    VRowNumber = VRowNumber + 1
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
Else
    MsgBox("Vous êtes déjà sur le dernier enregistrement")
End If
```

4. Se positionner sur l'enregistrement précédent et l'afficher

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Précédent**

Dans la procédure événementielle **Précédent_Click**, saisir le code suivant:

```
If VRowNumber - 1 >= 0 Then
    'Si l'enregistrement actif n'est pas le Premier alors on lui soustrait 1
    VRowNumber = VRowNumber - 1
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
Else
    MsgBox("Impossible d'aller à l'enregistrement précédent")
End If
```

d. Modifier un enregistrement:

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Modifier**

Dans la procédure événementielle **Modifier_Click**, saisir le code suivant :

```
'Extraire l'enregistrement courant
VDataRow = VDataSet.Tables("Filiere").Rows(VRowNumber)
'Modifier les valeurs des champs en récupérant le contenu des TextBox
VDataRow("Codfil") = Me.Codfil.Text
VDataRow("Libfil") = Me.Libfil.Text
VDataRow("effmax") = Me.Effmax.Text
'Pour modifier les valeurs changées dans le DataAdapter
```

```

VObjetCommandBuilder = New OleDbCommandBuilder(VDataAdapter)
'Mise à jour
VDataAdapter.Update(VDataSet, "Filiere")
'On vide le DataSet et on le 'recharge' de nouveau.
VDataSet.Clear()
VDataAdapter.Fill(VDataSet, "Filiere")
VDataTable = VDataSet.Tables("Filiere")

```

e. Supprimer un enregistrement

Ajouter un bouton de commande au formulaire et définir ses propriétés **Name** et **Text** à **Supprimer**
 Dans la procédure événementielle **Supprimer_Click**, saisir le code suivant :

```

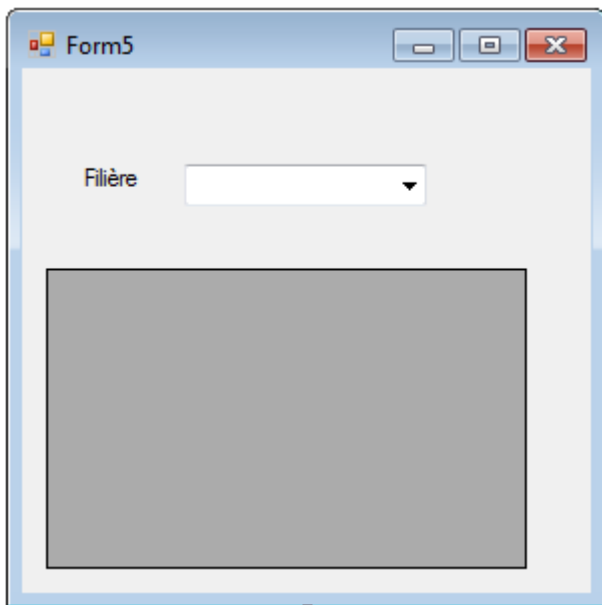
VDataSet.Tables("Filiere").Rows(VRowNumber).Delete()
VObjetCommandBuilder = New OleDbCommandBuilder(VDataAdapter)
VDataAdapter.Update(VDataSet, "Filiere")
If VRowNumber < VDataTable.Rows.Count Then
    'S'il existe un enregistrement après celui supprimé
    ' alors, ce dernier devient l'enregistrement courant et on l'affiche
    VRowNumber = VRowNumber
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
ElseIf VRowNumber - 1 >= 0 Then
    'S'il n'existe pas un enregistrement après celui supprimé mais qu'il
    ' en existe un avant, alors, ce dernier devient l'enregistrement courant et on l'affiche
    VRowNumber = VRowNumber - 1
    Codfil.Text = VDataTable.Rows(VRowNumber).Item("Codfil").ToString
    Libfil.Text = VDataTable.Rows(VRowNumber).Item("Libfil").ToString
    Effmax.Text = VDataTable.Rows(VRowNumber).Item("effmax").ToString
Else
    'si on est ni dans le premier ni dans le second cas, alors il n'y a plus
    ' d'enregistrement dans la table donc plus rien à afficher
    Codfil.Text = ""
    Libfil.Text = ""
    Effmax.Text = ""
End If

```

f. Remplir un DataGridView, un combobox ou une ListBox avec un DataSet

Exemple

On désire créer un formulaire qui affiche la liste des filières dans un combobox et la liste des étudiants de cette filière dans un datagrid. Chaque fois que l'utilisateur choisit une filière, les étudiants de cette filière s'affichent dans le datagrid



Dans la procédure événementielle Combobox1_dropdown, saisir le code suivant :

```

Dim VConnexion As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb")
Dim Vcommand As New OleDbCommand
Vcommand.CommandText = "Select * from filiere"
VConnexion.Open()
Dim VDataSet As New DataSet()
'Déclaration de l'Objet DataTable
Dim VDataTable As New DataTable
'Instancier un objet Adapter
Vcommand.Connection = VConnexion
Dim VDataAdapter As OleDbDataAdapter = New OleDbDataAdapter(Vcommand)
'initialiser l'objet Command
'Avec l'aide de la propriété Fill du DataAdapter charger le DataSet
VDataAdapter.Fill(VDataSet, "Filiere")
'Mettre dans un Objet DataTable une table du DataSet
ComboBox1.DataSource = VDataSet.Tables("Filiere")
ComboBox1.ValueMember = "Codfil"
ComboBox1.DisplayMember = "Libfil"
VConnexion.Close()

```

Dans la procédure événementielle Combobox1_SelectedIndexChange, saisir le code suivant :

```

Dim VConnexion As OleDbConnection = New
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data source=Gesco.mdb")
Dim Vcommand As OleDbCommand
'Dim Vcommand2 As OleDbCommand = VConnexion.CreateCommand()
VConnexion.Open()
'Déclaration de l'Objet DataAdapter
Dim VDataAdapter As OleDbDataAdapter
'Déclaration de l'Objet DataSet. Attention au New
Dim VDataSet As New DataSet()

```



```
' Déclaration de l'Objet DataTable
Dim VDataTable As New DataTable
' Déclaration de l'Objet DataRow c'est à dire ligne
Dim strSql As String = "SELECT Etudiant.* FROM Etudiant where codfil=" &
ComboBox1.SelectedValue.ToString & ""
Vcommand = New OleDbCommand(strSql)
VDataAdapter = New OleDbDataAdapter(Vcommand)
Vcommand.Connection = VConnexion
VDataAdapter.Fill(VDataSet, "Etudiant")
'Créer une datatable à partir du dataset
VDataTable = VDataSet.Tables("Etudiant")
'Mettre dans le DataGridView une table DataTable
DataGridView1.DataSource = VDataTable
```


Chapitre 5 : Gestion des Erreurs ou des Exceptions

Dans cette leçon, vous apprendrez comment créer un code de gestion des erreurs de base pour vos programmes.

Même les programmes les mieux conçus rencontrent parfois des bugs. Certaines erreurs sont des défauts au sein de votre code, qui peuvent être localisés, puis corrigés. D'autres erreurs sont une conséquence normale du fonctionnement du programme ; par exemple, votre programme peut tenter d'ouvrir un fichier qui est déjà en cours d'utilisation ou ouvrir une connexion déjà ouverte. Dans les cas de ce type, les erreurs peuvent être prévues, mais ne peuvent pas être empêchées. En tant que programmeur, votre travail consiste à prévoir ces erreurs et à aider votre programme à les traiter.

1. Erreurs d'exécution

Une erreur qui se produit pendant qu'un programme est en cours d'exécution est appelée *erreur d'exécution* (ou erreur de runtime). Une erreur d'exécution se produit lorsqu'un programme tente d'exécuter une opération pour laquelle il n'a pas été conçu. Par exemple, si votre programme tente d'exécuter une opération non valide, telle que la conversion d'une chaîne non numérique en valeur numérique, une erreur d'exécution se produit.

Lorsqu'une erreur d'exécution se produit, le programme publie une **Exception** qui gère les erreurs en recherchant dans le programme le code de gestion des erreurs correspondant. Si aucun code de ce type n'est détecté, le programme s'arrête et doit être redémarré. Cette situation pouvant conduire à une perte de données, il est recommandé de créer un code de gestion des erreurs en tout point où vous prévoyez qu'une erreur est susceptible de se produire.

2. Bloc Try...Catch...Finally

Vous pouvez utiliser le bloc **Try...Catch...Finally** pour gérer les erreurs d'exécution au sein de votre code. Vous pouvez appliquer **Try** à un segment de code ; si une exception est générée par ce code, le programme passe au bloc **Catch**. Le code présent dans le bloc **Catch** est alors exécuté. Une fois l'exécution de ce code terminée, tout code présent dans le bloc **Finally** est exécuté. Le bloc **Try...Catch...Finally** entier est fermé par une instruction **End Try**. L'exemple suivant illustre l'utilisation de chaque bloc :

Try

<Traitement 1>

Catch E as exception

<Traitement 2>

Finally

<Traitement 3>

End Try

En premier lieu, le code du bloc **Try** (<Traitement 1>) est exécuté. S'il s'exécute sans erreur, le programme ignore le bloc **Catch** (<Traitement 2>) et exécute le code dans le bloc **Finally** (<Traitement 3>). Si une erreur se produit dans le bloc **Try**, l'exécution passe immédiatement au bloc **Catch**. Le code présent dans ce bloc est alors exécuté. Ensuite, le code du bloc **Finally** est exécuté.

3. L'objet Err et l'exception E

L'objet **Err** et l'exception **E** du bloc **Catch** en Visual basic offre toutes les propriétés et méthodes relatives aux erreurs d'exécution. Il est très utile pour personnaliser le traitement à effectuer suivant les erreurs. Ainsi, on peut grâce à la propriété **Number**, récupérer le numéro d'une erreur, ou grâce à la propriété **Description** ou **Message**, récupérer la description d'une erreur, soit pour l'afficher, soit pour lui appliquer un traitement spécifique sans que le programme ne se plante ou sans que celui-ci ne s'interrompe brusquement.

4. Exemple

Retournons au formulaire permettant d'enregistrer des filières avec l'écriture de code VB. Lorsque l'utilisateur essaie d'enregistrer deux fois le même code filière, il se produit une erreur. « Enregistrement non effectué, risque de doublon..... ». Il en est de même si l'utilisateur au lieu de saisir un nombre dans la zone de texte **Effectif maximum**, saisit plutôt une chaîne de caractère. Ces deux erreurs ont le même numéro c'est-à-dire le numéro 5. Une telle erreur empêchera l'application de continuer à s'exécuter. Il faudra donc modifier le code comme suit :

```
Try
    VDataRow = VDataSet.Tables("Filiere").NewRow()
    VDataRow("Codfil") = Me.Codfil.Text
    VDataRow("Libfil") = Me.Libfil.Text
    VDataRow("effmax") = Me.Effmax.Text
    VDataSet.Tables("Filiere").Rows.Add(VDataRow)
    VObjetCommandBuilder = New OleDbCommandBuilder(VDataAdapter)
    VDataAdapter.Update(VDataSet, "Filiere")
Catch
    'Code qui sera exécuté en cas d'erreur
    If Err.Number = 5 Then
        'Si le numéro de l'erreur est 5 et qu'il s'agit de l'effectif max qui n'est pas un nombre
        If Not IsNumeric(Effmax.Text) Then
            MsgBox("Vous devez saisir une valeur numérique pour l'effectif maximum")
        End If
    Else
        'S'il s'agit d'une autre erreur qu'on avait pas prévue
        MsgBox(Err.Number & Chr(10) & Err.Description)
    End If
Finally
    'Code à exécuter dans tous les cas c'est à dire qu'il y ait erreur ou non
    VDataSet.Clear()
    VDataAdapter.Fill(VDataSet, "Filiere")
    VDataTable = VDataSet.Tables("Filiere")
    Codfil.Text = ""
    Libfil.Text = ""
    Effmax.Text = ""
End Try
```

Chapitre 6 : Les états ou rapport

I. Ajout de rapports affichables aux applications Visual Studio

Visual Studio prend en charge diverses solutions de création de rapport pour vous aider à ajouter des rapports de données élaborés à vos applications Visual Basic. Vous pouvez créer et ajouter des rapports à l'aide de contrôles ReportViewer, de Crystal Reports ou de SQL Server Reporting Services.

Remarque

SQL Server Reporting Services fait partie de SQL Server 2005 plutôt que de Visual Studio 2005. Reporting Services n'est pas installé sur votre système à moins que vous n'ayez installé SQL Server 2005.

a. Vue d'ensemble de la technologie de création de rapports de Microsoft dans les applications Visual Basic

Choisissez l'une des approches suivantes pour utiliser une technologie de création de rapports Microsoft dans votre application :

- Ajouter une ou plusieurs instances d'un contrôle ReportViewer à une application Windows Visual Basic.
- Intégrer SQL Server Reporting Services par programme en appelant le service Web Report Server.
- Utiliser conjointement le contrôle ReportViewer et Microsoft SQL Server 2005 Reporting Services, en utilisant le contrôle comme une visionneuse de rapports et un serveur de rapports comme processeur de rapport. (Notez que vous devez utiliser la version SQL Server 2005 de Reporting Services si vous souhaitez utiliser conjointement un serveur de rapports et le contrôle ReportViewer).

b. Utilisation de contrôles ReportViewer

La façon la plus simple d'incorporer les fonctionnalités de rapport dans une application Windows Visual Basic est d'ajouter le contrôle ReportViewer à un formulaire dans votre application. Le contrôle ajoute directement des fonctions de traitement de rapports à votre application et fournit un Concepteur de rapports intégré afin que vous puissiez générer des rapports à l'aide de données provenant de n'importe quel objet de données ADO.NET. Une API complète fournit l'accès par programme au contrôle et aux rapports afin que vous puissiez configurer les fonctionnalités à l'exécution.

ReportViewer fournit le traitement intégré de rapports et une fonction d'affichage dans un contrôle de données unique et librement distribuable. Choisissez des contrôles ReportViewer si vous avez besoin des fonctionnalités de rapport suivantes :

- Traitement des rapports dans l'application cliente. Un rapport traité apparaît dans une zone d'affichage fournie par le contrôle.
- Liaison des données à des tables de données ADO.NET. Vous pouvez créer des rapports qui utilisent des instances de DataTable fournies au contrôle. Vous pouvez également créer directement une liaison à des objets métier.
- Contrôles redistribuables que vous pouvez inclure dans votre application.
- Fonctionnalités au moment de l'exécution, telles que la navigation entre les pages, l'impression, la recherche et les formats d'exportation. Une barre d'outils ReportViewer fournit la prise en charge de ces opérations.

Pour utiliser le contrôle ReportViewer, vous pouvez le faire glisser de la section **Données** de la Boîte à outils Visual Studio vers un formulaire dans votre application Windows Visual Basic.

c. Création de rapports dans Visual Studio pour des contrôles ReportViewer

Pour générer un rapport qui s'exécute dans ReportViewer, ajoutez un modèle de **Rapport** à votre projet. Visual Studio crée un fichier de définition de rapport client (.rdlc), ajoute le fichier à votre projet et ouvre un Concepteur de rapports intégré dans l'espace de travail Visual Studio.

Le Concepteur de rapports Visual Studio s'intègre dans la fenêtre **Sources de données**. Lorsque vous faites glisser un champ de la fenêtre **Sources de données** vers le rapport, le Concepteur de rapports copie les métadonnées de la source de données dans le fichier de définition du rapport. Ces métadonnées sont utilisées par le contrôle ReportViewer pour générer automatiquement le code de liaison de données.

Le Concepteur de rapports Visual Studio n'inclut pas de fonctionnalité d'aperçu du rapport. Pour afficher un aperçu de votre rapport, exécutez l'application et affichez l'aperçu du rapport qu'elle contient.

d. Pour ajouter des fonctionnalités de rapport de base à votre application

- Faites glisser un contrôle ReportViewer de l'onglet **Données** de la **Boîte à outils** sur votre formulaire.
- Dans le menu **Projet**, choisissez **Ajouter un nouvel élément**. Dans la boîte de dialogue **Ajouter un nouvel élément**, sélectionnez l'icône **Rapport**, puis cliquez sur **Ajouter**.
Le Concepteur de rapports s'ouvre dans l'environnement de développement et un fichier de rapport (.rdlc) est ajouté au projet.
- Faites glisser les éléments de rapport de la **Boîte à outils** vers la disposition de rapport et placez-les comme vous le souhaitez.
- Faites glisser les champs de la fenêtre **Sources de données** vers les éléments de rapport dans la disposition de rapport.

e. Étapes suivantes de l'utilisation de ReportViewer et du Concepteur de rapports

Pour	Consultez
Ajouter et configurer un contrôle ReportViewer	Ajout et configuration de contrôles ReportViewer
Créer et afficher l'aperçu d'un rapport local	Création de fichiers de définition de rapport client (.rdlc)
Ajouter des sources de données pour des rapports ReportViewer	Création de sources de données pour un rapport
Définir la disposition d'un rapport	Définition d'une mise en page de rapport (Concepteur de rapports Visual Studio)
Utiliser des expressions Visual Basic dans les rapports ReportViewer	Utilisation d'expressions dans un rapport (Concepteur de rapports Visual Studio)
Placer du code personnalisé dans un rapport	Ajout de code personnalisé à un rapport (Concepteur de rapports Visual Studio)
Distribuer des rapports et des contrôles ReportViewer dans le cadre de votre application	Déploiement de rapports et de contrôles ReportViewer

Imprimer un rapport	Impression des rapports depuis ReportViewer
---------------------	---

Procédure pas à pas : Créer un rapport ReportViewer

Cette procédure pas à pas montre comment créer un rapport tabulaire simple dans un projet d'application Windows Microsoft Visual Studio 2008 reposant sur l'exemple de base de données Gesco. Vous allez ajouter un modèle de rapport à votre projet, définir des informations de connexion pour la base de données Gesco, définir une requête, ajouter une région de données de tableau et ajouter un contrôle Windows Form ReportViewer à un formulaire Windows afin que le rapport puisse être visualisé par les utilisateurs de l'application.

1. Pour créer un projet d'application Windows

- Ouvrez Visual Studio. Dans le menu **Fichier**, pointez sur **Nouveau**, puis sélectionnez **Projet**.
- Dans le volet Types de projets, choisissez **Visual Basic**.
- Dans le volet Modèles, choisissez **Application Windows** pour créer une application Microsoft Windows.
- Dans la zone **Nom**, tapez **SimpleReport**.
- Dans la zone **Emplacement**, entrez le répertoire dans lequel vous désirez enregistrer votre projet ou cliquez sur le bouton **Parcourir** pour naviguer jusqu'à lui. Le Concepteur Windows Forms s'ouvre et affiche le formulaire **Form1** du projet que vous avez créé.
- Cliquez sur le formulaire. Dans le menu **Affichage**, choisissez **Fenêtre Propriétés**. Développez la propriété **Size** pour afficher **Width** et **Height**. Affectez à **Width** la valeur 500 pixels.

2. Pour définir une connexion de source de données et une table de données

- Dans l'Explorateur de solutions, cliquez avec le bouton droit sur le projet nommé **SimpleReport** (pas la solution), pointez sur **Ajouter**, puis sélectionnez **Nouvel élément**. Si la fenêtre **Explorateur de solutions** n'est pas visible, dans le menu **Affichage**, cliquez sur **Explorateur de solutions**.
- Dans la boîte de dialogue **Ajouter un nouvel élément**, cliquez sur **DataSet**. Tapez un nom pour le dataset, puis cliquez sur **Ajouter**. Le nom par défaut est **DataSet1.xsd**. Un nouveau fichier XSD est ajouté au projet et le Concepteur de DataSet s'ouvre.
- Dans le menu **Affichage**, cliquez sur **Concepteur**. Ouvrez la boîte à outils, et faites glisser un contrôle **TableAdapter** sur l'aire de conception de DataSet. L'**Assistant Configuration de TableAdapter** démarre.
- Dans la page **Choisir votre connexion de données**, cliquez sur **Nouvelle connexion**.
- Dans la page **Ajouter une connexion**, procédez comme suit :
 - Dans la zone **Source de données**, sélectionnez **Microsoft Access**.
 - Dans la zone **Nom de la base**, cliquez sur le bouton **Parcourir** pour aller chercher et sélectionner la base de données **Gesco**.
 - Cliquez sur **OK** pour revenir dans l'Assistant, puis cliquez sur **Suivant**.
- Dans la page **Enregistrer la chaîne de connexion dans le fichier de configuration de l'application**, tapez un nom pour la chaîne de connexion ou acceptez la valeur par défaut **GescoConnectionString**. Cliquez sur **Suivant**.
- Dans la page **Choisissez un type de commande**, sélectionnez **Utiliser des instructions SQL**, puis cliquez sur **Suivant**.

- h. Dans la page **Entrez une instruction SQL**, entrez la requête SQL ou cliquez sur le bouton Générateur de requête pour utiliser l'assistant concepteur de requête pour générer la requête SQL.

