



Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Microsoft Chart Controls

Version 1.0



James RAVAILLE

<http://blogs.dotnet-france.com/jamesr>

Sommaire

1	Introduction.....	3
1.1	Présentation	3
1.2	Installation.....	3
1.3	Pré-requis	3
1.4	Terminologie.....	3
2	Présentation du contrôle <i>Microsoft Chart</i>	5
2.1	Dans les projets Windows Forms	5
2.2	Les types de graphique.....	5
2.3	Les régions.....	6
2.4	Les séries	8
2.5	Les légendes	9
2.6	Les annotations	9
2.7	Les titres	12
2.8	Design du graphique.....	14
2.8.1	Choix de la palette de couleurs	14
2.8.2	Modification de la couleur de fond	14
2.8.3	Modification des bordures	15
2.9	Définition d'une source de données	16
3	Mise en œuvre du contrôle	17
3.1	Présentation	17
3.2	Définition de la source de données.....	17
3.3	Visualiser les types de graphique	20
3.3.1	Affichage de la liste des types de graphique et de la source de données	21
3.3.2	Configuration du contrôle <i>Chart</i>	23
3.4	Suivi mensuel d'activité.....	23
3.5	Définition de la source de données.....	24
3.5.1	Configuration du contrôle <i>Chart</i>	24
4	Conclusion	30

1 Introduction

1.1 Présentation

En 2007, Microsoft annonçait l'acquisition des contrôles *Dundas Chart*, permettant de pouvoir créer des graphiques dans *Microsoft Reporting Services* (solution de création et génération de rapports de Microsoft). Fin 2008, Microsoft propose une version de ces contrôles (version 5.5), qu'il est possible d'utiliser dans les applications ASP .NET et Windows Forms. Nous vous proposons dans ce cours, de vous présenter ce contrôle, et de le mettre en œuvre dans une application Windows Forms.

Ce cours est écrit avec Visual Studio 2008 et le Framework .NET 3.5 Service Pack 1.

1.2 Installation

Pour réaliser les exercices de ce cours, vous devez installer les composants suivants :