```
SELECT      Etudiant.Matricule, Etudiant.Nom, Etudiant.Prenom, Etudiant.Sexe,
Filiere.Libfil
FROM      (Etudiant INNER JOIN Filiere ON Etudiant.Codfil = Filiere.Codfil)
```

- i. cliquez sur **Terminer**
- j. Vous pouvez également cliquer sur le bouton **Générateur de requêtes** et utiliser le Générateur de requêtes pour créer une requête et la valider à l'aide du bouton **Exécuter la requête**.
- k. Le Concepteur de Dataset affiche désormais la définition **DataTable** pour **DataTable1** avec les champs nommés à partir des colonnes et des alias de colonnes de la requête (Matricule, Nom, Prenom, Sexe, Libfil). Vous devez utiliser ces champs à partir de la fenêtre Source de données lorsque vous liez des données aux régions de données de rapport.

Remarque

Si vous devez modifier les champs de la table de données, cliquez avec le bouton droit sur l'en-tête **DataTable1** ou **DataTable1TableAdapter** dans la page Concepteur de DataSet. Choisissez **Configurer**, l'Assistant Configuration de TableAdapter redémarre.

3. Pour ajouter un fichier de définition de rapport

- Dans le menu **Projet**, pointez sur **Ajouter Nouvel élément**.
- Dans la boîte de dialogue **Ajouter un nouvel élément**, cliquez sur **Rapport**.
- Dans la zone **Nom**, tapez **Etudiant.rdlc**, puis cliquez sur **Ajouter** pour ouvrir une aire de conception graphique.

L'aire de conception graphique fait partie du composant Concepteur de rapports de Visual Studio 2008.

4. Pour ajouter un tableau à la mise en page de rapport

- Avec **Etudiant.rdlc** en mode Design graphique, dans le menu **Affichage**, sélectionnez la boîte à outils si elle n'était pas ouverte. La boîte à outils s'ouvre.
- Dans la section **Données ou Eléments du rapport** de la boîte à outils, cliquez sur **Tableau**, puis cliquez sur l'aire de conception de rapports. Le Concepteur de rapports affiche un tableau composé de trois colonnes, couvrant la largeur du rapport.
- Cliquez sur le tableau pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau.
- Cliquez avec le bouton droit sur la poignée de la première colonne, puis cliquez sur **Insérer une colonne à gauche**.
- Dans la fenêtre Propriétés de **table1**, développez le nœud **Size**. Par défaut, la fenêtre Propriétés est ancrée sous l'Explorateur de solutions. Vous pouvez également ouvrir cette fenêtre dans le menu **Affichage** en sélectionnant Fenêtre Propriétés.
- Attribuez à la propriété **Width** du nœud **Size** la valeur **4,8 in**. Ce paramètre définit la largeur du tableau et égalise la largeur des colonnes à afficher dans le formulaire.
- Dans le Concepteur de rapports, cliquez sur l'aire de conception.
- Dans la fenêtre Propriétés, développez le nœud **Size** et attribuez à **Width** la valeur **5 in**.

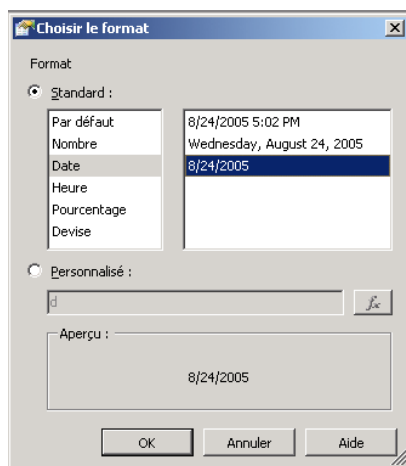
- i. Dans la fenêtre Sources de données, cliquez sur le nœud DataTable1 pour le développer et afficher les champs de données.
- j. Faites glisser le champ Nom de la fenêtre Sources de données vers la ligne (de détails) du milieu de la première colonne du tableau.
- k. Lorsque vous déposez le champ dans la cellule du milieu, deux événements se produisent.
 - Premièrement, la cellule de détails affiche le texte suivant : **=Fields!Nom.Value**. Ce texte est une expression de champ qui spécifie les valeurs de données du champ **Nom**. Les champs ajoutés à une ligne de **détails** sont toujours spécifiés sous forme d'expressions.
 - Deuxièmement, une valeur d'en-tête de colonne est placée automatiquement dans la première ligne, juste au-dessus de l'expression de champ. Par défaut, la colonne est générée à partir du nom du champ.
- l. Faites glisser les autres champs de la fenêtre Source de données vers les lignes (de détails) correspondantes du milieu du tableau.

5. Ajouter le contrôle ReportViewer à votre formulaire

- a. Dans l'Explorateur de solutions, cliquez sur **Form1.vb**.
- b. Dans le menu **Affichage**, choisissez **Concepteur**.
- c. Dans la section **Données** de la boîte à outils, faites glisser le contrôle ReportViewer vers le formulaire.
- d. Cliquez dans le formulaire. Dans la fenêtre Propriétés, développez **Size**. Définissez les propriétés de largeur (Width) et hauteur (Height) selon vos besoins.
- e. Ouvrez le panneau des balises actives du contrôle ReportViewer en cliquant sur le triangle dans le coin supérieur droit du contrôle. Cliquez sur la liste déroulante **Choisir un rapport**, puis sélectionnez **Etudiant.rdlc**.
- f. Dans le panneau des balises actives, cliquez sur **Ancrer dans le conteneur parent**.
- g. Dans la partie restante de cette procédure pas à pas, vous pouvez créer votre application et afficher le rapport dans le formulaire à tout moment. Si vous souhaitez voir de quelle manière les modifications incrémentielles de votre conception de rapport affecte le rapport final, créez et affichez le rapport en tant que dernière étape de chacune des procédures suivantes.
- h. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport dans le formulaire.

6. Pour mettre en forme un champ de date

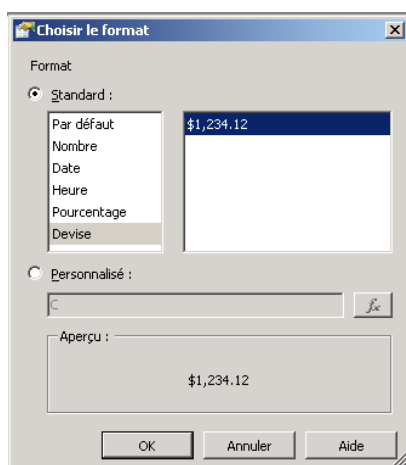
- a. Cliquez avec le bouton droit sur la cellule qui contient l'expression de champ **OrderDate**, puis cliquez sur **Propriétés**. La boîte de dialogue **Propriétés de la zone de texte** apparaît.
- b. Sélectionnez l'onglet **Format** et cliquez sur le bouton Parcourir (...) pour ouvrir la boîte de dialogue **Choisir le format**.
- c. Dans la zone **Format**, sélectionnez **Standard, Date**, puis sélectionnez un format de date.
- d. Cliquez sur **OK** pour fermer la boîte de dialogue **Choisir le format**, puis cliquez à nouveau sur **OK** pour fermer la boîte de dialogue **Propriétés de la zone de texte**.



- e.
- f. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Dans ce cas, la colonne de date s'affiche avec la mise en forme que vous venez de lui appliquer.

7. Pour mettre en forme un champ de devise

- Cliquez avec le bouton droit sur la cellule qui contient l'expression de champ à formater, puis cliquez sur Propriétés.
- Sélectionnez l'onglet **Format** et cliquez sur le bouton **Parcourir (...)** pour ouvrir la boîte de dialogue **Choisir le format**.
- Dans la zone **Format**, sélectionnez **Standard**, **Devise**, puis sélectionnez un format monétaire.
- Cliquez sur **OK**, puis cliquez à nouveau sur **OK** pour fermer la boîte de dialogue Propriétés de la zone de texte.



- e. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Dans ce cas, les **valeurs de la colonne Total Sales s'affichent dans un format monétaire**.

8. Pour mettre en forme des en-têtes de tableau

- Cliquez sur le tableau pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau.

Remarque

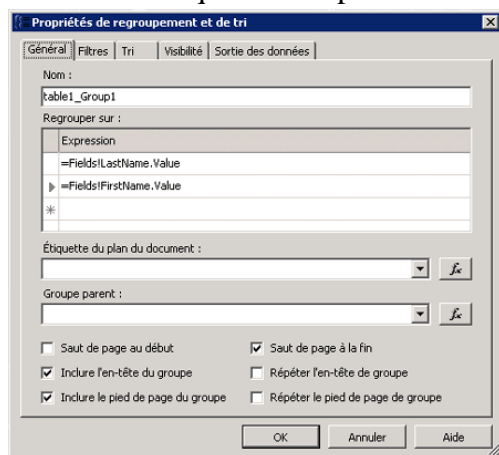
Les poignées sont des carrés gris qui apparaissent au-dessus et en regard du tableau. Les

poignées vous permettent d'effectuer différentes actions sur les colonnes, les lignes et le tableau lui-même. Les poignées qui s'étendent le long de la partie supérieure du tableau sont des poignées de colonne. Les poignées qui s'étendent le long du bord du tableau sont des poignées de ligne. La poignée qui se trouve à l'intersection des poignées de colonne et de ligne s'appelle la poignée d'angle. Pour voir un exemple de poignées de tableau, consultez Ajout de régions de données de tableau (Concepteur de rapports Visual Studio).

- Sélectionnez les poignées de la première ligne, qui contient les étiquettes d'en-tête de colonne, puis cliquez sur **Gras**.
- Cliquez sur la ligne d'en-tête du tableau, puis sur **Couleur d'arrière-plan**. Cliquez sur l'onglet **Web** et sélectionnez **MistyRose**. Cliquez sur **OK**.
- (Facultatif) Cliquez sur la touche F5 pour créer votre application et visualiser le rapport. Le diagramme suivant montre le rapport mis en forme.

9. Pour définir un groupe pour un rapport tabulaire

- Cliquez sur le tableau pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau.
- Cliquez avec le bouton droit sur la poignée de n'importe quelle ligne, puis cliquez sur **Insérer un groupe**.
- Sous l'onglet **Général**, pour **Grouper sur**, sélectionnez **=Fields!LastName.Value** dans la première ligne et **=Fields!FirstName.Value** dans la deuxième ligne. Cette opération permet de regrouper les données par nom de délégué commercial. Notez que vous pouvez regrouper les données selon des champs qui ne sont pas utilisés dans la région de données de tableau mais qui sont disponibles dans les champs **DataTable1**.



- Cliquez sur **OK**.

Un en-tête et un pied de page de groupe sont ajoutés au tableau.

10. Pour synthétiser les données par groupe

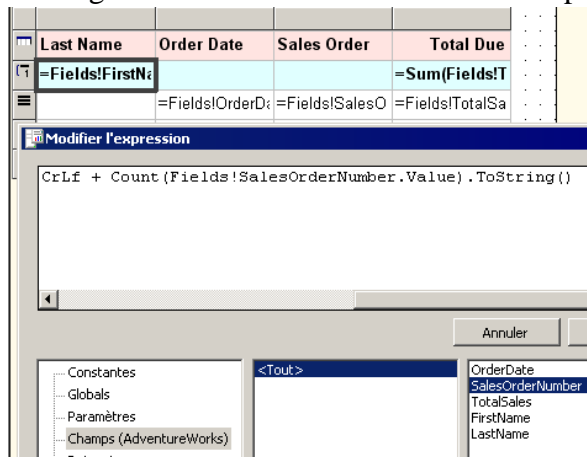
- Cliquez sur la ligne d'en-tête de groupe, puis sur **Couleur d'arrière-plan**. Choisissez l'onglet **Web** et sélectionnez **LightCyan**. Cliquez sur **OK**.
- Remplacez le champ **TotalSales** répété à chaque ligne par une expression dans l'en-tête de groupe qui représente un total combiné du groupe.
- Cliquez avec le bouton droit sur l'en-tête de groupe pour **Total Sales** et choisissez **Propriétés**. La boîte de dialogue **Propriétés de la zone de texte** s'ouvre.
- Copiez le texte ci-dessous et collez-le dans la liste déroulante **Valeur**.

=Sum(Fields!TotalSales.Value)

- c. Appliquez la mise en forme monétaire à cette zone de texte.
- c. Remplacez le nom répété à chaque ligne par une expression dans l'en-tête de groupe qui inclut deux lignes. La première ligne correspond au prénom et au nom. La deuxième ligne correspond au nombre des ventes dont la valeur n'était pas NULL.
- a. Sélectionnez l'expression dans la ligne de détails **Last Name** et supprimez-la.
- b. Cliquez avec le bouton droit dans la ligne d'en-tête de groupe correspond à Last Name et choisissez **Expression**. Copiez le texte suivant et collez-le dans la zone de dialogue **Modifier l'expression**.

```
Fields!FirstName.Value + " " + Fields!LastName.Value + ": " + vbCrLf +
Count(Fields!SalesOrderNumber.Value).ToString()
```

- d. Le diagramme suivant montre l'éditeur d'expression après cette étape.

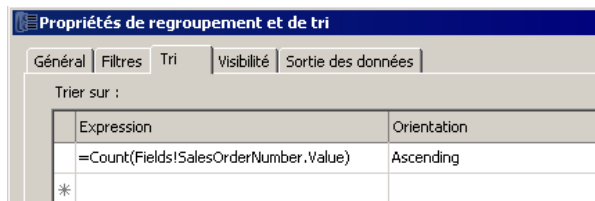


- e.
- f. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Le diagramme ci-dessous montre les lignes du tableau regroupées par nom de délégué commercial. Chaque en-tête de groupe résume les lignes de détails du groupe, en affichant le nom du délégué commercial, le nombre de ventes et la quantité totale vendue.

Last Name	Order Date	Sales Order Number	Total Due
Tsvi Reiter: 429			9 629 926,90 €
	01/07/2001	SO43659	27 231,55 €
	01/07/2001	SO43660	1 716,18 €
	01/07/2001	SO43681	18 274,86 €
	01/07/2001	SO43684	7 420,92 €
	01/07/2001	SO43685	3 635,46 €
	01/07/2001	SO43694	27 325,62 €

11. Pour trier les groupes dans un rapport tabulaire

- a. Sélectionnez la région de données de table, puis ouvrez **Propriétés du tableau**.
- b. Cliquez sur l'onglet **Groupes**. Le seul groupe défini, **table1_Group1**, est automatiquement sélectionné. Cliquez sur **Modifier**. La boîte de dialogue **Regroupement et tri** s'ouvre. Vérifiez que la boîte de dialogue ouverte est **Propriétés de regroupement et de tri**, et non **Propriétés du tableau**. L'onglet **Tri** dans la boîte de dialogue **Propriétés du tableau** contrôle le mode de tri des lignes de détails, et non celui des groupes. Vous allez définir le tri des données de détail dans la procédure suivante.
- c. Cliquez sur l'onglet **Tri**. Dans la zone **Trier sur**, sélectionnez **< Expression... >** dans la liste déroulante. Tapez l'expression à trier dans la zone **Modifier l'expression**. Par exemple :
=Count(Fields!SalesOrderNumber.Value)
- d. Cliquez sur **OK**.



- e. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Le rapport est maintenant trié en fonction du nombre de ventes réalisées par chaque délégué commercial.

Last Name	Order Date	Sales Order Number	Total Due
Syed Abbas: 16			242 093,22 €
	01/09/2003	SO53485	85 652,33 €
	01/09/2003	SO53492	25 139,98 €
	01/09/2003	SO53502	45 338,76 €
	01/09/2003	SO53554	4 544,18 €
	01/09/2003	SO53588	74,88 €
	01/09/2003	SO53594	1 465,51 €

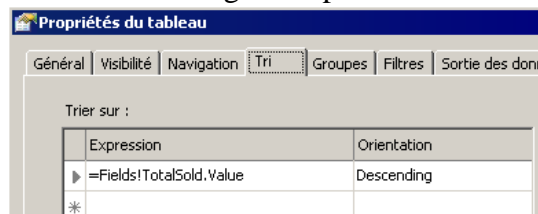
12. Pour trier les lignes de détails d'un groupe dans un rapport tabulaire

- Cliquez sur le tableau pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau.
- Cliquez avec le bouton droit sur la poignée d'angle, puis cliquez sur **Propriétés**. La boîte de dialogue **Propriétés du tableau** s'ouvre.

La poignée d'angle se trouve à l'intersection des poignées de colonne et de ligne.

- Sous l'onglet **Tri**, dans la zone **Trier sur**, sélectionnez **=Fields!TotalSold.Value**. Dans la zone **Direction**, sélectionnez **Décroissant**. Les données de détail sont triées par quantité vendue en commençant par les valeurs les plus élevées.
- Cliquez sur **OK**.

La boîte de dialogue Propriétés du tableau doit être similaire au diagramme suivant.



- e. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Le diagramme suivant présente la page 2 du rapport obtenu, pour montrer les groupes contenant plusieurs lignes de détails.

Last Name	Order Date	Sales Order Number	Total Due
Syed Abbas: 16			242 093,22 €
	01/09/2003	SO53485	85 652,33 €
	01/09/2003	SO53502	45 338,76 €
	01/12/2003	SO58915	36 317,54 €
	01/09/2003	SO53492	25 139,98 €
	01/05/2004	SO69564	21 479,49 €
	01/12/2003	SO59045	4 760,07 €

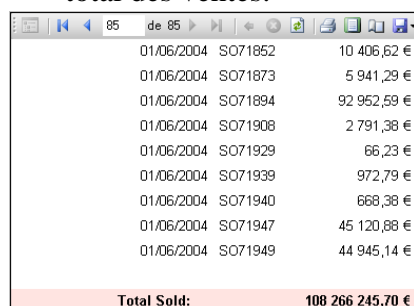
13. Pour ajouter des données de synthèse au pied de page dans un rapport tabulaire

1. Sélectionnez les cellules Sales Order et Total Sales dans la ligne de pied de page du tableau. Cliquez avec le bouton droit sur les cellules sélectionnées et choisissez **Fusionner les cellules**. Cette procédure libère de l'espace pour mettre en forme la somme de toutes les ventes pour le tableau.
2. Cliquez avec le bouton droit dans la cellule unique et fusionnée et choisissez **Propriétés**.
3. Dans la liste déroulante **Valeur**, tapez l'expression suivante :
=Sum(Fields!TotalSales.Value)
4. Appliquez la mise en forme monétaire à la cellule fusionnée.
5. Cliquez dans la cellule de zone de texte située en regard des cellules fusionnées, et créez une étiquette. Par exemple, tapez **Total vendu :**. Notez qu'il ne s'agit pas d'une expression, mais d'un simple texte.
6. (Facultatif) Définissez la couleur d'arrière-plan sur la ligne de pied de page pour qu'elle corresponde à celle de la ligne d'en-tête.

Le diagramme suivant montre la définition de table.

Last Name	Order Date	Sales	Total Sold
=Fields!FirstNa			=Sum(Fields!Tot
	=Fields!Order	=Fields!Sal	=Fields!TotalSold.
	Total Sold:	=Sum(Fields!TotalSold.Valu	

7. (Facultatif) Appuyez sur la touche F5 pour créer votre application et visualiser le rapport. Dans le rapport rendu, cliquez sur le bouton **LastPage** dans la barre d'outils Rapports pour accéder à la dernière page du rapport. Faites défiler vers le bas pour afficher la valeur du total des ventes.



01/06/2004	SO71852	10 406,62 €	
01/06/2004	SO71873	5 941,29 €	
01/06/2004	SO71894	92 952,59 €	
01/06/2004	SO71908	2 791,38 €	
01/06/2004	SO71929	66,23 €	
01/06/2004	SO71939	972,79 €	
01/06/2004	SO71940	668,38 €	
01/06/2004	SO71947	45 120,88 €	
01/06/2004	SO71949	44 945,14 €	
Total Sold:		108 266 245,70 €	

II. Interface utilisateur du Concepteur de rapports (Visual Studio)

Cette section fournit l'aide accessible via la touche F1 des boîtes de dialogue permettant de créer des fichiers de définition de rapport client (.rdlc) pour les contrôles ReportViewer dans Visual Studio.

1. Principales propriétés

Action

Spécifie la propriété Action d'une zone de texte ou d'une image. La définition de la propriété Action permet de lier la zone de texte ou l'image à un élément du rapport actuel, d'un autre rapport ou d'une page Web externe.

Propriétés du graphique (onglet Général)

Appliquez un style et une mise en forme, définissez des filtres et des valeurs de données et spécifiez les étiquettes d'axe des graphiques dans une mise en page de rapport. La boîte de dialogue **Propriétés du graphique** fournit une interface à onglets qui organise les propriétés de graphique en catégories.

Choisir le format

Spécifiez les propriétés de format sur une zone de texte pour déterminer l'aspect des dates, nombres et valeurs en devise au moment de l'exécution.

Vue design

Décrit l'aire de conception graphique qui permet de créer les fichiers de définition de rapport client (.rdlc).

Modifier une valeur du graphique (onglet Valeurs)

Modifiez les valeurs de données utilisées dans une instance de graphique ; vous pouvez spécifier des actions, changer l'aspect et modifier les étiquettes.

Modifier les classes

Mettez à jour les classes dans les assemblys externes que vous appelez à partir d'expressions dans un rapport.

Modifier les références

Ajoutez ou supprimez des références aux assemblys personnalisés que vous utilisez dans une définition de rapport.

Images incorporées

Incorporez des images dans un rapport pour fournir un ensemble portable d'images qui demeurent avec le rapport. Lorsque vous incorporez des images, le rapport devient la source des fichiers image utilisés dans le rapport.

Expression

Utilisez un éditeur d'expression pour écrire ou modifier des expressions évaluées au moment de l'exécution.

Filtres

Spécifiez des filtres qui incluent ou excluent des lignes de données dans une table de données ou un objet métier utilisé dans un rapport.

Propriétés de regroupement et de tri (onglet Général)

Regroupez des éléments dans une région de données et définissez des ordres de tri sur les groupes. La boîte de dialogue **Propriétés de regroupement et de tri** fournit une interface à onglets qui organise les propriétés en catégories.

Propriétés de l'image (onglet Général)

Définissez les options de navigation et de visibilité pour créer une image bascule qui affiche ou masque une zone du rapport.

Propriétés de ligne (onglet Général)

Spécifiez le style et les dimensions d'un élément de rapport (ligne) dans la mise en page du rapport.

Propriétés de la liste (onglet Général)

Configurez les paramètres d'une liste pour filtrer les données qu'elle contient. Vous pouvez spécifier les paramètres de visibilité et de navigation pour déterminer si la liste est visible en permanence ou uniquement par le biais d'éléments de bascule.

Propriétés de la matrice (onglet Général)

Appliquez un style et une mise en forme, définissez des filtres et des valeurs de données et spécifiez les étiquettes d'axe d'une matrice que vous ajoutez à une mise en page de rapport. La boîte de dialogue **Propriétés de la matrice** fournit une interface à onglets qui organise les propriétés de matrice en catégories.

Paramètres

Spécifiez un nom et une valeur de paramètre.

Propriétés du rectangle (onglet Général)

Configurez les paramètres d'un rectangle qui contient des éléments de rapport. Vous pouvez spécifier les paramètres de visibilité et de navigation pour déterminer si le rectangle est visible en permanence ou uniquement par le biais d'éléments de bascule.

Sources de données du rapport

Mettez à jour les informations de source de données stockées dans un fichier de définition de rapport.

Paramètres du rapport

Définissez les paramètres de rapport servant à filtrer les données de rapport et à associer des rapports par le biais de valeurs de données.

Propriétés du rapport (onglet Général)

Spécifiez les propriétés globales d'un rapport telles que l'auteur, la description et les options de mise en page permettant d'aligner les éléments du rapport. Vous pouvez également ajouter des références aux assemblés et fonctions personnalisés à utiliser dans le rapport. La boîte de dialogue **Propriétés du rapport** fournit une interface à onglets qui organise les propriétés de rapport en catégories.

Propriétés de style (onglet Bordure et ligne)

Spécifiez les propriétés de style de bordure, de couleur et de largeur.

Propriétés de style (onglet Remplissage)

Utilisez les options de cette boîte de dialogue pour définir les propriétés des éléments de graphique.

Propriétés de style (onglet Police)

Spécifiez les attributs de police et de famille de polices.

Propriétés du sous-rapport (onglet Général)

Configurez les paramètres fournissant les informations de connexion qui associent le sous-rapport au rapport parent.

Propriétés du tableau (onglet Général)

Appliquez un style et une mise en forme, définissez des filtres et des valeurs de données et spécifiez les étiquettes d'axe des tableaux que vous ajoutez à une mise en page de rapport. La boîte de dialogue **Propriétés du tableau** fournit une interface à onglets qui organise les propriétés de tableau en catégories.

Propriétés de la zone de texte (onglet Général)

Configurez une zone de texte pour spécifier la mise en forme des données, les styles, les actions et d'autres options d'interactivité. La boîte de dialogue **Propriétés de la zone de texte** fournit une interface à onglets qui organise les propriétés de zone de texte en catégories.

2. Boîte de dialogue Action (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Action** pour définir la propriété Action d'une zone de texte ou d'une image. La propriété **Action** est disponible dans la fenêtre Propriétés lorsqu'une zone de texte ou une image est sélectionnée. Au moment de l'exécution, l'action est effectuée lorsqu'un utilisateur clique sur l'élément dans le rapport.

Options

Aucune

Choisissez cette option pour indiquer qu'aucune action n'est associée à l'élément. Valeur par défaut.

Aller au rapport

Choisissez cette option pour définir un lien vers un rapport d'extraction. Vous pouvez sélectionner un fichier de définition de rapport client (.rdlc) ayant été ajouté au projet, ou vous pouvez taper le nom du chemin d'accès relatif ou complet d'un fichier qui est stocké sur le système de fichiers.

Paramètres

Choisissez cette option pour ajouter une liste de paramètres à passer au rapport d'extraction. Les noms des paramètres doivent correspondre aux paramètres définis pour le rapport cible.

Aller au signet

Choisissez cette option pour définir un lien vers un signet du rapport actuel. Tapez ou sélectionnez l'identificateur de signet du rapport à atteindre lorsque l'utilisateur clique sur le lien. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. L'identificateur du signet peut être soit un identificateur statique, soit une expression qui prend la valeur d'un identificateur de signet. L'expression peut inclure un champ qui contient un identificateur de signet.

Aller à l'URL

Choisissez cette option pour définir un lien vers une page Web. Tapez ou sélectionnez l'URL d'une page Web, ou une expression qui prend la valeur de l'URL d'une page Web. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. Cette expression peut inclure un champ qui contient une URL.

3. Boîte de dialogue Propriétés du graphique - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés du graphique** pour définir les options générales du graphique. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés du graphique**.

Rubrique	Description
Propriétés du graphique - Onglet Effet 3D (Concepteur de rapports Visual Studio)	Décrit l'onglet Effet 3D de la boîte de dialogue Propriétés du graphique .
Propriétés du graphique - Onglet Données (Concepteur de rapports Visual Studio)	Décrit l'onglet Données de la boîte de dialogue Propriétés du graphique .
Propriétés du graphique - Onglet Filtres (Concepteur de rapports Visual Studio)	Décrit l'onglet Filtres de la boîte de dialogue Propriétés du graphique .
Propriétés du graphique - Onglet Légende (Concepteur de rapports Visual Studio)	Décrit l'onglet Légende de la boîte de dialogue Propriétés du graphique .
Propriétés du graphique - Onglet Axe des X (Concepteur de rapports Visual Studio)	Décrit l'onglet Axe des X de la boîte de dialogue Propriétés du graphique .
Propriétés du graphique - Onglet Axe des Y (Concepteur de rapports Visual Studio)	Décrit l'onglet Axe des Y de la boîte de dialogue Propriétés du graphique .

Options

Nom

Tapez le nom du graphique. Ce nom doit être unique dans le rapport.

Titre

Tapez le titre qui doit s'afficher dans la zone du graphique. Cliquez sur le bouton de style pour modifier le style du texte.

Palette

Sélectionnez une palette à utiliser pour le graphique.

Type de graphique

Sélectionnez un type de graphique à utiliser.

Sous-type de graphique

Sélectionnez un sous-type de graphique à utiliser. Les sous-types disponibles varient en fonction du type de graphique.

Style de zone de graphique

Choisissez cette option pour modifier les propriétés de style de la zone du graphique.

Style de la zone de traçage

Choisissez cette option pour modifier les propriétés de style de la zone de traçage du graphique.

4. Boîte dialogue Choisir le format (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Choisir le format** pour spécifier un format de zone de texte. Cette boîte de dialogue s'ouvre lorsque vous cliquez sur le bouton avec des points de suspension (...) dans l'onglet **Format** de la boîte de dialogue **Propriétés de la zone de texte**.

Options

Format

Spécifiez le format de données à utiliser pour les valeurs qui apparaissent dans la zone de texte. Les valeurs valides sont **Default**, **Number**, **Date**, **Time**, **Percentage** et **Currency**. Cliquez sur un format pour afficher les codes de format disponibles.

Personnalisé

Tapez un code de format personnalisé. Cliquez sur le bouton de fonction (fx) pour créer un code personnalisé à l'aide de l'éditeur d'expressions.

Exemple

Affiche un exemple de la valeur telle que mise en forme par le code spécifié.

5. Mode Design (Concepteur de rapports Visual Studio)

Le mode de conception graphique fournit une surface de dessin visuel que vous pouvez utiliser pour créer des fichiers de définition de rapport client (.rdlc) avec des contrôles ReportViewer. Le mode de conception graphique inclut une grille de conception que vous pouvez utiliser pour aligner les éléments de rapport sur la page. Lorsque vous sélectionnez la définition de rapport, un menu **Rapport** apparaît vous permettant d'accéder à des boîtes de dialogue de propriétés, de définir des images incorporées, d'ajouter des paramètres de rapport, etc.

Remarque

Si le menu **Rapport** n'est pas visible, cliquez dans la zone de conception du rapport.

6. Grille de conception du rapport

La grille de conception se compose de trois sections : le corps, l'en-tête de page et le pied de page. L'en-tête et le pied de page ne font pas partie du modèle de rapport. Pour ajouter un en-tête ou un pied de page, choisissez **En-tête de page** ou **Pied de page** dans le menu **Rapport**.

Pour ajouter des éléments à la grille de conception, utilisez la boîte à outils. La boîte à outils contient des régions de données et d'autres éléments de rapport pouvant être ajoutés au rapport. Une fois que vous avez ajouté une région de données ou une zone de texte au rapport, vous pouvez ouvrir la fenêtre Sources de données et faire glisser des champs dans des cellules spécifiques. Chaque élément présent dans la surface de dessin du rapport contient des propriétés qui peuvent être gérées à l'aide d'une boîte de dialogue de propriétés ou la fenêtre **Propriétés**.

7. Boîte à outils, fenêtre Sources de données et fenêtre Propriétés

La boîte à outils, la liste des champs et la fenêtre Propriétés permettent d'ajouter des éléments au rapport, et de les manipuler. Pour ce faire, dans le menu **Affichage**, cliquez sur **Boîte à outils** ou **Fenêtre Propriétés**. Pour afficher la fenêtre Sources de données, choisissez **Afficher les sources de données** dans le menu **Données**. Pour utiliser les boîtes de dialogue de propriétés, cliquez avec le bouton droit sur l'élément de rapport, puis sur **Propriétés**.

8. Boîte de dialogue Modifier une valeur du graphique - Onglet Valeurs (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Valeurs** de la boîte de dialogue **Modifier une valeur du graphique** pour définir les valeurs du graphique. Les options situées sous cet onglet varient selon le type de graphique. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Modifier une valeur du graphique**.

Rubrique	Description
Modifier une valeur du graphique - Onglet Action (Concepteur de rapports Visual Studio)	Décrit l'onglet Action de la boîte de dialogue Modifier une valeur du graphique .
Modifier une valeur du graphique - Onglet Apparence (Concepteur de rapports Visual Studio)	Décrit l'onglet Apparence de la boîte de dialogue Modifier une valeur du graphique .
Modifier une valeur du graphique - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Modifier une valeur du graphique .
Modifier une valeur du graphique - Onglet Étiquettes de points (Concepteur de rapports Visual Studio)	Décrit l'onglet Étiquettes de point de la boîte de dialogue Modifier une valeur du graphique .

Options

Étiquette de séries

Tapez une étiquette de séries, ou une expression qui prend la valeur d'une étiquette de séries. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Valeur

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux histogrammes, et aux graphiques à barres, en courbes, à secteurs et en anneau.

X

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques à bulles et aux graphiques à nuages de points (XY).

O

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques à bulles et aux graphiques à nuages de points (XY).

Taille

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques à bulles.

Haute

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques boursiers.

Basse

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques boursiers.

Ouvrir

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques boursiers.

Fermer

Tapez ou sélectionnez une valeur, ou une expression qui prend cette valeur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. S'applique uniquement aux graphiques boursiers.

9. Boîte dialogue Modifier les classes (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Modifier les classes** pour mettre à jour la propriété **Classes** pour le rapport. Vous pouvez ajouter des classes à partir d'un assembly personnalisé pour un rapport. N'ajoutez de classes que pour les membres d'instances. Ne spécifiez pas de membres statiques dans la liste **Classes**.

Vous pouvez ajouter des classes par le biais de la fenêtre Propriétés ou de l'onglet Propriétés du rapport - Onglet Références (Concepteur de rapports Visual Studio) de la boîte de dialogue Boîte de dialogue Propriétés du rapport - Onglet Général (Concepteur de rapports Visual Studio).

Options**Nom de classe**

Tapez le nom de la classe.

Nom de l'instance

Tapez le nom de l'instance de la classe.

10. Boîte dialogue Modifier les références (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Modifier les références** pour mettre à jour la propriété Références sur un rapport. Vous pouvez utiliser cette boîte de dialogue pour ajouter ou supprimer des références. Vous pouvez aussi utiliser l'onglet Propriétés du rapport - Onglet Références (Concepteur de rapports Visual Studio) de la boîte de dialogue Boîte de dialogue Propriétés du rapport - Onglet Général (Concepteur de rapports Visual Studio).

Options**Nom de l'assembly**

Cliquez sur le bouton Ajouter une référence (...) pour récupérer le nom d'assembly d'un assembly. Cliquez sur le bouton Supprimer (X) pour supprimer la référence d'assembly.

11. Boîte dialogue Images incorporées (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Images incorporées** pour incorporer des images dans un rapport. Vous pouvez créer des images incorporées pour les rendre disponibles dans la définition de rapport en tant que ressource interne. Ces images peuvent ensuite être utilisées dans un élément de rapport de type image ou en tant qu'images d'arrière-plan.

Options

Image

Affiche une représentation graphique de l'image. Pour ajouter une nouvelle image, cliquez sur la cellule située en fin de liste de la colonne **Image**, puis cliquez sur le bouton du Générateur d'images (...).

Nom

Affiche le nom utilisé pour faire référence à l'image dans le rapport. Ce nom est dérivé du nom de fichier utilisé lors de l'importation de l'image. Pour le modifier, cliquez sur celui-ci, puis tapez un nouveau nom.

Type MIME

Affiche le type MIME (Multipurpose Internet Mail Extensions) de l'image. Le type MIME est détecté lors de l'importation de l'image. Pour le modifier, cliquez sur celui-ci, puis sélectionnez un nouveau type MIME.

Nouvelle image

Cliquez sur **Nouvelle image** pour sélectionner un fichier image dans le système de fichiers et l'ajouter à la définition de rapport.

Supprimer

Choisissez cette option pour supprimer l'image sélectionnée du rapport.

12. Boîte dialogue Expression (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue Expression pour écrire une expression. La boîte de dialogue Expression inclut une fenêtre Code, une arborescence de catégorie, des éléments de catégorie et un volet Description. Vous pouvez ouvrir la boîte de dialogue Expression pour les éléments suivants :

- Une zone de texte sur un rapport.
- Une propriété dans la fenêtre Propriétés.

Vous pouvez ouvrir la boîte de dialogue Expression en cliquant avec le bouton droit sur une zone de texte ou en choisissant **<expression>** dans la liste de valeurs d'une propriété. Vous pouvez utiliser des expressions pour définir de nombreuses propriétés, notamment la couleur, la police et les bordures.

La boîte de dialogue Expression est contextuelle ; les éléments de catégories et les descriptions changent en fonction de la catégorie d'expression sur laquelle vous travaillez. Il prend en charge IntelliSense, le complément automatique des instructions et la coloration de la syntaxe afin de vous permettre de détecter facilement les erreurs de syntaxe. Vous pouvez déplacer et redimensionner la boîte de dialogue Expression si vous souhaitez disposer d'une plus grande aire de travail.

Constructions d'expression

Les expressions que vous créez peuvent inclure des constantes, des valeurs globales, des champs et d'autres éléments. La liste suivante décrit les catégories et les diverses parties d'une expression.

Constantes

Sélectionnez le nœud Constantes pour définir des expressions incluant une valeur constante. Les constantes sont utilisées pour spécifier la couleur, la police et des énumérateurs.

Certains éléments de rapport ne prennent pas en charge les expressions pouvant prendre la valeur d'une constante. Si une propriété ne peut pas prendre la valeur d'une constante, la description fournit cette information.

Globals

Fournit une liste des éléments dans la collection globale que vous pouvez utiliser dans une expression.

Paramètres

Fournit une liste de paramètres de rapport.

Champs

Affiche la liste des champs qui peuvent être utilisés dans l'expression. La liste varie selon la propriété. Double-cliquez sur un champ pour le copier dans la zone **Expression**. Vous pouvez également faire glisser le champ vers la zone **Expression**.

Datasets

Fournit une liste des datasets disponibles et montre les champs qui sont membres du dataset.

Opérateurs

Affiche les opérateurs que vous pouvez inclure dans un calcul ou une manipulation de chaînes.

Fonctions communes

Affiche les fonctions communes, regroupées par type (par exemple, texte, date et heure, etc.).

Options

Fenêtre Code

Utilisez la fenêtre Code dans le volet supérieur pour taper une expression. Lorsque vous ouvrez la boîte de dialogue Expression, la fenêtre Code contient l'expression courante. Vous pouvez remplacer ou réviser l'expression ou coller des fonctions, des opérateurs, des constantes, des champs, des paramètres et des éléments à partir des collections globales. La fenêtre Code reflète vos modifications.

Arbre de catégories

Affiche des catégories d'expressions. Le choix d'une catégorie établit un contexte pour la création d'une expression. Par exemple, la sélection de fonctions d'agrégation affiche **AVG**, **Count** et d'autres fonctions d'agrégation que vous pouvez incorporer dans l'expression.

Descriptions, exemples ou liste des membres

Selon l'élément de catégorie que vous sélectionnez, le troisième volet contient une description, un exemple d'expression ou une liste de membres.

13. Boîte de dialogue Filtres (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Filtres** pour définir les filtres de la région de données.

Options

Expression

Tapez ou sélectionnez l'expression à évaluer.

Opérateur

Sélectionnez l'opérateur à utiliser pour comparer l'expression et la valeur.

Valeur

Tapez l'expression ou la valeur à comparer au contenu de la zone **Expression**.

et/ou

Affiche la relation avec le filtre de la ligne suivante.

Haut

Choisissez cette option pour déplacer le filtre sélectionné vers le haut de la liste.

Bas

Choisissez cette option pour déplacer le filtre sélectionné vers le bas de la liste.

Supprimer

Choisissez cette option pour supprimer le filtre sélectionné.

14. Boîte de dialogue Propriétés de l'image - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés de l'image** pour définir les options générales de l'image. Pour ouvrir cette boîte de dialogue, cliquez avec le bouton droit sur un élément de rapport de type image dans un fichier de définition de rapport client (.rdlc), puis sélectionnez **Propriétés**. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés de l'image**.

Rubrique	Description
Propriétés de l'image - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés de l'image .
Propriétés de l'image - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés de l'image .

Options

Nom

Tapez le nom de l'image. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Répéter l'élément de rapport avec la région de données à chaque page

Choisissez cette option pour afficher l'image avec une région de données chaque fois que cette région s'affiche sur une page. Cela peut être utile pour les titres qui doivent s'afficher avec la région de données sur chaque page du rapport.

Région de données

Tapez ou sélectionnez la région de données associée à l'image qui doit être répétée sur chacune des pages où apparaît la région de données.

15. Boîte de dialogue Propriétés de la ligne - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés de la ligne** pour définir les options générales de la ligne. Pour ouvrir la boîte de dialogue **Propriétés de la ligne**, cliquez avec le bouton droit sur une ligne dans la mise en page du rapport puis sélectionnez **Propriétés**. Pour définir les propriétés de couleur, de style et de largeur de la ligne, vous devez utiliser la fenêtre Propriétés de Visual Studio. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés de la ligne**.

Rubrique	Description
Propriétés de la ligne - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés de la ligne .
Propriétés de la ligne - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés de la ligne .

Options

Nom

Tapez le nom de la ligne. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Répéter l'élément de rapport avec la région de données à chaque page

Choisissez cette option pour afficher la ligne avec une région de données chaque fois que cette dernière s'affiche sur une page. Cela peut être utile pour les titres qui doivent s'afficher avec la région de données sur chaque page du rapport.

Région de données

Tapez ou sélectionnez la région de données associée à la ligne qui doit être répétée sur chacune des pages où apparaît la région de données.

16. Boîte de dialogue Propriétés de la liste - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés de la liste** pour définir les options générales de la liste. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés de la liste**.

Rubrique	Description
Propriétés de la liste - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés de la liste .
Propriétés de la liste - Onglet Filtres (Concepteur de rapports Visual Studio)	Décrit l'onglet Filtres de la boîte de dialogue Propriétés de la liste .
Propriétés de la liste - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés de la liste .
Propriétés de la liste - Onglet Tri (Concepteur de rapports Visual Studio)	Décrit l'onglet Tri de la boîte de dialogue Propriétés de la liste .
Propriétés de la liste - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés de la liste .

Options

Nom

Tapez le nom de la liste. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Nom du dataset

Tapez ou sélectionnez la table de données ou l'objet d'entreprise à utiliser pour la liste.

Sauts de page

Sélectionnez les options indiquant le mode d'application des sauts de page.

Insérer un saut de page avant cette liste

Choisissez cette option pour placer un saut de page au début de chaque instance de la liste.

Insérer un saut de page après cette liste

Choisissez cette option pour placer un saut de page à la fin de chaque instance de la liste.

Faire tenir cette liste sur une page si possible

Choisissez cette option pour indiquer que l'ensemble de la liste doit tenir sur une seule page, dans la mesure du possible.

Modifier le groupe de détails

Choisissez cette option pour modifier les informations de regroupement des détails de la liste.

17. Boîte de dialogue Propriétés de la matrice - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés de la matrice** pour définir les options générales de la matrice. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés de la matrice**.

Rubrique	Description
Propriétés de la matrice - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés de la matrice .
Propriétés de la matrice - Onglet Filtres (Concepteur de rapports Visual Studio)	Décrit l'onglet Filtres de la boîte de dialogue Propriétés de la matrice .
Propriétés de la matrice - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés de la matrice .
Propriétés de la matrice - Onglet Groupes (Concepteur de rapports Visual Studio)	Décrit l'onglet Groupes de la boîte de dialogue Propriétés de la matrice .
Propriétés de la matrice - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés de la matrice .

Options

Nom

Tapez le nom de la matrice. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Nom du dataset

Tapez ou sélectionnez la table de données ou l'objet métier à utiliser pour la matrice.

Sauts de page

Sélectionnez les options indiquant le mode d'application des sauts de page.

Insérer un saut de page avant la matrice

Choisissez cette option pour placer un saut de page au début de chaque instance de la matrice.

Insérer un saut de page après la matrice

Choisissez cette option pour placer un saut de page à la fin de chaque instance de la matrice.

Faire tenir cette matrice sur une page si possible

Choisissez cette option pour indiquer que l'ensemble de la matrice doit tenir sur une seule page, dans la mesure du possible.

Extension des colonnes de la matrice

Sélectionnez une option pour indiquer le sens d'extension de la matrice.

De gauche à droite

Choisissez cette option pour étendre la matrice de gauche à droite.

De droite à gauche

Choisissez cette option pour étendre la matrice de droite à gauche.

Groupes avant les en-têtes de ligne

Sélectionnez le nombre de groupes (ensembles de colonnes) à afficher avant les colonnes d'en-têtes de ligne.

18. Boîte dialogue Paramètres (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Paramètres** pour définir les paramètres à passer au lien de sous-rapport ou d'extraction.

Options**Nom du paramètre**

Tapez le nom du paramètre dans le rapport cible.

Valeur du paramètre

Tapez ou sélectionnez une valeur pour le paramètre, ou une expression qui prend la valeur d'un paramètre.

19. Boîte de dialogue Propriétés du rectangle - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés du rectangle** pour définir les options générales du rectangle. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés du rectangle**.

Rubrique	Description
Propriétés du rectangle - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés du rectangle .
Propriétés du rectangle - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés du rectangle .
Propriétés du rectangle - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés du rectangle .

Options**Nom**

Tapez le nom du rectangle. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Répéter l'élément de rapport avec la région de données à chaque page

Choisissez cette option pour afficher le rectangle avec une région de données chaque fois que cette région s'affiche sur une page. Cela peut être utile pour le texte descriptif ou les en-têtes qui doivent s'afficher avec la région de données sur chaque page du rapport.

Région de données

Tapez ou sélectionnez la région de données associée au rectangle qui doit être répété sur chacune des pages où apparaît la région de données.

Sauts de page

Sélectionnez les options indiquant le mode d'application des sauts de page.

Insérer avant le rectangle

Choisissez cette option pour placer un saut de page au début du rectangle.

Insérer après le rectangle

Choisissez cette option pour placer un saut de page à la fin du rectangle.

20. Boîte de dialogue Sources de données du rapport (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Sources de données du rapport** pour ajouter et supprimer explicitement des informations de source de données de projet dans un fichier de définition de rapport. Si vous avez copié une définition de rapport à partir d'un autre projet, vous pouvez utiliser cette boîte de dialogue pour supprimer des sources de données qui n'existent pas dans le projet actuel.

Options

Sources de données du projet

Affiche la liste des sources de données disponibles dans le projet. Les sources de données valides comprennent des tables de données et des objets métier.

Les sources de données vides ou non définies n'apparaissent pas dans cette liste.

Ajouter au rapport

Cliquez sur **Ajouter au rapport** pour ajouter la source de données de projet que vous avez sélectionnée à la définition de rapport.

Sources de données du rapport

Affiche les sources de données spécifiées dans la définition de rapport.

Supprimer

Cliquez sur **Supprimer** pour supprimer les informations de schéma du fichier de définition de rapport.

Renommer

Cliquez sur **Renommer** pour mettre à jour le fichier de définition de rapport avec un dataset renommé.

Actualiser tout

Cliquez sur **Actualiser tout** pour synchroniser les informations de schéma de source de données de la définition de rapport (.rdlc) avec les informations de source de données du projet.

21. Boîte de dialogue Paramètres du rapport (Concepteur de rapports Visual Studio)

Utilisez la boîte de dialogue **Paramètres du rapport** pour définir les paramètres d'un rapport traité en mode local. Vous pouvez définir des paramètres de manière à prendre en charge la mise en forme conditionnelle ou pour les utiliser dans des expressions ou dans du code. Vous ne pouvez pas recourir à la boîte de dialogue **Paramètres du rapport** pour mapper des paramètres de rapport à des paramètres de requête ou pour les utiliser dans des filtres de source de données. En mode de traitement local, la totalité du traitement des données est gérée indépendamment du traitement du rapport. Si vous souhaitez transmettre des paramètres de rapport à un filtre de requête ou de source de données, vous devez

effectuer cette opération dans du code d'application. À titre d'exemple, consultez Exemple : Utilisation d'un flux RSS dans ReportViewer.

Une fois spécifiées dans la boîte de dialogue **Paramètres du rapport**, les propriétés de paramètres font partie de la définition de rapport. Certaines propriétés ne sont utilisées qu'à des fins de programmation. Contrairement aux rapports traités sur un serveur de rapports distant, un rapport traité localement ne possède pas de zone d'entrée de paramètre pour la sélection ou la saisie de valeurs de paramètre. Pour plus d'informations sur l'utilisation par programme des propriétés de paramètre, consultez [ReportParameterInfo](#) et [ReportParameterInfo](#).

Options

Paramètres

Utilisez cette liste pour visualiser et modifier les propriétés de paramètre de rapport dans la définition de rapport actuelle. Pour déplacer un paramètre dans la liste, cliquez sur celui-ci, puis sur la flèche vers le haut ou vers le bas.

Ajouter

Choisissez cette option pour créer un paramètre de rapport.

Supprimer

Cliquez sur un paramètre, puis choisissez cette option pour le supprimer.

Nom

Tapez le nom du paramètre. N'utilisez pas de caractères spéciaux dans le nom. Le nom doit être compatible avec la spécification CLS (Common Language Specification).

Invite

Tapez une chaîne de texte pour définir une valeur d'invite de paramètre. Cette valeur n'a aucun impact dans une définition de rapport client.

Type de données

Sélectionnez le type de données du paramètre.

Caché

Spécifiez cette option pour définir la propriété **PromptUser**. Cette valeur n'a aucun impact dans une définition de rapport client.

Interne

Spécifiez cette option pour définir la propriété **Visible**.

Valeurs multiples

Spécifiez cette option pour définir la propriété **Multivalued**.

Autoriser les valeurs Null

Spécifiez cette option si la valeur du paramètre peut être NULL.

Autoriser une valeur vide

Spécifiez cette option si la valeur du paramètre peut être une chaîne vide.

Valeurs disponibles

Spécifiez une liste statique de valeurs de paramètre de rapport. Vous pouvez taper une valeur ou écrire une expression qui fournit la valeur à utiliser. Les valeurs disponibles sont spécifiées en tant que paire nom-valeur.

Valeurs par défaut

Spécifiez la valeur par défaut à utiliser. Si vous configurez un paramètre à valeurs multiples, vous pouvez spécifier plusieurs valeurs par défaut. Sinon, seule la première valeur est utilisée.

NULL

Sélectionnez **Null** si vous ne souhaitez pas fournir de valeur par défaut au paramètre.

22. Boîte de dialogue Propriétés du rapport - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés du tableau** pour définir les options générales du rapport. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés du rapport**.

Rubrique	Description
Propriétés du rapport - Onglet Code (Concepteur de rapports Visual Studio)	Décrit l'onglet Code de la boîte de dialogue Propriétés du rapport .
Propriétés du rapport - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés du rapport .
Propriétés du rapport - Onglet Mise en page (Concepteur de rapports Visual Studio)	Décrit l'onglet Mise en page de la boîte de dialogue Propriétés du rapport .
Propriétés du rapport - Onglet Références (Concepteur de rapports Visual Studio)	Décrit l'onglet Références de la boîte de dialogue Propriétés du rapport .

Options

Auteur

Tapez le nom de l'auteur du rapport. Cette valeur est accessible par programme.

Description

Tapez une description du rapport. Cette valeur est accessible par programme. Si le rapport est ensuite converti et publié sur un serveur de rapports, la description est visible dans le Gestionnaire de rapports et dans d'autres outils qui affichent des informations de rapport.

Espacement de la grille

Tapez la valeur correspondant à l'espacement de la grille pour la surface de dessin. La grille est visible dans la section Corps, la section En-tête de page et la section Pied de page.

Dessiner une grille

Choisissez cette option pour afficher la grille sur la surface de dessin.

Aligner sur la grille

Choisissez cette option pour aligner les éléments du rapport sur la grille de la surface de dessin.

En-têtes de page

Sélectionnez les options indiquant le mode d'affichage des en-têtes de page dans le rapport. Cette option est accessible après la sélection de **En-tête de page** dans le menu **Rapport**.

Imprimer sur la première page

Choisissez cette option pour afficher l'en-tête de page sur la première page du rapport.

Imprimer sur la dernière page

Choisissez cette option pour afficher l'en-tête de page sur la dernière page du rapport.

Pieds de page

Sélectionnez les options indiquant le mode d'affichage des pieds de page dans le rapport. Cette option est accessible après la sélection de **Pied de page** dans le menu **Rapport**.

Imprimer sur la première page

Choisissez cette option pour afficher le pied de page sur la première page.

Imprimer sur la dernière page

Choisissez cette option pour afficher le pied de page sur la dernière page.

Actualisation automatique

Choisissez cette option pour actualiser la page à intervalles réguliers lors de son exécution. Tapez ou sélectionnez l'intervalle, en secondes.

23. Boîte de dialogue Propriétés de style - Onglet Bordure et ligne (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Bordure et ligne** de la boîte de dialogue **Propriétés de style** pour modifier les propriétés des bordures et des lignes.

Options

Style

Tapez ou sélectionnez un style de ligne ou une expression qui prend la valeur d'un style de ligne. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Largeur

Tapez ou sélectionnez une largeur de ligne, ou une expression qui prend la valeur d'une largeur de ligne. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Couleur

Tapez une couleur ou une expression qui prend la valeur d'une couleur. Cliquez sur le bouton de couleur pour choisir une couleur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

24. Boîte de dialogue Propriétés de style - Onglet Remplissage (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Remplissage** de la boîte de dialogue **Propriétés de style** pour modifier les propriétés de remplissage.

Options

Couleur

Tapez une couleur ou une expression qui prend la valeur d'une couleur. Cliquez sur le bouton de couleur pour choisir une couleur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Dégradé

Tapez ou sélectionnez un dégradé, ou une expression qui prend la valeur d'un dégradé. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Couleur de fin

Tapez une couleur ou une expression qui prend la valeur d'une couleur. Cliquez sur le bouton de couleur pour choisir une couleur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

25. Boîte de dialogue Propriétés de style - Onglet Police (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Police** de la boîte de dialogue **Propriétés de style** pour modifier les propriétés de police.

Options

Famille

Tapez ou sélectionnez une famille de polices ou une expression qui prend la valeur d'une famille de polices. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Taille

Tapez ou sélectionnez une taille de police ou une expression qui prend la valeur d'une taille de police. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Style

Tapez ou sélectionnez un style de police, ou une expression qui prend la valeur d'un style de police. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. Les styles valides sont **Normal** et **Italique**.

Épaisseur

Tapez ou sélectionnez une épaisseur de police ou une expression qui prend la valeur d'une épaisseur de police. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Couleur

Tapez une couleur ou une expression qui prend la valeur d'une couleur. Cliquez sur le bouton de couleur pour choisir une couleur. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Ornement

Tapez ou sélectionnez un ornement de police ou une expression qui prend la valeur d'un ornement de police. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. Les ornements de police valides sont **Aucun**, **Souligné**, **Ligne au-dessus** et **Barré**.

26. Boîte de dialogue Propriétés du sous-rapport - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés du sous-rapport** pour définir les options générales du sous-rapport. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés du sous-rapport**.

Rubrique	Description
Propriétés du sous-rapport - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés du sous-rapport .
Propriétés du sous-rapport - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés du sous-rapport .
Propriétés du sous-rapport - Onglet Paramètres (Concepteur de rapports Visual Studio)	Décrit l'onglet Paramètres de la boîte de dialogue Propriétés du sous-rapport .
Propriétés du sous-rapport - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés du sous-rapport .

Options

Nom

Tapez le nom du sous-rapport. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Sous-rapport

Tapez ou sélectionnez le nom du sous-rapport à inclure dans le rapport. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

27. Boîte de dialogue Propriétés du tableau - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés du tableau** pour définir les options générales du tableau. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés du tableau**.

Rubrique	Description
Propriétés du tableau - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Décrit l'onglet Sortie des données de la boîte de dialogue Propriétés du tableau .
Propriétés du tableau - Onglet Filtres (Concepteur de rapports Visual Studio)	Décrit l'onglet Filtres de la boîte de dialogue Propriétés du tableau .
Propriétés du tableau - Onglet Groupes (Concepteur de rapports Visual Studio)	Décrit l'onglet Groupes de la boîte de dialogue Propriétés du tableau .
Propriétés du tableau - Onglet Navigation (Concepteur de rapports Visual Studio)	Décrit l'onglet Navigation de la boîte de dialogue Propriétés du tableau .
Propriétés du tableau - Onglet Tri (Concepteur de rapports Visual Studio)	Décrit l'onglet Tri de la boîte de dialogue Propriétés du tableau .
Propriétés du tableau - Onglet Visibilité (Concepteur de rapports Visual Studio)	Décrit l'onglet Visibilité de la boîte de dialogue Propriétés du tableau .

Options

Nom

Tapez le nom du tableau. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans **Info-bulle** s'affiche lorsque l'utilisateur maintient le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Nom du dataset

Tapez ou sélectionnez le nom du dataset à utiliser pour le tableau.

Sauts de page

Sélectionnez les options indiquant le mode d'application des sauts de page.

Insérer un saut de page avant ce tableau

Choisissez cette option pour placer un saut de page au début de chaque instance du tableau.

Insérer un saut de page après ce tableau

Choisissez cette option pour placer un saut de page à la fin de chaque instance du tableau.

Faire tenir le tableau sur une page si possible

Choisissez cette option pour indiquer que l'ensemble du tableau doit tenir sur une seule page, dans la mesure du possible.

En-tête/Pied de page

Sélectionnez les options permettant d'afficher les lignes d'en-tête et de pied de page du tableau sur chaque page.

Répéter les lignes d'en-tête sur chaque page

Choisissez cette option pour afficher les lignes d'en-tête de tableau sur chacune des pages où apparaît le tableau.

Répéter les lignes de pied de page sur chaque page

Choisissez cette option pour afficher les lignes de pied de page de tableau sur chacune des pages où apparaît le tableau.

28. Boîte de dialogue Propriétés de la zone de texte - Onglet Général (Concepteur de rapports Visual Studio)

Utilisez l'onglet **Général** de la boîte de dialogue **Propriétés de zone de texte** pour définir les options générales de la zone de texte. Le tableau suivant décrit les autres onglets de la boîte de dialogue **Propriétés de la zone de texte**.

Rubrique	Description
Propriétés de la zone de texte - Onglet Sortie des données (Concepteur de rapports Visual Studio)	Utilisez cette page pour spécifier les options de la sortie de données XML (celle-ci n'est toutefois pas prise en charge pour les fichiers de définition de rapport client).
Propriétés de la zone de texte - Onglet Police (Concepteur de rapports Visual Studio)	Utilisez cette page pour spécifier les propriétés de police de la chaîne qui apparaît dans la zone de texte.
Propriétés de la zone de texte - Onglet Format (Concepteur de rapports Visual Studio)	Utilisez cette page pour définir les propriétés de mise en forme des données dans la zone de texte.
Propriétés de la zone de texte - Onglet Navigation (Concepteur de rapports Visual Studio)	Utilisez cette page pour définir les actions de liaison associées à la zone de texte.
Propriétés de la zone de texte - Tri interactif (Concepteur de rapports Visual Studio)	Utilisez cette page afin de configurer une colonne pour le tri interactif de manière à ce qu'un utilisateur puisse cliquer sur l'en-tête de la colonne pour trier les données en fonction des valeurs de celle-ci.
Propriétés de la zone de texte - Onglet Visibilité (Concepteur de rapports Visual Studio)	Utilisez cette page pour spécifier si la zone de texte est visible. Vous pouvez également définir des propriétés qui configurent la zone de texte en tant qu'élément de bascule afin d'afficher ou de masquer telle ou telle zone du rapport.

Options**Nom**

Tapez le nom de la zone de texte. Ce nom doit être unique dans le rapport.

Info-bulle

Tapez une info-bulle ou une expression qui prend la valeur d'une info-bulle. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière. La valeur contenue dans Info-bulle s'affiche lorsque l'utilisateur positionne le pointeur de la souris au-dessus de la zone de texte d'un rapport HTML.

Répéter l'élément de rapport avec la région de données à chaque page

Choisissez cette option pour afficher la zone de texte avec une région de données chaque fois que cette région s'affiche sur une page. Cela peut être utile pour le texte descriptif ou les titres qui doivent s'afficher avec la région de données sur chaque page du rapport.

Région de données

Tapez ou sélectionnez la région de données associée à la zone de texte qui doit être répétée sur chacune des pages où apparaît la région de données.

Valeur

Tapez la valeur de la zone de texte. Il peut s'agir d'une expression de champ, d'un autre type d'expression ou d'une étiquette. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

Masquer les doublons

Choisissez cette option pour afficher uniquement la première instance d'une valeur lorsqu'une zone de texte se répète dans une région de données. Par exemple, si une table de données renvoie le même nom de produit pour plusieurs lignes, seule la première instance du nom de produit s'affiche.

Contenant un groupe ou un dataset

Tapez un groupe ou un dataset, ou une expression qui prend la valeur d'un groupe ou d'une table de données, où les doublons doivent être masqués. Cliquez sur le bouton d'expression (fx) pour modifier cette dernière.

29. Ajout d'interactivité, de visibilité et de navigation à un rapport (Concepteur de rapports Visual Studio)

Si vous utilisez le contrôle de serveur Web ReportViewer, vous pouvez créer des rapports incluant des fonctions interactives. Les fonctions interactives incluent le tri défini par l'utilisateur, des liens, des signets, des plans du document et des éléments de bascule que vous ajoutez pour afficher ou masquer certaines parties d'un rapport. Les fonctions interactives sont utilisées dans des rapports HTML. Les plans du document et les parties à bascule du rapport sont des fonctions basées sur un script qui fonctionnent uniquement si la configuration du navigateur prévoit l'exécution d'un script. Pour plus d'informations sur la prise en charge du navigateur, consultez *Prise en charge d'un navigateur pour les contrôles de serveur Web ReportViewer*. Pour plus d'informations sur le tri interactif, consultez *Tri de données dans un rapport (Concepteur de rapports Visual Studio)*.

Liens

Vous pouvez ajouter plusieurs types de liens à un rapport. Lorsque vous ajoutez un lien à une zone de texte, à une image ou à un élément de graphique, vous pouvez spécifier l'un des trois types de liens suivants :

Élément	Description
URL	<p>Une URL fournit un lien vers une page Web, située généralement en dehors du serveur de rapports. Un lien hypertexte peut être une URL statique ou une expression qui est évaluée en une URL. Si une base de données comprend un champ avec des URL, vous pouvez insérer ce champ dans l'expression de façon à produire une liste dynamique de liens hypertexte dans le rapport. Vous pouvez ajouter des liens hypertexte aux zones de texte et aux images uniquement.</p> <p>Pour créer un lien hypertexte, cliquez avec le bouton droit sur la zone de texte ou l'image à laquelle ajouter un lien, puis cliquez sur Propriétés. Sous l'onglet Navigation, sélectionnez Aller à l'URL. Tapez ou sélectionnez une URL ou une expression qui</p>

	correspond à une URL.
Signet	Un lien de signet, qui fournit un lien vers un signet ou un ancrage dans le rapport en cours. Pour créer un signet, définissez un signet sur l'élément de rapport de destination et ajoutez des liens de signet sur les éléments de rapport, tels qu'un mot ou un bouton, sur lesquels les utilisateurs doivent cliquer pour aller à l'élément du rapport associé à un signet. Vous pouvez créer des signets sur n'importe quel élément de rapport, mais seule une zone de texte ou une image peut comporter un lien de signet. Pour plus d'informations, consultez Procédure : créer un signet (Concepteur de rapports Visual Studio).
Extraction	Un lien de rapport d'extraction fournit un lien vers un autre rapport stocké sur le serveur de rapports et peut passer des valeurs de paramètres à ce rapport. Il contient en général des détails sur un élément figurant dans le rapport de synthèse d'origine. Prenons l'exemple d'un rapport de synthèse sur les ventes, contenant une liste des totaux des ventes et des commandes. Lorsque l'utilisateur clique sur un numéro de commande dans la liste de synthèse, un autre rapport s'affiche pour présenter des informations sur la commande en question. Pour plus d'informations sur les rapports d'extraction, consultez Configuration de sous-rapports et de rapports d'extraction (Concepteur de rapports Visual Studio).

III. Masquer des éléments

Chaque élément dans un rapport est associé à un ensemble de propriétés qui déterminent si l'élément est visible ou masqué. Vous pouvez utiliser ces propriétés pour masquer les éléments dans un rapport, masquer de façon conditionnelle des données en fonction d'autres données présentes dans le rapport, et fournir un élément sur lequel l'utilisateur peut cliquer pour afficher ou masquer les éléments. Par exemple, vous pouvez créer un rapport d'extraction qui affiche des données de synthèse lors du premier chargement du rapport et qui affiche des lignes de détails lorsque les utilisateurs cliquent sur une zone de texte en particulier.

Le masquage d'éléments vise principalement à créer un rapport qui affiche des données de synthèse tout en laissant à l'utilisateur la possibilité de descendre dans la hiérarchie pour faire apparaître des données détaillées. Pour créer cet effet d'exploration vers le bas, sélectionnez le groupe, la colonne ou la ligne à masquer, spécifiez **True** pour son état masqué et attribuez à l'élément qui masque et affiche tour à tour les données le nom d'une zone de texte dans un groupe conteneur. Au moment de l'exécution, l'utilisateur peut cliquer sur la zone de texte pour développer et réduire les données détaillées. Pour plus d'informations, consultez Procédure : masquer ou définir Activer/désactiver la visibilité sur un élément de rapport (Concepteur de rapports Visual Studio).

Remarque

Lorsque vous créez un rapport d'extraction vers le bas, les informations de visibilité doivent être définies sur le groupe, la colonne ou la ligne à masquer, et non sur une simple zone de texte dans la ligne ou la colonne. Si vous spécifiez ces options sur la zone de texte uniquement, les lignes ou les colonnes ne pourront pas être réduites.

IV. Plan du rapport

Les utilisateurs disposent d'une autre méthode pour interagir avec un rapport : ils peuvent utiliser le plan du document. Dans le contrôle ReportViewer, un plan du document apparaît sous la forme d'une table des matières placée à côté du rapport. Les sections et les groupes de rapport sont organisés en une hiérarchie de liens. Lorsque vous cliquez sur un de ses éléments, le rapport est actualisé de façon à afficher la zone du rapport correspondant à l'élément sélectionné.

Le plan du document est destiné à une utilisation dans des rapports HTML. Les autres formats d'exportation ont différentes manières d'articuler un plan du document :

- PDF affiche un plan du document sous la forme du volet Signets. Tous les éléments du plan du document sont répertoriés sur ce volet. La liste de liens n'est pas hiérarchisée.
- Excel montre un plan du document sous la forme d'une feuille de calcul incluant une hiérarchie de liens. Les sections de rapport sont rendues dans des feuilles de calcul séparées qui sont incluses avec le plan du document dans le même classeur.

Pour créer un plan du document, ajoutez des étiquettes Plan du document aux éléments du rapport devant apparaître dans le plan, par exemple les en-têtes et les graphiques. Si un élément de rapport dispose d'une étiquette, un plan du document est automatiquement généré quand un utilisateur affiche le rapport. Procédure : créer un plan du document (Concepteur de rapports Visual Studio).

V. Ajout de graphiques à un rapport (Concepteur de rapports Visual Studio)

Un graphique est une représentation graphique des données d'un rapport. L'affichage de données sous la forme d'un graphique est visuellement attrayante et permet aux utilisateurs d'afficher des comparaisons, des modèles et des tendances pour les données. Par exemple, au lieu de devoir analyser plusieurs colonnes de données dans un rapport tabulaire ou matriciel, vous pouvez voir d'un coup d'œil l'évolution des ventes au cours des trois derniers trimestres, ou la comparaison des chiffres de ventes des trois dernières années. Vous disposez d'une large sélection de types de graphiques et vous pouvez changer le style d'un graphique en incluant différentes couleurs, différents symboles ainsi que des effets tridimensionnels (3D).

Remarque

Microsoft propose sous licence le contrôle graphique de Dundas Software utilisé dans les rapports. Pour plus d'informations sur les fonctionnalités de graphique proposées par Dundas Software, visitez le site Web de la société à l'adresse www.dundas.com.

a. Comment ajouter, déplacer ou supprimer un graphique

Pour travailler avec un graphique, ouvrez un fichier de définition de rapport client (.rdlc) en mode de conception graphique. Pour ajouter un graphique, faites glisser l'élément de rapport Graphique de la boîte à outils à la mise en page du rapport. Une fois que le graphique est sur le rapport, vous pouvez sélectionner le type de graphique et autres attributs en définissant les propriétés du graphique. Vous pouvez dimensionner et positionner un graphique en pointant sur la bordure ombrée et en la faisant glisser vers un autre emplacement. Vous pouvez ajouter des données en faisant glisser des champs dans des zones de dépôt, ou utiliser la boîte de dialogue à onglets Boîte de dialogue Modifier une valeur du graphique - Onglet Valeurs (Concepteur de rapports Visual Studio) pour définir ou modifier les données d'instance du graphique. Pour supprimer un graphique, cliquez avec le bouton droit sur la bordure ombrée du graphique, puis cliquez sur **Supprimer**. Vous pouvez aussi appuyer sur la touche SUPPR.

b. Modification de l'apparence du graphique

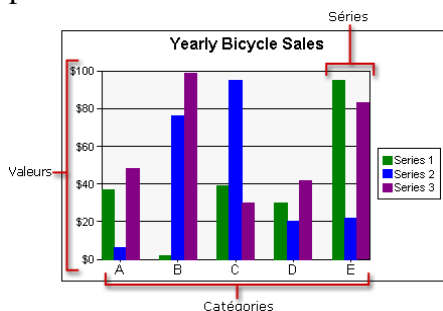
Dans un graphique, vous pouvez modifier l'apparence des zones de traçage et de graphique, des axes des ordonnées et des abscisses, ainsi que de la légende. Vous pouvez également appliquer un effet tridimensionnel. Le dégradé du remplissage de la zone de traçage ne s'affiche pas lorsque l'effet tridimensionnel est appliqué. Si vous utilisez une police dont la taille est supérieure à 10 points dans la légende du graphique, certaines étiquettes de légende risquent d'être tronquées. Cela peut n'être perceptible qu'avec des tailles de polices plus grandes.

Pour modifier un graphique, définissez des propriétés dans la boîte de dialogue à onglets Boîte de dialogue Propriétés du graphique - Onglet Général (Concepteur de rapports Visual Studio). Pour plus

d'informations sur la manière de mettre en forme un graphique, consultez Procédure : attribuer un style à un élément de graphique (Concepteur de rapports Visual Studio).

c. Données de graphique

Les données des graphiques dans les fichiers de définition de rapport sont organisées selon trois axes : Valeurs, catégories et séries. Les graphiques ont généralement deux axes servant à catégoriser et mesurer les données : un axe de catégorie (x) et un axe de valeur (y). L'axe des Y est généralement l'axe vertical et contient les données de l'onglet Valeur. L'axe des X est généralement l'axe horizontal et contient les catégories de l'onglet Catégorie. Les points du graphique résultant contiennent les séries de données de l'axe Série. L'axe Série est qualifié d'axe (z). Les graphiques à secteurs ou en anneau n'ont pas d'axes. Pour afficher les zones d'axes, cliquez d'abord sur le graphique.



Lorsque vous cliquez sur le graphique, trois zones de dépôt apparaissent, une pour chaque zone. Vous pouvez faire glisser des champs de la liste Champs sur chacune de ces zones de dépôt.

Pour ajouter des données à la disposition du graphique, faites glisser les champs de la fenêtre Sources de données sur une zone de dépôt du graphique. Pour mieux comprendre l'interaction entre ces différentes zones, vous pourriez comparer une disposition de graphique à une disposition de matrice. Les valeurs dans un groupe de valeurs équivalent à des lignes ou des colonnes statiques dans une disposition de matrice. Les champs dans un groupe de catégories équivalent à des groupes de colonnes dynamiques dans une disposition de matrice. Les champs dans un groupe de séries équivalent à des lignes dynamiques dans une disposition de matrice.

d. Valeurs

Le groupe de valeurs détermine la taille de l'élément de graphique pour chaque groupe de catégories. Par exemple, un champ Valeur détermine la hauteur d'une colonne dans un histogramme et la taille d'un secteur dans un graphique à secteurs.

Lorsque vous concevez un rapport de type graphique, vous devez ajouter au moins un champ au groupe de valeurs. Les groupes de valeurs sont statiques dans les graphiques. Si vous ajoutez uniquement un champ au groupe de valeurs et si vous n'ajoutez pas de champ au groupe de séries, un seul élément de graphique est ajouté. Si vous ajoutez plusieurs champs au groupe de valeurs, un élément de graphique est affiché pour chaque valeur. En présence de plusieurs champs dans le groupe de valeurs, la légende du graphique affiche le nom de chaque valeur.

Les données sont généralement groupées par catégorie dans un graphique. Si vous ne groupez pas vos données en fonction des catégories ou des séries, vous devez employer une expression d'agrégation pour les expressions de valeurs dans le graphique. Si vous ne groupez pas les données parce que vous n'avez qu'un champ dans le groupe de catégories, vous n'avez pas besoin d'utiliser un champ d'agrégation dans le groupe de valeurs.

e. Catégories

Utilisez des catégories pour regrouper des données. Les catégories fournissent les étiquettes des éléments de graphique. Par exemple, dans un histogramme, les étiquettes de catégories sont placées sur l'axe des abscisses (X), avec une étiquette pour chaque ensemble de colonnes.

Vous pouvez imbriquer des catégories. Si vous en définissez plusieurs, chaque catégorie est imbriquée dans une autre. Dans un histogramme qui affiche les produits par modèle, par exemple, le premier groupe de catégories serait le modèle et le second, le produit. L'histogramme indiquerait des regroupements de produits par modèle sur l'axe des abscisses.

f. Groupes de séries

Le groupe de séries sert à ajouter une dimension supplémentaire aux données dans le rapport. Par exemple, dans un histogramme représentant les ventes par produit, par exemple, vous pouvez ajouter un groupe de séries pour afficher les données par année pour chaque produit. Il n'est pas obligatoire d'ajouter un champ dans le groupe de séries ; la conception d'un graphique est facultative.

Si vous utilisez un groupe de séries, le nom du groupe est placé dans la légende du graphique. Les groupes de séries sont dynamiques.

g. Choix d'un type de graphique

Vous pouvez choisir l'un de ces types de graphiques : histogramme, graphiques à barres, en courbes, à secteurs, à nuages de points, en bulles, en aires, en anneau et boursier. Le tableau suivant décrit chaque type et fournit un lien à une rubrique qui contient plus d'informations.

Rubrique	Description
Type d'histogramme (Concepteur de rapports Visual Studio)	Décrit les histogrammes. Les histogrammes affichent les données sous forme d'ensembles de colonnes verticales. Inclut des informations sur les graphiques hybrides colonnes/lignes.
Type de graphique à barres (Concepteur de rapports Visual Studio)	Décrit les graphiques à barres. Les graphiques à barres affichent les données sous la forme de barres horizontales.
Type de graphique en courbes (Concepteur de rapports Visual Studio)	Décrit les graphiques en courbes. Les graphiques en courbes affichent les données sous la forme d'un ensemble de points reliés par une courbe.
Type de graphique à secteurs (Concepteur de rapports Visual Studio)	Décrit les graphiques à secteurs. Les graphiques à secteurs affichent les données sous la forme de pourcentages de la totalité.
Type de graphique Nuages de points (XY) (Concepteur de rapports Visual Studio)	Décrit les graphiques à nuages de points. Les graphiques à nuages de points affichent les données sous la forme d'un ensemble de points dans l'espace.
Type de graphique à bulles (Concepteur de rapports Visual Studio)	Décrit les graphiques en bulles. Les graphiques en bulles affichent un ensemble de symboles dont la position et la taille sont basées sur les données dans le graphique.
Type de graphique en aires (Concepteur de rapports Visual Studio)	Décrit les graphiques en aires. Les graphiques en aires affichent les données sous la forme d'un ensemble de points reliés par une courbe, la zone sous la courbe étant remplie.

Type de graphique en anneau (Concepteur de rapports Visual Studio)	Décrit les graphiques en anneau. Les graphiques en anneau affichent les données sous la forme de pourcentages par rapport à un total.
Type de graphique boursier (Concepteur de rapports Visual Studio)	Décrit les graphiques boursiers. Les graphiques boursiers affichent les données sous la forme d'un ensemble de courbes avec marques de données affichées pour les valeurs maximale, minimale, d'ouverture et de clôture.

VI. Utilisation d'expressions dans un rapport (Concepteur de rapports Visual Studio)

Vous pouvez inclure des expressions Microsoft Visual Basic dans des rapports ReportViewer. Des expressions s'utilisent pour calculer la valeur d'un élément de rapport ou les valeurs de propriétés d'un élément de rapport (style, mise en forme, etc.). Vous pouvez utiliser toutes les fonctions Visual Basic ainsi que des fonctions intégrées prises en charge uniquement dans des fichiers de définition de rapport.

Pour plus d'informations sur les expressions couramment utilisées dans des rapports ReportViewer, consultez Expressions communes pour les rapports (Concepteur de rapports Visual Studio).

a. Expressions de champ

Le type d'expression le plus élémentaire est celui qui affiche une valeur de champ dans une zone de texte. C'est ce que l'on nomme expression de champ. Pour lier un champ de données à un élément de rapport, l'expression doit inclure la collection **Fields**, le nom du champ et la propriété **Value**. La valeur est créée automatiquement lorsque vous faites glisser un champ dans le rapport. Dans l'exemple suivant, une expression affiche un nom de produit dans une zone de texte :

```
=Fields!Product.Value
```

Une expression peut être une expression courte faisant référence à un objet de champ ou une expression longue prenant en charge des fonctions de décision ou l'application d'une mise en forme tribunaire de champs ou d'autres éléments de rapport. Les expressions dans les éléments de rapport et les propriétés doivent commencer par un signe égal (=). Si vous ne commencez pas un texte avec ce caractère, le texte prendra la valeur réelle du champ.

Voici quelques exemples d'expressions de champ :

- Cette expression concatène les champs FirstName et LastName.

```
=Fields!FirstName.Value & " " & Fields!LastName.Value
```

- L'expression suivante effectue une agrégation de type somme sur le champ LineTotal.

```
=Sum(Fields!LineTotal.Value)
```

b. Mise en forme conditionnelle

Vous pouvez utiliser des expressions pour contrôler l'apparence d'un élément de rapport. Par exemple, vous pouvez écrire une expression pour la propriété **Color** d'une zone de texte de sorte que les données soient affichées dans une couleur différente en fonction d'une condition. Voici un exemple de mise en forme conditionnelle :

- Cette expression, utilisée dans la propriété **Color** d'une zone de texte, affiche la valeur du champ Cost en rouge si elle est supérieure à celle du champ Revenue. Si cette condition n'est pas satisfaite, le texte s'affiche en noir.

```
=IIf(Fields!Cost.Value > Fields!Revenue.Value, "Red", "Black")
```

Pour plus d'informations sur la mise en forme conditionnelle, consultez Ajout d'un style et d'une mise en forme à un rapport (Concepteur de rapports Visual Studio).

VII. Ajout de code personnalisé à un rapport (Concepteur de rapports Visual Studio)

Vous pouvez ajouter des fonctions personnalisées à un rapport et ajouter des références à des fonctions dans des assemblies externes. Lors de la compilation du projet, les assemblies suivants sont référencés automatiquement : `Microsoft.VisualBasic`, `AllMembers.T:System.Convert` et `AllMembers.T:System.Math`. D'autres assemblies peuvent être référencés à l'aide de la boîte de dialogue **Propriétés du rapport** ou de l'élément **CodeModules** dans la définition de rapport. La boîte de dialogue **Propriétés du rapport** peut également être utilisée pour définir des fonctions personnalisées. Pour définir des fonctions personnalisées dans la définition de rapport, utilisez l'élément **Code**. Notez qu'il est impossible de passer des ensembles de valeurs de données à des fonctions (les agrégations personnalisées ne sont notamment pas prises en charge).

Vous pouvez écrire votre propre code sur mesure afin de l'utiliser dans les expressions d'un rapport. Pour cela, deux solutions s'offrent à vous : vous pouvez incorporer le code dans un rapport ou faire référence à des méthodes présentes dans un assembly personnalisé. L'incorporation de code convient dans le cas de fonctions complexes ou de fonctions utilisées plusieurs fois dans un même rapport. Utilisez des assemblies personnalisés si vous voulez centraliser du code en un emplacement unique et le partager entre plusieurs rapports.

a. Code incorporé

Pour utiliser du code dans un rapport, vous devez ajouter un bloc de code dans le rapport. Ce bloc de code peut contenir plusieurs méthodes. Les méthodes du code incorporé doivent être écrites en Microsoft Visual Basic et doivent être basées sur des instances.

Pour ajouter du code à un rapport

- a. Dans le menu **Rapport**, cliquez sur **Paramètres du rapport**.

Remarque

Si le menu **Rapport** n'est pas visible, cliquez dans la zone de conception du rapport.

- a. Sous l'onglet **Code**, tapez le code voulu dans **Code personnalisé**.

Les méthodes du code incorporé sont disponibles par l'intermédiaire d'un membre **Code** globalement défini. Pour y accéder, il faut faire référence au membre **Code** et au nom de la méthode voulue. L'exemple de code ci-dessous appelle la méthode **ToUSD**, qui convertit la valeur du champ **StandardCost** en une valeur exprimée en dollars :

```
=Code.ToUSD(Fields!StandardCost.Value)
```

b. Assemblies personnalisés

Pour utiliser des assemblies personnalisés dans un rapport, vous devez d'abord créer l'assembly, le rendre accessible au projet, ajouter une référence à l'assembly dans le rapport, puis utiliser une expression qui fait référence aux méthodes de cet assembly. Lorsque le rapport est déployé sur le serveur de rapports, vous devez également déployer l'assembly personnalisé sur le serveur de rapports.

Pour ajouter une référence d'assembly à un rapport

- a. Dans le menu **Rapport**, cliquez sur **Paramètres du rapport**.

Remarque

Si le menu **Rapport** n'est pas visible, cliquez dans la zone de conception du rapport.

- b. Dans l'onglet **Références**, procédez comme suit :

- Dans **Références**, cliquez sur le bouton Ajouter (...) et sélectionnez l'assembly ou recherchez-le dans la boîte de dialogue **Ajouter une référence**.
- Dans **Classes**, tapez le nom de la classe et fournissez le nom de l'instance à utiliser dans le rapport.

Remarque

Spécifiez une classe et un nom d'instance uniquement pour les membres basés sur une instance. Ne spécifiez pas de membres statiques dans la liste **Classes**.

Pour faire référence à du code personnalisé dans une expression, vous devez appeler le membre d'une classe au sein de l'assembly. La procédure pour ce faire dépend du type de méthode, à savoir statique ou basée sur une instance. Les méthodes statiques d'un assembly personnalisé sont disponibles de façon globale dans le rapport. Elles sont accessibles dans les expressions au moyen de l'espace de noms, de la classe et du nom de la méthode. L'exemple de code ci-dessous appelle la méthode **ToGBP**, qui convertit la valeur du champ StandardCost en une valeur exprimée en livres sterling :

```
=CurrencyConversion.DollarCurrencyConversion.ToGBP(Fields!StandardCost.Value)
```

Les méthodes basées sur des instances sont disponibles par l'intermédiaire d'un membre **Code** globalement défini. Pour y accéder, il faut faire référence au membre **Code** et ensuite au nom de l'instance et de la méthode. L'exemple de code ci-dessous appelle la méthode **ToEUR**, qui convertit la valeur du champ StandardCost en une valeur exprimée en euros :

```
=Code.m_myDollarConversion.ToEUR(Fields!StandardCost.Value)
```

Remarque

Dans le Générateur de rapports, un assembly personnalisé est chargé une fois et il n'est pas déchargé tant que vous ne fermez pas Visual Studio. Si vous affichez un aperçu d'un rapport, modifiez un assembly personnalisé utilisé dans ce rapport, puis affichez à nouveau l'aperçu, vos modifications n'apparaissent pas dans ce deuxième aperçu. Pour recharger l'assembly, fermez et rouvrez Visual Studio, puis affichez l'aperçu du rapport.

Pour en savoir plus sur les assemblies personnalisés, vous pouvez rechercher « Utilisation d'assemblies personnalisés avec des rapports » dans la documentation en ligne de SQL Server 2005. La documentation en ligne est disponible dans la plupart des éditions de Visual Studio 2008. Vous pouvez également la consulter sur MSDN à l'adresse www.microsoft.com. Les informations sur les assemblies personnalisés dans la documentation en ligne s'appliquent aux rapports que vous déployez avec les contrôles ReportViewer.

VIII. Ajout d'images à un rapport (Concepteur de rapports Visual Studio)

Une image est un élément de rapport qui contient une référence à une image qui est stockée sur le serveur de rapports, incorporée dans le rapport ou enregistrée dans une base de données. Une image peut être un logo ou une image apparaissant une fois sur le rapport, ou encore une image répétée avec des lignes de données. Vous pouvez également utiliser une image en guise d'arrière-plan pour certains éléments de rapport. Vous pouvez obtenir des images des sources suivantes :

- Base de données
- Partage de fichiers externe ou site Web
- Incorporées localement dans le rapport

Vous ne pouvez pas utiliser des fichiers image qui ont été ajoutés à un projet.

Des images liées aux données peuvent également être affichées à partir de données binaires stockées dans une base de données. Par exemple, les images qui jouxtent les noms de produits dans une liste de produits sont des images de base de données.

Les images externes sont spécifiées sous la forme d'une URL pointant vers un fichier image. Les images externes sont parfaitement adaptées aux logos et images statiques partagées par plusieurs rapports ou pages Web.

Les images de rapport locales sont incorporées dans le rapport, puis référencées. Les données de type image sont stockées dans la définition de rapport et n'existent pas en tant que fichier séparé. En revanche, il est préférable de recourir à des images incorporées pour vous assurer qu'elles sont disponibles en permanence pour le rapport. Cependant, elles ne peuvent pas être partagées. L'incorporation d'une image augmente la taille du fichier de définition de rapport. Lorsque vous incorporez une image, le Concepteur de rapports lui applique un codage MIME, puis la stocke en tant que texte dans la définition de rapport.

1. Comment ajouter une image incorporée à un rapport

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- Dans le menu Rapport, sélectionnez **Images incorporées**, puis ajoutez toutes les images que vous souhaitez utiliser dans le rapport. Si le menu Rapport n'est pas visible, cliquez dans la zone de conception du rapport.
- Faites glisser le contrôle image de la boîte à outils à la mise en page du rapport. Vous devez utiliser le contrôle image employé pour les rapports ReportViewer.
- Sélectionnez l'image, puis ouvrez la fenêtre Propriétés dans Visual Studio.
- Attribuez à **Source** la valeur **Embedded**.
- Pour **Valeur** choisissez une image qui est incorporée dans la définition de rapport. Vous pouvez effectuer ce choix dans une liste déroulante.

2. Images d'arrière-plan

Vous pouvez utiliser une image comme arrière-plan dans le corps du rapport ou dans un rectangle, une zone de texte, une liste, une matrice ou une table. Une image d'arrière-plan possède les mêmes propriétés qu'une image. Vous pouvez également indiquer la manière dont l'image est reproduite pour occuper tout l'arrière-plan de l'élément.

Remarque

Si vous définissez une image d'arrière-plan pour le corps du rapport, mais pas pour l'en-tête ou le pied de page, les rapports HTML utiliseront automatiquement la même image d'arrière-plan dans les trois régions du rapport. Si vous ne voulez pas utiliser d'image d'arrière-plan dans l'en-tête ou le pied de page, vous devez explicitement définir une image différente pour ces régions. Si vous ne voulez pas que ces régions comportent une image, vous pouvez créer et affecter une image transparente de petite taille à l'en-tête ou au pied de page afin de donner l'impression qu'il n'y a aucune image.

Pour ajouter une image d'arrière-plan, procédez comme suit :

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- Sélectionnez l'élément de rapport auquel ajouter une image d'arrière-plan.

Remarque

Les images d'arrière-plan s'appliquent uniquement aux éléments de type rectangle, zone de texte, tableau, matrice, liste ou corps de rapport.

- Dans la fenêtre Propriétés, développez **BackgroundImage**, puis procédez comme suit :
 - Pour **Source**, sélectionnez **External**, **Embedded** ou **Database**.

Remarque

Pour utiliser une image du projet, sélectionnez **External**.

- Pour **Value**, tapez ou sélectionnez une expression qui correspond à la source de l'image.

Si la propriété **Source** prend la valeur **External**, l'expression doit correspondre à un chemin d'accès valide à une image. Il peut s'agir d'un chemin statique d'accès à une image hébergée sur un serveur de rapports ou bien, d'un chemin d'accès basé sur la valeur d'un champ. Si l'image est stockée dans le projet, tapez son nom tel qu'il apparaît dans le projet.

Si la propriété **Source** prend la valeur **Embedded**, l'expression doit correspondre au nom d'une image incorporée dans le rapport. Si la propriété **Source** prend la valeur **Database**, l'expression doit correspondre à un champ contenant des données image binaires.

- Pour **MIMEType**, sélectionnez le type MIME approprié pour l'image.

Remarque

MIMEType s'applique uniquement si la propriété **Source** prend la valeur **Database**. Si la propriété **Source** prend la valeur **External** ou **Embedded**, la valeur de **MIMEType** est ignorée.

Pour **BackgroundRepeat**, sélectionnez **Repeat**, **NoRepeat**, **RepeatX** ou **RepeatY**.

IX. Fonctions intégrées pour les rapports (Concepteur de rapports Visual Studio)

ReportViewer fournit des fonctions intégrées que vous pouvez utiliser dans des expressions de rapport. Les fonctions intégrées peuvent être utilisées dans des expressions que vous incluez dans des fichiers de définition de rapport client (.rdlc) et dans des fichiers de définition de rapport (.rdl) sur un serveur de rapports SQL Server 2005 Reporting Services. La prise en charge des fonctions est fournie par les contrôles ReportViewer et Reporting Services.

Les fonctions intégrées sont organisées en deux catégories : fonctions d'agrégation et fonctions diverses. Exemple d'agrégation de données : calcul d'une somme de toutes les valeurs d'un champ particulier au moyen de la fonction **Sum**. Vous pouvez utiliser des fonctions d'agrégation dans des expressions pour tout élément de rapport.

Outre les fonctions intégrées, les expressions que vous incluez dans des rapports peuvent utiliser n'importe quelle fonction Visual Basic. Pour plus d'informations sur les fonctions Visual Basic, consultez *Functions (Visual Basic)*.

a. Fonctions diverses

Le tableau suivant décrit les fonctions d'usage général dont vous disposez.

Fonction	Description
Fonction InScope (Concepteur de rapports Visual Studio)	Indique si l'instance en cours d'un élément se trouve dans l'étendue spécifiée.
Fonction Level (Concepteur de rapports Visual Studio)	Retourne le niveau de profondeur actuel d'une hiérarchie récursive.
Fonction Previous (Concepteur de rapports Visual Studio)	Retourne l'instance précédente dans l'étendue spécifiée.

b. Agrégations standard

Le tableau suivant décrit les fonctions d'agrégation standard dont vous disposez.

Fonction	Description
Avg	Retourne la moyenne de toutes les valeurs non nulles de l'expression spécifiée.

Count	Retourne le nombre de valeurs de l'expression spécifiée.
CountDistinct	Retourne le nombre de toutes les valeurs distinctes de l'expression spécifiée.
CountRows	Retourne le nombre de lignes de l'étendue spécifiée.
First	Retourne la première valeur de l'expression spécifiée.
Last	Retourne la dernière valeur de l'expression spécifiée.
Max	Retourne la valeur maximale de toutes les valeurs non nulles de l'expression spécifiée.
Min	Retourne la valeur minimale de toutes les valeurs non nulles de l'expression spécifiée.
StDev	Retourne l'écart-type standard de toutes les valeurs non nulles de l'expression spécifiée.
StDevP	Retourne l'écart-type de population de toutes les valeurs non nulles de l'expression spécifiée.
Sum	Retourne la somme des valeurs de l'expression spécifiée.
Var	Retourne la variance de toutes les valeurs non nulles de l'expression spécifiée.
VarP	Retourne la variance de population de toutes les valeurs non nulles de l'expression spécifiée.

c. Agrégations cumulées

Le tableau suivant décrit les fonctions d'agrégation cumulée dont vous disposez.

Fonction	Description
RowNumber	Retourne le cumul de toutes les lignes de l'étendue spécifiée.
RunningValue	Utilise une fonction spécifiée pour retourner un agrégat cumulé de l'expression spécifiée.

d. Agrégations personnalisées

Le tableau décrit la fonction d'agrégation personnalisée.

Fonction	Description
Aggregate	Retourne une agrégation personnalisée de l'expression spécifiée, telle que définie par le fournisseur de données.

e. Scope

Chaque fonction d'agrégation utilise le paramètre Scope, qui définit l'étendue dans laquelle la fonction est exécutée. Une étendue valide peut être le nom d'un regroupement, d'un dataset ou d'une région de données. Seuls les regroupements ou régions de données contenant directement ou indirectement l'expression peuvent être utilisés en guise d'étendue. Pour les expressions situées dans une région de données, le paramètre Scope est facultatif pour toutes les fonctions d'agrégation. Si vous omettez le

paramètre Scope, l'étendue de l'agrégation correspond à la région de données ou au regroupement le plus à l'intérieur auquel appartient l'élément de rapport. Si vous spécifiez l'étendue **Nothing**, l'étendue est la région de données la plus à l'extérieur à laquelle appartient l'élément.

Pour les expressions à l'extérieur de régions de données, Scope se réfère à une table de données ou un objet d'entreprise. Si un rapport contient plusieurs datasets, le paramètre Scope est obligatoire. S'il ne contient qu'un dataset et que le paramètre Scope est omis, le dataset est défini comme l'étendue. Vous ne pouvez pas spécifier le mot clé Nothing pour les éléments de rapport situés à l'extérieur d'une région de données.

Vous ne pouvez pas utiliser le paramètre Scope dans les en-têtes et les pieds de page.

X. Définition des paramètres de rapport dans un rapport (Concepteur de rapports Visual Studio)

Vous pouvez définir des paramètres de rapport dans un fichier de définition de rapport client (.rdlc) pour prendre en charge une mise en forme conditionnelle ou pour utiliser dans des expressions ou du code. Vous ne pouvez pas mapper des paramètres de rapport à des paramètres de requête ni les utiliser dans des filtres. Dans un mode de traitement local, tout le traitement des données est assuré indépendamment du traitement des rapports.

Pour définir des paramètres, vous utilisez la boîte de dialogue **Paramètres du rapport** pour ajouter des paramètres à la définition de rapport definition.

Pour fournir une valeur de données aux paramètres au moment de l'exécution, vous devez écrire un code qui passe la valeur. En mode de traitement local, les contrôles ReportViewer ne fournissent pas une zone de paramètres pour accepter des valeurs de paramètres de l'utilisateur. Vous devez ajouter des champs au formulaire pour obtenir des valeurs de données, puis utiliser la méthode **SetParameters** pour passer ces valeurs au contrôle ReportViewer. Un exemple de code fourni illustre cette technique. Pour plus d'informations, consultez Exemple : Utilisation d'un flux RSS dans ReportViewer.

XI. Configuration de sous-rapports et de rapports d'extraction (Concepteur de rapports Visual Studio)

Vous pouvez utiliser des sous-rapports et des rapports d'extraction pour lier ensemble des rapports associés.

- Un sous-rapport est un contrôle incorporé à l'intérieur du corps d'un rapport parent. D'un point de vue conceptuel, un sous-rapport est similaire à un cadre dans une page Web qui reçoit le contenu d'une autre page Web. Le sous-rapport est rendu à l'intérieur du rapport parent qui le contient. Les deux rapports sont traités et affichés simultanément.
- Un rapport d'extraction s'ouvre en tant que résultat d'une action d'extraction définie sur un élément de rapport. Cette action ouvre généralement un rapport associé ou le même rapport avec des paramètres différents dans le même espace que le rapport parent. En général, le rapport d'extraction est associé au rapport parent par l'intermédiaire de données. Un exemple courant de rapport d'extraction est un rapport de ventes mensuelles contenant des liens vers des commandes individuelles de ce mois. Lorsque vous cliquez sur un lien d'extraction, le rapport parent est remplacé par un autre rapport fournissant les détails concernés.

Les rapports d'extraction et les sous-rapports prennent en charge de la notion de rapport parent et de rapport enfant. Pour relier les rapports, vous devez créer des paramètres de rapport pouvant être utilisés pour passer des données établissant la relation. Pour lier les rapports au moment de l'exécution, vous devez écrire un code prenant en charge la connexion. Lors du déploiement de rapports associés, assurez-vous que les fichiers de définition de rapport client (.rdlc) sont stockés ensemble dans le même dossier sur le système de fichiers.

a. Ajout de sous-rapports

Un sous-rapport et son rapport parent peuvent être associés au moyen de paramètres partagés. Vous devez ajouter des paramètres à chaque rapport. Les paramètres doivent être configurés avant que vous ajoutiez un sous-rapport au rapport parent. Lorsque vous ajoutez le sous-rapport, vous configurez le rapport parent pour passer des paramètres au sous-rapport.

Pour ajouter un sous-rapport

- a. Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- b. Dans la boîte à outils, cliquez sur **Sous-rapport**.
- c. Sur la surface de conception, faites glisser une zone de la taille souhaitée pour le sous-rapport. Une autre solution consiste à cliquer sur la surface de conception pour créer un sous-rapport de taille fixe.

Remarque

Si le sous-rapport existe dans le projet, vous pouvez faire glisser le rapport depuis l'Explorateur de solutions afin de créer un sous-rapport.

- d. Cliquez avec le bouton droit sur le sous-rapport, puis sur **Propriétés**.
- e. Dans **Sous-rapport**, sélectionnez le rapport devant apparaître dans la zone du sous-rapport.
- f. Sous l'onglet **Paramètres**, procédez comme suit :
 - Dans **Nom du paramètre**, tapez le nom d'un paramètre dans le sous-rapport. Ce nom doit correspondre à un paramètre de rapport présent dans le rapport désigné par la propriété **ReportName**.
 - Dans **Valeur du paramètre**, tapez une valeur à passer au sous-rapport. Cette valeur peut être du texte statique ou une expression pointant vers un champ ou un autre objet situé dans le rapport parent.
- g. Répétez l'étape 2 pour ajouter des paramètres supplémentaires à passer au sous-rapport.

Remarque

Tous les paramètres qui sont requis par le sous-rapport doivent figurer dans la liste **Paramètres**. S'il manque un paramètre obligatoire, le sous-rapport ne s'affiche pas correctement à l'intérieur du rapport parent.

Pour fournir des données pour des sous-rapports, votre application doit traiter l'événement **SubreportProcessing** de l'objet **LocalReport**. L'un des arguments passés à la méthode de gestionnaire d'événements **SubreportProcessing** est **SubreportProcessingEventArgs**. Dans le gestionnaire d'événements, l'application peut examiner les valeurs des paramètres passés aux sous-rapport à l'aide de la propriété **Parameters** de **SubreportProcessingEventArgs**. L'application doit ensuite fournir les données du sous-rapport en utilisant la propriété **DataSources** de **SubreportProcessingEventArgs**.

b. Configuration de rapports d'extraction

Vous pouvez configurer des rapports d'extraction en définissant des actions d'extraction assurant une navigation d'un rapport parent à un rapport secondaire contenant des données pertinentes.

Un rapport d'extraction comprend généralement des paramètres qui lui sont passés par le rapport parent. Dans l'exemple de rapport de synthèse sur les ventes, le rapport d'extraction comporte un paramètre qui prend pour valeur la commande. Le rapport de synthèse comprend un lien vers un rapport d'extraction pour chaque numéro de commande. Lorsque l'utilisateur clique sur l'un de ces liens, le rapport détaillé cible s'affiche, et le numéro de commande lui est passé.

Tout rapport stocké sur le serveur de rapports peut être un rapport d'extraction. Vous pouvez ajouter des liens d'extraction aux zones de texte et aux images uniquement.

Pour ajouter un lien de rapport d'extraction

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- Cliquez avec le bouton droit sur la zone de texte ou l'image à laquelle ajouter un lien, puis cliquez sur **Propriétés**.
- Sous l'onglet **Navigation**, sélectionnez **Aller au rapport**.
- Tapez ou sélectionnez un nom de rapport. Le rapport doit être un fichier .rdlc faisant partie du même projet.
- Pour définir les paramètres à passer au rapport d'extraction, cliquez sur **Paramètres**, puis procédez comme suit :
 - Dans **Nom du paramètre**, tapez les noms des paramètres de rapport dans le rapport d'extraction.

Remarque

Les noms dans la liste des paramètres doivent correspondre de façon exacte aux paramètres du rapport cible. Si les noms ne correspondent pas, ou si un paramètre attendu ne figure pas dans la liste, la génération du rapport d'extraction échoue.

- Dans **Valeur du paramètre**, tapez ou sélectionnez les valeurs à passer aux paramètres dans le rapport d'extraction.

Remarque

Ces valeurs peuvent contenir une expression qui correspond à une valeur à passer au paramètre de rapport. Les expressions de la liste de valeurs incluent la liste des champs du rapport en cours.

Pour fournir des données pour des rapports d'extraction, votre application doit traiter l'événement **Drillthrough** de l'objet **ReportViewer**. L'un des arguments passés à la méthode du gestionnaire d'événements **Drillthrough** est **DrillthroughEventArgs**. **DrillThroughEventArgs** a une propriété **Report** qui représente le rapport d'extraction. Dans le gestionnaire d'événements, l'application hôte peut examiner les valeurs des paramètres passés au rapport d'extraction en appelant la méthode **GetParameters()**. L'application hôte doit ensuite fournir les données du rapport d'extinction en utilisant la propriété **DataSources** du rapport d'extraction.

XII. Filtrer des données dans un rapport (Concepteur de rapports Visual Studio)

Vous pouvez définir des filtres sur une région de données pour sélectionner ou exclure les parties du dataset qui sont utilisées par la région de données. Les filtres limitent les données affichées à l'utilisateur après que toutes les données ont été récupérées. Comme l'ensemble de données est entièrement récupéré puis filtré lors du traitement du rapport, ce dernier peut ne pas s'exécuter aussi bien qu'un rapport bénéficiant d'un autre mode de filtrage des données (notamment, si vous écrivez un code qui filtre les données avant que celles-ci ne soient passées au rapport).

Pour ajouter un filtre à une région de données

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- Sélectionnez la région de données sur laquelle vous souhaitez définir le filtre. Si la région de données est un tableau ou une matrice, cliquez sur le tableau ou sur la matrice afin que les poignées de colonne et de ligne apparaissent au-dessus et sur le côté du tableau ou de la matrice.
- Cliquez avec le bouton droit sur la poignée d'angle du tableau ou de la matrice ou à un emplacement quelconque dans la liste ou le graphique, puis cliquez sur **Propriétés**.
- Sous l'onglet **Filtres**, pour **Filtre**, procédez comme suit :
 - Dans **Expression**, tapez ou sélectionnez l'expression que le filtre doit évaluer.

- Dans **Opérateur**, sélectionnez l'opérateur que le filtre doit utiliser pour comparer le champ évalué et la valeur.
- Dans **Valeur**, tapez l'expression ou la valeur par rapport à laquelle le filtre doit évaluer la valeur indiquée dans **Expression**.

Pour ajouter un filtre à un groupe d'un tableau ou d'une matrice

- a. Cliquez sur le tableau ou la matrice pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau ou de la matrice.
- b. Cliquez avec le bouton droit sur la poignée d'angle du tableau ou de la matrice, puis cliquez sur **Propriétés**.
- c. Sous l'onglet **Groupe**, sélectionnez le groupe à modifier, puis cliquez sur **Modifier**.
- d. Sous l'onglet **Filtres**, pour **Filtre**, procédez comme suit :
 - Dans **Expression**, tapez ou sélectionnez l'expression pour le champ que le filtre doit évaluer.
 - Dans **Opérateur**, sélectionnez l'opérateur que le filtre doit utiliser pour comparer le champ évalué et la valeur.
 - Dans **Valeur**, tapez l'expression ou la valeur par rapport à laquelle le filtre doit évaluer la valeur indiquée dans **Expression**.

Pour ajouter un filtre à un groupe d'une liste

- a. Cliquez avec le bouton droit sur la liste, puis sur **Propriétés**.
- b. Sous l'onglet **Général**, cliquez sur **Modifier le groupe de détails**.
- c. Sous l'onglet **Filtres**, pour **Filtre**, procédez comme suit :
 - Dans **Expression**, tapez ou sélectionnez l'expression pour le champ que le filtre doit évaluer.
 - Dans **Opérateur**, sélectionnez l'opérateur que le filtre doit utiliser pour comparer le champ évalué et la valeur.
 - Dans **Valeur**, tapez l'expression ou la valeur par rapport à laquelle le filtre doit évaluer la valeur indiquée dans **Expression**.

Pour ajouter un filtre à des groupes d'un graphique

- a. Cliquez avec le bouton droit sur le graphique, puis sur **Propriétés**.
- b. Sous l'onglet **Données**, sélectionnez un groupe de catégories ou un groupe de séries, puis cliquez sur **Modifier**.
- c. Sous l'onglet **Filtres**, pour **Filtre**, procédez comme suit :
 - Dans **Expression**, tapez ou sélectionnez l'expression pour le champ que le filtre doit évaluer.
 - Dans **Opérateur**, sélectionnez l'opérateur que le filtre doit utiliser pour comparer le champ évalué et la valeur.
 - Dans **Valeur**, tapez l'expression ou la valeur par rapport à laquelle le filtre doit évaluer la valeur indiquée dans **Expression**.

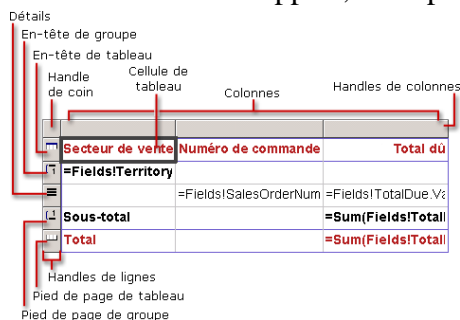
XIII. Ajout de régions de données de tableau (Concepteur de rapports Visual Studio)

Une région de données de tableau est un élément de rapport lié à des données qui contient des données multi-colonne, multi-instance sur un rapport. Un tableau a un ensemble statique de colonnes et un nombre variable de lignes selon les données du dataset. Les tableaux peuvent contenir un nombre illimité de colonnes.

Une région de données de tableau est composée de plusieurs parties. Elle inclut au minimum des lignes de détails. Vous pouvez également spécifier des en-têtes de tableau, des pieds de page de tableau, des en-têtes de groupe et des pieds de page de groupe. Vous pouvez imbriquer d'autres éléments de rapport liés aux données dans une région de données de tableau, notamment un autre tableau.

1. Utilisation de régions de données de tableau

Lors de la conception d'un rapport, vous ajoutez une région de données de tableau en sélectionnant **Tableau** dans la boîte à outils et en la faisant glisser sur le rapport. Vous pouvez également ajouter un tableau à un nouveau rapport vide ou à un rapport existant. Une fois que la région de données de tableau est incluse dans le rapport, vous pouvez lui ajouter des colonnes et des lignes.



2. Poignées

Pour travailler avec une région de données d'un tableau, vous utilisez des *poignées* qui se trouvent sur les colonnes, les lignes ; les poignées de tableau sont des cases grises qui apparaissent au-dessus et en regard du tableau lorsque celui-ci est sélectionné. Les poignées qui s'étendent le long de la partie supérieure du tableau sont des poignées de colonne. Les poignées qui s'étendent le long du bord du tableau sont des poignées de ligne. La poignée qui se trouve à l'intersection des poignées de colonne et de ligne s'appelle la poignée d'angle. La plupart des actions sur les colonnes, les lignes et le tableau s'effectuent en cliquant avec le bouton droit respectivement sur les poignées de colonne, les poignées de ligne ou la poignée d'angle. Pour sélectionner le tableau, cliquez sur la poignée d'angle.

3. Comment ajouter, déplacer ou supprimer un tableau

Pour ajouter un tableau

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de conception graphique.
- Dans la boîte à outils, cliquez sur **Tableau**.
- Cliquez sur la surface de dessin du rapport.

Une fois que le tableau se trouve sur le rapport, vous pouvez le déplacer en pointant sur sa bordure ombrée et en le faisant glisser à un nouvel emplacement (cliquez n'importe où sur une zone vide du tableau pour faire apparaître la bordure ombrée). Pour supprimer un tableau, cliquez avec le bouton droit sur un espace vide à l'intérieur du tableau et appuyez sur SUPPR.

Chaque région de données du tableau est associée à une source de données. Si le rapport contient un seul dataset, le tableau est automatiquement associé à ce dataset lorsque vous le placez dans le rapport. Si le rapport contient plusieurs datasets, vous devez associer le tableau à un dataset spécifique en faisant glisser des champs sur le tableau.

4. Comment insérer des colonnes et des lignes

Vous pouvez ajouter de nouveaux nœuds à la hiérarchie. Vous pouvez également ajouter des lignes. Notez que chaque ligne contient un type de données spécifique. Un tableau ne peut comporter qu'une seule ligne Détails, mais vous pouvez ajouter de nouvelles lignes pour créer des groupes.

Pour insérer une colonne, cliquez dans le tableau afin de faire apparaître les poignées de colonne et de ligne au-dessus et en regard du tableau. Cliquez avec le bouton droit sur une poignée de colonne, puis cliquez sur **Insérer une colonne à droite** ou sur **Insérer une colonne à gauche**.

Pour insérer une ligne, cliquez avec le bouton droit sur une poignée de ligne à l'emplacement où vous souhaitez insérer une ligne, puis cliquez sur **Insérer une ligne au-dessus** ou sur **Insérer une ligne en**

dessous. Le type de ligne sélectionné détermine le type de ligne inséré. Par exemple, si vous sélectionnez une ligne d'en-tête ou de pied de page de groupe, le type de ligne qui sera inséré est un autre en-tête ou pied de page de groupe.

5. Comment lier des données à un tableau

Après l'ajout d'un tableau à la mise en page d'un rapport, vous ajoutez des champs ou des expressions aux cellules du tableau.

Pour lier des données à un tableau, faites glisser un champ de la fenêtre Sources de données sur une cellule de tableau. Si vous faites glisser le champ dans la cellule d'une colonne de tableau sans en-tête (par exemple, la ligne Détails), le nom du champ vient automatiquement s'insérer dans l'en-tête de la colonne.

Remarque

Si la fenêtre Sources de données n'apparaît pas, dans le menu **Données**, cliquez sur **Afficher les sources de données**.

Chaque cellule du tableau contient par défaut une zone de texte. Vous pouvez changer le type d'élément à l'intérieur de la cellule (par exemple, vous pouvez changer une zone de texte dans une cellule en une image). Leurs cellules peuvent recouvrir plusieurs colonnes.

Les données d'un tableau peuvent être rassemblées dans des groupes, puis triées. Par exemple, vous pourriez regrouper des produits en catégories, en départements, puis trier les données en ordre croissant. Pour plus d'informations sur le regroupement et le tri de données, consultez Regroupement de données dans un rapport (Concepteur de rapports Visual Studio) et Tri de données dans un rapport (Concepteur de rapports Visual Studio).

6. Comment regrouper des données dans un tableau

Pour ajouter un groupe à un tableau

- Cliquez sur le tableau pour que les poignées de ligne et de colonne apparaissent au-dessus et en regard du tableau.
- Cliquez avec le bouton droit sur la poignée de la ligne où vous souhaitez insérer le groupe, puis cliquez sur **Insérer un groupe**.

Remarque

L'emplacement de la ligne du nouveau groupe est déterminé par la ligne qui est sélectionnée. Si la ligne sélectionnée est une ligne de détails, le nouveau groupe vient se placer juste à l'extérieur de la ligne de détails. Si la ligne sélectionnée est une ligne de groupe, le nouveau groupe est positionné à l'intérieur de la ligne de groupe.

- Sous l'onglet **Général**, procédez comme suit :
 - Dans **Nom**, tapez le nom du groupe.
 - Dans **Grouper sur**, tapez ou sélectionnez les expressions selon lesquelles vous voulez regrouper les données.
 - (Facultatif) Dans **Étiquette du plan du document**, tapez ou sélectionnez l'expression à utiliser en tant qu'étiquette du plan du document.
 - (Facultatif) Si ce groupe constitue une hiérarchie récursive, pour **Groupe parent**, tapez ou sélectionnez l'expression à utiliser en tant que groupe parent récursif.
 - (Facultatif) Cliquez sur **Saut de page au début** ou sur **Saut de page à la fin** pour insérer un saut de page au début ou à la fin de chaque instance de groupe.
 - (Facultatif) Cliquez sur **Inclure l'en-tête du groupe** ou sur **Inclure le pied de page du groupe** pour placer un en-tête ou un pied de page pour le groupe dans le tableau.

- (Facultatif) Cliquez sur **Répéter l'en-tête de groupe** ou sur **Répéter le pied de page de groupe** pour répéter l'en-tête ou le pied de page de groupe sur chaque page dans laquelle le tableau apparaît.
- d. (Facultatif) Sous l'onglet **Tri**, sélectionnez ou tapez les expressions sur lesquelles trier les données dans le groupe.
- e. (Facultatif) Sous l'onglet **Filtres**, sélectionnez ou tapez les expressions sur lesquelles filtrer les données dans le groupe.
- f. (Facultatif) Sous l'onglet **Visibilité**, sélectionnez les options de visibilité de l'élément. Pour plus d'informations sur la visibilité, consultez Ajout d'interactivité, de visibilité et de navigation à un rapport (Concepteur de rapports Visual Studio).
- g. (Facultatif) Sous l'onglet **Sortie des données**, sélectionnez les options de sortie des données de l'élément.

7. Fusion de cellules

Plusieurs cellules contiguës d'un tableau peuvent être combinées pour n'en former qu'une seule. Cette opération se nomme recouvrement de colonnes ou fusion de cellules. Les cellules peuvent uniquement être réunies entre des colonnes adjacentes. Lors de la fusion, seules les données de la première cellule sont préservées. Si des données sont présentes dans d'autres cellules, elles sont supprimées. Les cellules fusionnées peuvent être fractionnées de sorte que les colonnes initiales soient rétablies.

Pour fusionner des cellules, sélectionnez-les, cliquez avec le bouton droit sur la sélection, puis cliquez sur **Fusionner les cellules**. Pour fractionner des cellules fusionnées, cliquez avec le bouton droit sur ces dernières, puis sélectionnez **Fractionner les cellules**.

8. Ajout de sous-totaux

Pour ajouter un sous-total à un tableau, ajoutez une expression d'agrégation à une cellule dans une ligne de groupe. Imaginons par exemple un tableau regroupé par catégories de produits, par sous-catégories de produits et par produits. Si vous voulez afficher la somme des ventes par catégorie et par sous-catégorie, vous devez placer l'expression `=Sum(Fields!Sales.Value)` dans les lignes d'en-tête ou de pied de page pour la catégorie et la sous-catégorie. **Sum** est une fonction d'agrégation. Reporting Services calcule la somme des valeurs de ce groupe et affiche le sous-total.

9. Ajout de régions de données à un rapport (Concepteur de rapports Visual Studio)

Les régions de données sont des éléments de rapport liés aux données qui affichent des lignes de données répétées à partir de datasets sous-jacents. Vous pouvez ajouter des régions de données à un rapport pour créer des rapports tabulaires, matriciels, graphiques et de listes.

Pour ajouter une région de données, faites-la glisser de la boîte à outils à la disposition du rapport. La région de données apparaît sur le rapport en mode de conception graphique ; chaque partie de la région de données que vous pouvez spécifier comporte une représentation visuelle afin que vous puissiez voir où placer les champs et définir des propriétés.

Pour ajouter des champs à une région de données, faites simplement glisser des champs de la fenêtre Source de données sur la disposition de la région de données. La plupart des régions de données incluent des étiquettes qui identifient où placer les champs.

Pour configurer une région de données, vous devez définir des propriétés. Selon le type de région de données avec lequel vous travaillez, vous pouvez ajouter des étiquettes, des expressions incorporées, spécifier des actions et appliquer une mise en forme. Pour les régions de données qui sont composées de

plusieurs parties (par exemple, un tableau ou une matrice), vous pouvez définir des propriétés ayant une incidence sur des zones spécifiques de la région de données.

Un rapport peut contenir plusieurs régions de données. Chaque région de données ne peut contenir des données que d'un seul dataset. Pour utiliser les données de sources multiples dans une région de données spécifique (par exemple, dans un tableau ou un graphique spécifique), vous devez combiner les données dans un dataset unique avant de concevoir le rapport.

Vous pouvez imbriquer des régions de données dans d'autres régions de données. Par exemple, si vous souhaitez créer un rapport des ventes pour chaque commercial existant dans une base de données, vous pouvez créer une liste avec des zones de texte et une image pour afficher des informations sur un employé, puis ajouter des régions de données de type tableau et graphique pour afficher les ventes effectuées par cet employé.

Remarque

Les régions de données sont conçues pour recevoir des lignes ou collections de données répétées. Si vous associez une collection de données à un élément de rapport d'une instance spécifique (notamment une zone de texte), vous voyez uniquement le premier élément de la collection au moment de l'exécution.

10. Types de régions de données

La section suivante résume chaque région de données. Pour plus d'informations sur les autres éléments de rapports liés aux données, consultez Ajout de zones de texte à un rapport (Concepteur de rapports Visual Studio) et Ajout d'images à un rapport (Concepteur de rapports Visual Studio). Pour afficher la liste complète de tous les éléments que vous pouvez ajouter à un rapport, consultez Vue d'ensemble des rapports (Visual Studio).

a. Tableau

Un tableau est une région de données qui présente les données ligne par ligne. Les colonnes de tableau sont statiques. Les lignes de tableau s'étendent vers le bas pour contenir les données. Vous pouvez ajouter aux tableaux des groupes, qui organisent les données par champs ou expressions sélectionnés. Pour plus d'informations sur l'ajout d'un tableau dans un rapport à l'aide du Concepteur de rapports, consultez Ajout de régions de données de tableau (Concepteur de rapports Visual Studio).

b. Matrice

Une matrice est une région de données qui contient des colonnes et des lignes qui s'étendent pour contenir les données. Une matrice est également qualifiée de tableau croisé dynamique. Une matrice peut comporter des colonnes et des lignes dynamiques et statiques. Les colonnes ou les lignes peuvent contenir d'autres colonnes ou lignes, et peuvent être utilisées pour regrouper des données. Pour plus d'informations sur l'ajout d'une matrice dans un rapport à l'aide du Concepteur de rapports, consultez Ajout de régions de données de matrice (Concepteur de rapports Visual Studio).

c. Liste

Une liste est une région de données qui présente des données organisées librement. Vous pouvez organiser les éléments de rapport de façon à créer un formulaire avec des zones de texte, des images et d'autres régions de données placées aux emplacements de votre choix dans la liste. Pour plus d'informations sur l'ajout d'une liste à un rapport à l'aide du Concepteur de rapports, consultez Ajout de régions de données de liste (Concepteur de rapports Visual Studio).

d. Graphique

Un graphique présente les données graphiquement. Les exemples de graphiques courants sont les graphiques à barres, à secteurs et en courbes, mais de nombreux autres styles de graphiques sont pris en charge. Pour plus d'informations sur l'ajout d'un graphique dans un rapport à l'aide du Concepteur de rapports, consultez Ajout de graphiques à un rapport (Concepteur de rapports Visual Studio).

11. Ajout de régions de données de matrice (Concepteur de rapports Visual Studio)

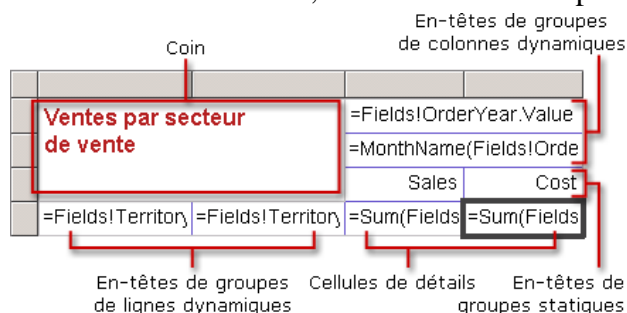
Une matrice est un élément de rapport lié aux données dans lequel les données sont organisées en colonnes et en lignes qui se croisent en des points de données spécifiques. Elle fournit des fonctionnalités similaires à celles des analyses croisées et des tableaux croisés dynamiques. À la différence d'une table, qui est un ensemble statique de colonnes, une matrice peut comporter des colonnes dynamiques. Vous pouvez définir des matrices contenant des lignes et des colonnes statiques et dynamiques.

Remarque

Si vous exportez un rapport de matrice vers Excel, la totalité des lignes et des colonnes sont visibles dans la feuille de calcul, indépendamment de la configuration des propriétés de visibilité.

a. Création d'une région de données de matrice

Lorsque vous créez une matrice pour la première fois, celle-ci comprend quatre cellules. La cellule supérieure gauche se nomme cellule de coin. Elle peut s'utiliser pour afficher une étiquette pour la matrice ou demeurer vide. La cellule supérieure droite est un en-tête de colonne et peut contenir un champ ou une expression de regroupement de données. La cellule inférieure gauche est un en-tête de ligne et peut comporter elle aussi un champ ou une expression de regroupement de données. Quant à la cellule inférieure droite, elle contient une expression d'agrégation pour les données de détails.



Lors de l'exécution du rapport, des en-têtes de colonnes dynamiques se développent vers la droite (ou la gauche si la propriété **Direction** de la matrice est définie avec la valeur RTL), le nombre de colonnes étant égal au nombre de groupes existants. Les lignes dynamiques se développent vers le bas de la page. Les données qui apparaissent dans les cellules de détails sont des agrégats basés sur les intersections des colonnes et des lignes.

b. Ajout, déplacement et suppression d'une matrice

Pour ajouter une matrice

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de création graphique.
- Dans la boîte à outils, cliquez sur **Matrice**.
- Cliquez sur la zone de conception du rapport.

Une fois la matrice positionnée sur le rapport, vous pouvez la déplacer en pointant sur sa bordure ombrée, puis en la faisant glisser jusqu'au nouvel emplacement (cliquez en un point quelconque d'une zone vide de la matrice pour faire apparaître la bordure ombrée). Pour supprimer une matrice, cliquez avec le bouton droit sur un espace vide dans celle-ci, puis appuyez sur la touche Suppr.

c. Liaison de données à une matrice

Après avoir créé une matrice, vous pouvez y ajouter des champs. Chaque cellule de la matrice contient par défaut une zone de texte. Vous pouvez taper n'importe quelle expression dans n'importe quelle cellule, ou bien vous pouvez remplacer l'élément de la cellule par un autre (par exemple, une zone de texte par une image). Pour lier des données à une matrice, faites glisser un champ depuis la fenêtre **Sources de données** jusqu'à une cellule de la matrice. Si vous faites glisser le champ dans la cellule d'une colonne de matrice sans en-tête (par exemple, la ligne de Détails), le nom du champ vient automatiquement s'insérer dans l'en-tête de la colonne.

Remarque

Si la fenêtre Sources de données n'apparaît pas, dans le menu **Données**, cliquez sur **Afficher les sources de données**.

Chaque matrice d'un rapport est associée à un dataset. Si le rapport contient un seul dataset, la matrice est automatiquement associée à ce dataset lorsque vous la placez dans le rapport. Si le rapport contient plusieurs datasets, vous devez associer explicitement la matrice au dataset correct.

12. Colonnes et lignes dynamiques (groupes)

Vous pouvez ajouter des lignes et des colonnes dynamiques à la matrice par défaut pour regrouper les données par champ. Quand vous ajoutez une nouvelle colonne dynamique en créant un nouvel en-tête de ligne ou de colonne, ce nouvel en-tête est imbriqué dans l'en-tête d'origine. Lors de l'exécution du rapport, il est répété à l'intérieur de l'en-tête d'origine. Ainsi, une ligne ou une colonne dynamique imbriquée peuvent avoir un en-tête contenant un champ pour la région et, dans cet en-tête, un autre en-tête comportant un champ pour la ville.

Vous pouvez ajouter des colonnes et des lignes dynamiques en faisant glisser des champs de la liste des champs vers la matrice. Lorsque vous faites glisser un champ sur une matrice comportant un en-tête de ligne ou de colonne, vous pouvez le placer à l'intérieur ou à l'extérieur de cet en-tête. Dans le Concepteur de rapports, une barre s'affiche en haut ou en bas d'un en-tête de colonne ou bien à gauche ou à droite d'un en-tête de ligne, en fonction de l'endroit où vous déposez le champ. Par exemple, pour créer une nouvelle colonne dynamique contenant une colonne dynamique existante, faites glisser le champ vers l'en-tête de colonne existant, positionnez-le de sorte que la barre s'affiche sur la bordure supérieure de la cellule d'en-tête, puis déposez le champ.

Pour ajouter une colonne ou une ligne dynamique à une matrice

- Faites glisser le champ depuis la fenêtre Sources de données vers un en-tête de colonne ou de ligne de la matrice.
- Répétez l'étape 1 pour ajouter plusieurs lignes ou colonnes dynamiques à une matrice. La colonne ou la ligne se scinde pour créer une autre colonne ou une autre ligne dynamique. La position de la nouvelle colonne ou ligne dépend de l'endroit où le champ a été placé. Pour l'ajout d'une colonne, une barre apparaît au-dessus ou sous la cellule existante, et pour l'ajout d'une ligne, une barre apparaît à gauche ou à droite de la cellule existante.

13. Colonnes et lignes statiques

Vous pouvez ajouter des colonnes et des lignes statiques pour afficher des données de détail supplémentaires. Quand vous insérez une colonne ou une ligne statique, le Concepteur de rapports divise l'en-tête en deux. Au lieu d'organiser les en-têtes en les joignant, chaque cellule de détail est affichée côte à côte avec les en-têtes contenant une étiquette statique. Par exemple, une colonne ou une ligne statique peut être une cellule de détail contenant un champ des recettes projetées, placée à côté d'une autre cellule de détail contenant un champ des recettes réelles.

Pour ajouter une ligne ou une colonne statique à une matrice

- Cliquez avec le bouton droit sur la cellule de données (détails) de la matrice, puis cliquez sur **Ajouter une colonne** ou sur **Ajouter une ligne**. Une autre solution consiste à faire glisser un champ depuis la fenêtre Sources de données vers une cellule de données remplie.
- Répétez l'étape 1 pour ajouter plusieurs lignes ou colonnes statiques à une matrice. La cellule se divise pour créer une autre ligne ou colonne statique. Si vous ajoutez une colonne ou une ligne en faisant glisser un champ, une ligne noire s'affiche sur un côté de la cellule. Cette ligne indique l'emplacement du champ lorsqu'il sera déposé sur la matrice.

Remarque

Lorsqu'une matrice contient une seule ligne ou colonne statique, la cellule n'a pas d'en-tête de colonne ou de ligne. Lorsque plusieurs lignes ou colonnes statiques sont ajoutées à une matrice, un en-tête statique est créé pour chaque colonne ou ligne statique.

14. Ajout de sous-totaux

Pour insérer un sous-total dans une matrice, il faut l'ajouter à un groupe individuel de la matrice. Par défaut, les groupes ne possèdent pas de sous-total. Pour ajouter un sous-total à un groupe, cliquez avec le bouton droit sur l'en-tête de ligne ou de colonne du groupe, puis choisissez **Sous-total**. Un nouvel en-tête s'affiche pour le sous-total. Le contrôle ReportViewer calculera le sous-total du groupe.

15. Affichage de données de l'un des côtés des en-têtes de ligne

Vous n'êtes pas tenu d'afficher des en-têtes de ligne sur le côté de la matrice. Vous pouvez les placer entre les colonnes, de sorte que les colonnes de données soient placées avant ces en-têtes. Pour cela, modifiez la propriété **GroupsBeforeRowHeaders** de la matrice. Cette propriété est accessible via la fenêtre **Propriétés** ou l'onglet **Général** de la boîte de dialogue **Propriétés de la matrice**. Sa valeur est un entier. Par exemple, la valeur 2 affiche deux groupes de données de matrice avant la colonne contenant les en-têtes de ligne.

XIV. Ajout de régions de données de liste (Concepteur de rapports Visual Studio)

Une région de données de liste est un élément de rapport lié aux données qui contient une colonne unique de données multi-instance sur un rapport. Elle peut s'employer pour les rapports de forme libre ou conjointement avec d'autres régions de données. Vous pouvez définir des listes qui contiennent un nombre quelconque d'éléments de rapport. Une liste peut être imbriquée dans une autre pour offrir plusieurs regroupements de données.

A. Ajout, déplacement et suppression d'une liste**Pour ajouter une liste**

- Ouvrez le fichier de définition de rapport client (.rdlc) en mode de création graphique.
- Dans la boîte à outils, cliquez sur **Liste**.
- Dans la surface de conception, tracez une zone correspondant à la taille de la liste. Vous pouvez également cliquer dans la surface de conception pour créer une liste dont la taille est fixe.

Une fois la liste positionnée sur le rapport, vous pouvez la déplacer en pointant sur sa bordure ombrée puis en la faisant glisser jusqu'au nouvel emplacement (cliquez en un point quelconque d'une zone vide pour faire apparaître la bordure ombrée). Pour supprimer une liste, cliquez avec le bouton droit sur un espace vide dans celle-ci, puis appuyez sur la touche Suppr.

B. Liaison de données à une liste

Pour lier des données à une liste, faites glisser un champ depuis la fenêtre Sources de données jusqu'à la liste.

Remarque

Si la fenêtre Sources de données n'apparaît pas, dans le menu **Données**, cliquez sur **Afficher les sources de données**.

Les données d'une liste peuvent être regroupées puis triées. Par exemple, vous pouvez regrouper les produits en catégories au sein des départements, puis trier les données dans l'ordre croissant. Pour plus d'informations sur le regroupement et le tri des données, consultez Regroupement de données dans un rapport (Concepteur de rapports Visual Studio) et Tri de données dans un rapport (Concepteur de rapports Visual Studio).

C. Regroupement des données d'une liste

Pour ajouter un groupe à une liste

- a. Cliquez avec le bouton droit sur la liste, puis cliquez sur **Propriétés**.
- b. Sous l'onglet **Général**, cliquez sur **Modifier le groupe de détails**. La boîte de dialogue Regroupement des détails s'affiche.
- c. Sous l'onglet **Général**, procédez comme suit :
 - Dans **Nom**, tapez le nom du groupe.
 - Dans **Regrouper sur**, tapez ou sélectionnez les expressions par lesquelles vous voulez regrouper les données.
 - (Facultatif) Dans **Étiquette du plan du document**, tapez ou sélectionnez l'expression à utiliser en tant qu'étiquette du plan du document.
 - (Facultatif) Si ce groupe constitue une hiérarchie récursive, pour **Groupe parent**, tapez ou sélectionnez l'expression à utiliser en tant que groupe parent récursif.
 - (Facultatif) Cliquez sur **Saut de page au début** ou sur **Saut de page à la fin** pour insérer un saut de page au début ou à la fin de chaque instance de groupe.
- d. (Facultatif) Sous l'onglet **Filtres**, sélectionnez ou tapez les expressions sur lesquelles filtrer les données dans le groupe.
- e. (Facultatif) Sous l'onglet **Visibilité**, sélectionnez les options de visibilité de l'élément. Pour plus d'informations sur la visibilité, consultez Ajout d'interactivité, de visibilité et de navigation à un rapport (Concepteur de rapports Visual Studio).
- f. (Facultatif) Sous l'onglet **Sortie des données**, sélectionnez les options de sortie des données de l'élément, puis cliquez sur **OK**.

XV. Tri de données dans un rapport (Concepteur de rapports Visual Studio)

Les données présentes dans chacune des régions de données (table, matrice et liste) d'un rapport peuvent être triées selon des champs et des expressions. Vous pouvez configurer un rapport de manière à ce que les utilisateurs finaux puissent le trier à leur guise de façon interactive lorsqu'ils le visualisent. Vous pouvez aussi utiliser des paramètres pour trier les données avant de les afficher dans le rapport.

1. Définition des propriétés de tri sur les régions de données

Les régions de données sont des éléments de rapport liés aux données permettant d'afficher des lignes de données répétitives. Vous pouvez définir des propriétés de tri sur n'importe quelle région de données afin de déterminer le mode et l'ordre de tri des lignes.

- Pour spécifier le tri sur un tableau ou sur une liste, définissez les options de la boîte de dialogue Propriétés du tableau - Onglet Tri (Concepteur de rapports Visual Studio) ou Propriétés de la liste - Onglet Tri (Concepteur de rapports Visual Studio).

- Pour spécifier le tri sur une matrice, ouvrez l'onglet Groupes de la boîte de dialogue Boîte de dialogue Propriétés de la matrice - Onglet Général (Concepteur de rapports Visual Studio), puis cliquez sur **Modifier** pour ouvrir la boîte de dialogue Propriétés de regroupement et de tri - Onglet Tri (Concepteur de rapports Visual Studio).

2. Tri interactif

Vous pouvez définir des propriétés sur les en-têtes de colonne afin de permettre le tri interactif d'un rapport publié. Le tri est spécifié par le biais de propriétés de zone de texte. Vous pouvez définir le tri pour plusieurs colonnes d'un tableau, d'une liste ou d'une matrice, et pour des données imbriquées ou regroupées.

Pour définir le tri interactif

- a. Cliquez avec le bouton droit sur l'en-tête d'une colonne (par exemple, une colonne dans un en-tête de table), puis sélectionnez **Propriétés** pour ouvrir la boîte de dialogue Propriétés de la zone de texte.
- b. Cliquez sur l'onglet **Tri interactif**.
- c. Sélectionnez **Ajouter une action de tri interactif à cette zone de texte**.
- d. Pour spécifier une expression de tri, sélectionnez le champ correspondant à la colonne pour laquelle vous définissez une action de tri (par exemple, pour un en-tête de colonne nommé « Department », choisissez =Fields!Department.Value). La spécification d'une expression de tri est obligatoire.
- e. Sélectionnez l'étendue et la région de données du tri. Cette étape détermine si l'action de tri s'applique à toutes les régions de données d'un rapport ou si elle est limitée à la région de données qui contient la zone de texte ou à un ensemble de régions de données de votre choix.
- f. Cliquez sur **OK**.

Pour vérifier l'action de tri, vous pouvez afficher un aperçu du rapport. Les colonnes qui prennent en charge le tri interactif ont des icônes fléchées indiquant l'ordre du tri. Pour passer du tri par ordre croissant au tri par ordre décroissant, et vice versa, cliquez sur l'en-tête de colonne.

3. Tri paramétré

Vous pouvez utiliser des paramètres de rapport pour modifier les propriétés de tri d'une région de données ou d'un groupe. Par exemple, vous pouvez modifier l'expression de tri pour le tableau Product du rapport, de façon à ce que le tri s'effectue sur le nom de produit ou sur le prix. Au moment de l'exécution, l'utilisateur sélectionne le champ de tri à utiliser.

- Créez un paramètre de rapport à utiliser pour accepter la sélection de tri de l'utilisateur.
- Dans la boîte de dialogue **Paramètres du rapport**, sélectionnez les champs sur lesquels le tri peut être effectué. Les valeurs disponibles doivent correspondre à des champs du dataset. Si le tri porte sur un regroupement, vous pouvez spécifier un ordre de tri sur des valeurs agrégées. Le tri sur des valeurs agrégées n'est pas pris en charge pour les datasets ou les régions de données.
- Ajoutez une liste déroulante à la page Web ou de formulaire afin que l'utilisateur puisse opérer un choix parmi les valeurs disponibles.

D. Regroupement de données dans un rapport (Concepteur de rapports Visual Studio)

Les données présentes dans les régions de données de tableau, de matrice et de liste peuvent être regroupées d'après des champs et des expressions. Vous pouvez utiliser des groupes à l'intérieur d'un tableau afin de distinguer des sections logiques de données au sein du tableau. Vous pouvez aussi ajouter des sous-totaux et d'autres expressions à l'en-tête ou au pied de page du groupe. Dans une matrice, les groupes sont affichés sous la forme de lignes ou de colonnes dynamiques. Vous pouvez imbriquer des groupes dans d'autres groupes et y ajouter des sous-totaux. Vous pouvez utiliser des listes pour créer des

groupes séparés dans un rapport et même créer des groupes imbriqués en plaçant des listes dans d'autres listes.

1. Définition des propriétés de groupe sur les régions de données

Les régions de données sont des éléments de rapport liés aux données permettant d'afficher des lignes de données répétitives. Vous pouvez définir des propriétés de groupe sur n'importe quelle région de données pour organiser la présentation des données.

- Pour savoir comment spécifier des groupes sur un tableau ou sur une liste, consultez Ajout de régions de données de tableau (Concepteur de rapports Visual Studio) et Ajout de régions de données de liste (Concepteur de rapports Visual Studio).
- Pour spécifier des groupes sur une matrice, ajoutez des lignes ou des colonnes dynamiques. Pour plus d'informations, consultez Ajout de régions de données de matrice (Concepteur de rapports Visual Studio).

2. Hiérarchies récursives

Une hiérarchie récursive est une hiérarchie de données dans laquelle toutes les relations parent-enfant sont représentées dans les données. Par exemple, il est possible de créer un organigramme illustrant les relations entre directeur et employé à l'aide d'une hiérarchie récursive. Dans une hiérarchie de ce type, la table comporte une colonne pour les matricules (ID) des employés et une autre pour ceux des directeurs. L'ID de directeur fait référence à l'ID d'employé d'un autre employé, ce qui donne une hiérarchie du personnel.

Pour construire une hiérarchie récursive, vous devez définir certaines propriétés pour un groupe dans une région de données. Utilisez un champ contenant un ID unique (par exemple, l'ID d'employé) en tant qu'expression de groupe, et ensuite un champ contenant l'ID du parent (par exemple, l'ID de directeur) dans la propriété **Parent**. Un groupe défini en tant que hiérarchie récursive (c'est-à-dire un groupe utilisant la propriété **Parent**) peut comporter une et une seule expression de groupe.

Suivez les étapes ci-après pour créer une hiérarchie récursive utilisant la table **Employee** de la base de données **Gesco**. Ce didacticiel suppose que vous savez comment créer des rapports, des datasets, des requêtes et des tables. Pour plus d'informations sur ces opérations, consultez les sections qui y sont consacrées dans la documentation.

1. Créez un dataset basé sur la base de données **Gesco** qui renvoie des données à partir de la requête suivante :

```
SELECT FirstName, LastName, EmployeeID, ManagerID
```

2. Ajoutez une définition de rapport client (.rdlc) au projet, puis ouvrez le rapport en mode de création graphique.
3. Ajoutez une région de données de tableau à la mise en page du rapport.
4. Dans la première cellule de détail de la table, tapez l'expression suivante :
=Fields!FirstName.Value & " " & Fields!LastName.Value
5. Cliquez avec le bouton droit sur le coin de la table, puis sélectionnez **Propriétés**.
6. Dans l'onglet **Groupe**, cliquez sur **Regroupement des détails**.
7. Dans l'onglet **Général**, tapez ou sélectionnez l'expression suivante dans la zone **Expression** :
=Fields!EmployeeID.Value
8. Dans la zone **Groupe parent**, tapez ou sélectionnez l'expression suivante :
=Fields!ManagerID.Value

3. Fonction Level

Vous pouvez utiliser la fonction **Level** dans la définition de la marge intérieure de la zone de texte pour appliquer un retrait aux noms des employés en fonction du niveau que ceux-ci occupent dans la

hiérarchie. Si nous reprenons l'exemple et la table ci-dessus, vous devez pour cela utiliser l'expression suivante pour la marge intérieure gauche de la zone de texte dans la première cellule de détail :

```
=Convert.ToString(2 + (Level()*10)) & "pt"
```

Les propriétés de marge intérieure nécessitent toutes une chaîne au format *nnxx*, où *nn* est un nombre et *xx*, une unité de mesure. Par défaut, la marge intérieure d'une zone de texte est de 2 points. L'expression ci-dessus crée une chaîne utilisant la fonction **Level** pour agrandir la marge intérieure en fonction du niveau de l'employé. Ainsi, une ligne de niveau 1 implique une marge intérieure de 12 points (2 + (1*10)), alors qu'une ligne de niveau 3 correspond à une marge intérieure de 32 points (2 + (3*10)).

Pour plus d'informations sur les fonctions utilisables, consultez Fonctions intégrées pour les rapports (Concepteur de rapports Visual Studio).

E. Définition de la taille de la page et des sauts de page dans un rapport (Concepteur de rapports Visual Studio)

Vous pouvez contrôler la taille de la page et les sauts de page en spécifiant les propriétés de page dans la définition de rapport. Le type de contrôle utilisé et le format d'exportation servant à visualiser un rapport peuvent affecter la pagination. Le mode de visualisation d'un rapport détermine sa pagination. La liste suivante décrit brièvement le mode de pagination pour différents scénarios :

- Le contrôle Windows Forms présente par défaut un rapport dans le format GDI (Graphical Device Instrumentation). Ce format de sortie utilise des sauts de page conditionnels (ou sauts de page logiques) qui sont calculés au moment de l'exécution.
- Le contrôle de serveur Web présente un rapport dans le format HTML, qui utilise des sauts de page conditionnels (ou sauts de page logiques) calculés au moment de l'exécution. Les pages HTML sont calculées à partir du nombre de lignes et de colonnes renvoyées pour le rapport.
- Le format de sortie Excel utilise des sauts de page conditionnels (ou sauts de page logiques) qui sont calculés au moment de l'exécution. Les pages Excel sont calculées en tant que feuilles de calcul d'un même classeur. Si un classeur contient quatre feuilles de calcul, chaque feuille est considérée comme une page unique.
- Les extensions de rendu PDF et Image sont des formats orientés page. Par conséquent, vous pouvez définir des propriétés pour contrôler avec précision les sauts de page des rapports visualisés dans les formats PDF ou Image (TIFF).

Les formats de sortie HTML et Excel ne gèrent pas les pages physiques. En outre, les rapports HTML peuvent être interactifs ; cela signifie que les actions réalisées par l'utilisateur dans un rapport peuvent déclencher un traitement supplémentaire qui provoque l'extension du rapport horizontalement ou verticalement afin qu'il puisse accueillir du contenu additionnel. Vous ne pouvez pas contrôler avec précision la pagination des rapports visualisés dans ce format s'ils contiennent des fonctionnalités interactives.

1. Utilisation de sauts de page pour améliorer les performances du traitement des rapports

Vous pouvez utiliser des sauts de page pour améliorer les performances des rapports volumineux pendant les opérations de rendu et de visualisation. Selon la façon dont le rapport est visualisé et selon que le contrôle est configuré ou non pour un traitement asynchrone, le contrôle affiche la page du rapport dans la visionneuse tandis que les autres pages sont restituées en arrière-plan. Cela permet à un utilisateur de commencer à visualiser les pages initiales du rapport en attendant que d'autres pages soient disponibles.

La sortie HTML et Excel affiche un rapport sur une seule page s'il n'y a pas de saut de page. Si vous définissez les propriétés **InteractiveHeight** et **InteractiveWidth**, les rapports restitués par les formats

de sortie HTML et Excel comportent des sauts de page conditionnels. Les sauts de page conditionnels sont placés sur une page à partir d'une taille de page estimée, ce qui rend la taille des rapports moins exacte que celle des rapports générés par un format de sortie qui prend en charge la taille de page. Les sauts de page conditionnels sont calculés au moment de l'exécution par le contrôle. Bien que cela ne soit pas recommandé, vous pouvez désactiver les sauts de page conditionnels en attribuant la valeur 0 à la propriété **InteractiveHeight**.

2. Spécification des sauts de page et des tailles de page

Les sauts de page déterminent l'adaptation du contenu à une page de rapport. Vous pouvez définir des sauts de page avant ou après des éléments de rapport à l'aide des propriétés de ces éléments.

Vous pouvez ajouter des sauts de page au début ou à la fin d'un rectangle, d'une table, d'une matrice, d'une liste, d'un graphique ou d'un groupe. Par défaut, les éléments de rapport ne comprennent pas de saut de page. Pour ajouter un saut de page au début ou à la fin d'un élément, modifiez la propriété **PageBreakAtEnd** ou **PageBreakAtStart** de ce dernier.

En outre, les sauts de page se produisent automatiquement pour les extensions de rendu orientées page, telles que PDF et Image, qui appliquent une taille de page uniforme à l'ensemble du rapport. Les propriétés suivantes permettent de spécifier des sauts de page basés sur la taille de page :

- Les propriétés **PageHeight** et **PageWidth** sont utilisées par les extensions de rendu PDF et Image pour définir l'apparition régulière de sauts de page en fonction d'une mesure physique.
- Les propriétés **InteractiveHeight** et **InteractiveWidth** sont équivalentes aux propriétés **PageHeight** et **PageWidth**, mais concernent l'extension de rendu HTML. Étant donné que le format de sortie HTML redimensionne dynamiquement un rapport pour pouvoir prendre en charge les fonctionnalités d'exploration, d'extraction, d'affichage et de masquage, le serveur de rapports utilise différentes propriétés pour gérer la pagination des pages dynamiques.

Remarque

La largeur d'un rapport peut être supérieure à la largeur de la page. Si un rapport plus large que le format de papier spécifié est généré à l'aide d'une extension de rendu prenant en charge les formats de papier, le rapport résultant peut s'étaler horizontalement sur plusieurs pages. Si vous avez conçu un rapport de sorte qu'il soit affiché sur une seule page alors que son rendu occupe plusieurs pages, vérifiez si sa largeur n'est pas supérieure à celle du format de papier.

F. Création de sources de données pour un rapport

Le mode de traitement d'un contrôle ReportViewer détermine le type de données que vous pouvez utiliser derrière un rapport.

Dans un mode de traitement distant, où les rapports sont traités sur un serveur de rapports SQL Server 2005, des extensions pour le traitement des données sur le serveur de rapports déterminent les types de sources de données que vous pouvez utiliser. Des extensions pour le traitement des données par défaut sont disponibles pour SQL Server, Analysis Services, SQL Server Integration Services, Oracle, SAP NetWeaver® Business Intelligence, XML, ODBC et OLE DB. Des connexions aux sources de données et aux requêtes qui extraient des données pour un rapport sont spécifiées dans une définition de rapport lors de la conception, quand vous créez le rapport dans Business Intelligence Development Studio (il s'agit de l'outil de création de rapport qui est inclus dans SQL Server 2005). Les champs qui sont retournés par la requête peuvent être utilisés dans la mise en page du rapport lorsque vous les faites glisser vers l'aire de conception du rapport. Pour plus d'informations sur l'utilisation de données dans un rapport de serveur, consultez Connecting to a Data Source dans la documentation en ligne de SQL Server.

Dans un mode de traitement local, où les rapports sont traités par le contrôle au sein de votre application, vous pouvez spécifier des sources de données et des champs Visual Studio pour définir les données derrière le rapport. Les connexions à la source de données et aux requêtes qui extraient des données pour un rapport sont définies dans votre projet d'application au moment de la conception. Un rapport client peut utiliser des données de n'importe quelle source à condition qu'elles puissent être fournies en tant que DataTable ADO.NET ou collection énumérable d'objets d'entreprise. La table de données où l'objet d'entreprise retourne une liste de champs qui peuvent être utilisés par le rapport. Les champs contiennent un pointeur vers un champ de base de données et une propriété de nom. Vous pouvez faire glisser les champs de la fenêtre Source de données sur une disposition de rapport.

1. Liaison des données dans un rapport

Les définitions de rapport utilisent des *régions de données* pour lier des données au rapport. Vous pouvez choisir parmi plusieurs régions de données pour prendre en charge différentes structures de données dans votre rapport. La table, la matrice, le graphique et la liste constituent des exemples de région de données que vous pouvez utiliser dans une définition de rapport. Pour lier des données dans un rapport, vous devez procéder comme suit :

- Définissez des sources de données et des datasets à utiliser dans le rapport (vous pouvez utiliser les informations et les liens fournis dans cette rubrique pour en savoir plus). La fenêtre Source de données affiche les sources de données et les champs que vous pouvez utiliser.
- Faites glisser les régions de données que vous souhaitez utiliser à partir de la boîte à outils du rapport sur l'aire de conception.
- Vous pouvez faire glisser les champs de la fenêtre Source de données sur une région de données.

Chaque région de données peut utiliser des champs d'un dataset unique. Ni le contrôle ReportViewer, ni le serveur de rapports ne peut effectuer des jointures entre différents jeux de données. Par exemple, si vous utilisez des objets d'entreprise, la jointure doit être effectuée à l'intérieur de l'objet d'entreprise. Pour plus d'informations sur les régions de données, consultez Ajout de régions de données à un rapport (Concepteur de rapports Visual Studio).

2. Comment ajouter des sources de données pour des rapports ReportViewer

Les approches suivantes sont recommandées pour l'établissement d'une source de données pouvant être utilisée dans une définition de rapport client.

- Ajouter une source de données au projet d'application et configurer la connexion des données aux données sous-jacentes. Pour une banque de données sous-jacente, sélectionnez les données spécifiques que vous souhaitez utiliser. Les données que vous souhaitez utiliser dans le rapport doivent être disponibles dans la fenêtre Sources de données. Pour une table de données, la source de données montre les noms de colonnes retournés à partir d'une requête. Pour une collection énumérable d'objets d'entreprise, la source de données montre une liste de propriétés publiques de types de données simples exposés par l'objet de classe.
- Pour créer une table de données, ajoutez un dataset au projet et utilisez l'Assistant TableAdapter pour configurer la table de données. L'Assistant TableAdapter fournit un générateur de requête et une fonction d'aperçu de données afin que vous puissiez vérifier immédiatement les résultats de la requête.
- Pour créer des objets d'entreprise, utilisez la commande **Ajouter un nouvel élément** du menu **Projet** et sélectionnez l'objet de la classe. Fournissez des implémentations qui exposent des propriétés publiques de types de données simples en vue d'une utilisation au moment de la conception. Fournissez des méthodes pour retourner des collections énumérables des propriétés en vue d'une utilisation au moment de l'exécution.

Une fois qu'une source de données a été configurée, les données pouvant être liées apparaissent dans une hiérarchie dans la fenêtre Source de données. Pour relier des données à un rapport, faites glisser les

nœuds de la hiérarchie étendue sur des zones de texte ou des régions de données dans une mise en page de rapport. Si vous modifiez ultérieurement les datasets après la définition du rapport, vous devez mettre à jour les liaisons de données dans le rapport et le contrôle. Pour plus d'informations, consultez Mise à jour et reliaison des références de source de données (Concepteur de rapports Visual Studio).

Vous pouvez avoir n'importe quel nombre de sources de données dans votre projet. Pour afficher la liste de sources de données actuellement employées par un rapport, choisissez **Sources de données** dans le menu Rapport. Pour prévisualiser le rapport et ses données dans votre application, vous devez construire ou déployer l'application pour vérifier que le rapport contient les données attendues. Pour plus d'informations sur la configuration du contrôle et la définition de la mise en page du rapport, consultez Configuration de ReportViewer pour le traitement local et Création de fichiers de définition de rapport client (.rdlc).

3. Utilisation d'objets d'entreprise

Un objet d'entreprise est un objet d'application qui expose les propriétés publiques de types de données simples. Les exemples d'objets d'entreprise peuvent inclure une collection d'objets de client provenant d'une banque de données sous-jacentes, un groupe d'objets de titre d'ouvrage créés par votre application ou une liste d'objets de canal que vous créez à partir d'une source RSS XML.

Pour être accessible en tant que source de données, la collection doit prendre en charge IEnumerable. Les données pouvant être liées apparaissent dans une vue hiérarchique dans la fenêtre Sources de données. Pour une collection énumérable d'objets d'entreprise, la source de données montre une liste de propriétés publiques de types de données simples exposés par l'objet de classe. Vous pouvez les lier à des régions de données et des zones de texte dans une définition de rapport.

Pour fournir une collection d'objets d'entreprise, vous pouvez ajouter une classe et une référence à l'assembly pour une classe à votre projet.

4. Utilisation de tables de données

Un **DataTable**, en tant que partie intégrante d'un DataSet ADO.NET, peut être facilement ajouté à votre projet à partir du menu Projet en sélectionnant **Ajouter un nouvel élément**. Pour configurer la connexion de données, démarrez l'Assistant Configuration de TableAdapter. Dans le menu Données, sélectionnez Ajouter, puis sélectionnez TableAdapter. Vous pouvez également définir une requête pour limiter votre jeu de résultats. À la fin de l'Assistant, les colonnes du **DataTable** sont accessibles par le biais de la fenêtre Sources de données afin que vous puissiez les lier à des régions de données et des zones de texte dans une définition de rapport client.

Chapitre 7 : Empaquetage et déploiement

Le déploiement est le processus selon lequel une application ou un composant fini est distribué en vue de son installation sur d'autres ordinateurs. Dans Visual Studio 2005, vous pouvez déployer des applications ou des composants à l'aide du déploiement ClickOnce ou de la technologie de déploiement de Windows Installer

I. Choix d'une stratégie de déploiement

Visual Studio fournit deux stratégies différentes pour le déploiement d'applications Windows : la publication d'une application à l'aide de la technologie ClickOnce ou le déploiement avec un programme d'installation traditionnel à l'aide de la technologie Windows Installer. Avec le déploiement ClickOnce, vous publiez l'application à un emplacement centralisé et l'utilisateur l'installe ou l'exécute à partir de cet emplacement. Avec le déploiement Windows Installer, vous empaquetez l'application dans un fichier setup.exe et distribuez ce fichier aux utilisateurs ; ceux-ci exécutent le fichier Setup.exe pour installer l'application.

Plusieurs facteurs sont à prendre en considération lorsque vous choisissez une stratégie de déploiement : le type d'application, le type et l'emplacement des utilisateurs, la fréquence des mises à jour de l'application, ainsi que les conditions requises pour l'installation.

Dans la plupart des cas, le déploiement ClickOnce procure à l'utilisateur final une plus grande facilité d'installation et exige moins d'efforts de la part du développeur. Toutefois, le déploiement Windows Installer peut être nécessaire dans certains cas.

Remarque :

Les outils de déploiement fournis dans Visual Studio sont conçus pour répondre aux besoins de déploiement standard de l'entreprise ; ils ne couvrent pas chaque scénario de déploiement possible. Pour les scénarios de déploiement plus avancés, il se peut que vous deviez envisager l'utilisation d'un outil de déploiement tiers ou d'un outil de distribution logicielle, tel que SMS (Systems Management Server).

II. Déploiement ClickOnce

Le déploiement ClickOnce vous permet de déployer des applications Windows et des applications console à mise à jour automatique qui peuvent être installées, mises à jour et exécutées à partir d'un site Web. Pour plus d'informations, consultez Déploiement ClickOnce.

Le déploiement ClickOnce a été amélioré de plusieurs façons :

- ClickOnce prend en charge le déploiement d'applications de navigateur Web WPF. Comme elles sont hébergées dans un navigateur Web, ces applications nécessitent un déploiement et des paramètres de sécurité spéciaux. Lorsque vous générez et déployez ces applications, Visual Studio fournit l'interface utilisateur appropriée et les valeurs par défaut dans l'IDE. Pour plus d'informations, consultez Page Publier, Concepteur de projets.
- ClickOnce permet aux éditeurs de logiciels indépendants de signer à nouveau le manifeste de l'application

III. Déploiement de Windows Installer

Le déploiement de Windows Installer vous permet de créer un package de programme d'installation à distribuer aux utilisateurs ; l'utilisateur exécute le fichier d'installation et suit les étapes d'un Assistant pour installer l'application. Pour cela, vous ajoutez un projet d'installation à votre solution. Une fois

généralisé, le projet crée un fichier d'installation distribué aux utilisateurs ; l'utilisateur exécute le fichier d'installation et suit les étapes d'un Assistant pour installer l'application.

Les projets de déploiement permettent de spécifier où et comment votre solution sera déployée à la fois pendant et après le développement. Une solution peut contenir plusieurs projets de déploiement qui peuvent varier selon la configuration.

Remarque :

Selon vos paramètres actifs ou votre édition, les boîtes de dialogue et les commandes de menu que vous voyez peuvent différer de celles qui sont décrites dans l'aide. Pour modifier vos paramètres, choisissez **Importation et exportation de paramètres** dans le menu **Outils**. Pour plus d'informations, consultez Paramètres Visual Studio.

Pour créer un projet de déploiement

1. Dans le menu **Fichier**, pointez sur **Ajouter**, puis cliquez sur **Nouveau projet**.
2. Dans le volet **Types de projets** de la boîte de dialogue **Ajouter un nouveau projet**, ouvrez le nœud **Autres types de projets** et sélectionnez **Projets d'installation et de déploiement**.
3. Dans le volet **Modèles**, sélectionnez le type de projet de déploiement à créer.

Pour ajouter un projet de déploiement existant à une solution

1. Dans le menu **Fichier**, pointez sur **Ajouter**, puis cliquez sur **Projet existant**.
2. Dans la boîte de dialogue **Ajouter un projet existant**, recherchez l'emplacement du projet de déploiement, puis cliquez sur **Ouvrir**.

Conseil :

Pour ne voir que des projets de déploiement, dans la zone **Types de fichiers**, sélectionnez **Projets d'installation et de déploiement**.

Pour créer un programme d'installation, commencez par spécifier ce qu'il faut y inclure et où effectuer l'installation sur l'ordinateur cible. Pour cela, ajoutez des éléments au projet de déploiement.

Les types d'éléments pouvant être ajoutés à un projet de déploiement incluent les sorties de projet, les fichiers, les modules de fusion et les assemblies.

Remarque :

Selon vos paramètres actifs ou votre édition, les boîtes de dialogue et les commandes de menu que vous voyez peuvent différer de celles qui sont décrites dans l'aide. Pour modifier vos paramètres, choisissez **Importation et exportation de paramètres** dans le menu **Outils**. Pour plus d'informations, consultez Paramètres Visual Studio.

Pour ajouter une sortie du projet ou un fichier à un projet de déploiement

1. Ouvrez l'**Éditeur du système de fichiers**.
2. Sélectionnez un dossier sur l'ordinateur cible où l'élément sera installé.
3. Dans le menu **Action**, pointez sur **Ajouter**, puis cliquez sur **Sortie du projet** ou sur **Fichier**. Dans la boîte de dialogue obtenue, sélectionnez l'élément à ajouter.

Remarque :

Pour ajouter des éléments à un projet de déploiement, vous pouvez aussi cliquer avec le bouton droit sur le nœud du projet dans l'**Explorateur de solutions**. Tous les éléments ajoutés de cette façon seront placés dans le dossier par défaut. Pour les applications standard, c'est le dossier **Application** ; pour les applications Web, c'est le dossier **Application Web**. Vous pouvez ensuite déplacer les éléments vers un autre dossier.

Pour définir des propriétés dépendantes de la configuration

1. Dans l'**Explorateur de solutions**, sélectionnez le nœud du projet d'installation.
2. Dans le menu **Affichage**, choisissez **Pages de propriétés**.
3. Dans la liste **Configuration**, sélectionnez une configuration.

Conseil :

Si la plupart des propriétés de déploiement sont communes à toutes les configurations, choisissez **Toutes les configurations** pour éviter de répéter le travail de sélection déjà effectué. Une fois les propriétés communes définies, choisissez une autre configuration et définissez les propriétés qui lui sont propres.

4. Sélectionnez une catégorie dans la liste des catégories. Les propriétés de la catégorie sélectionnée s'affichent.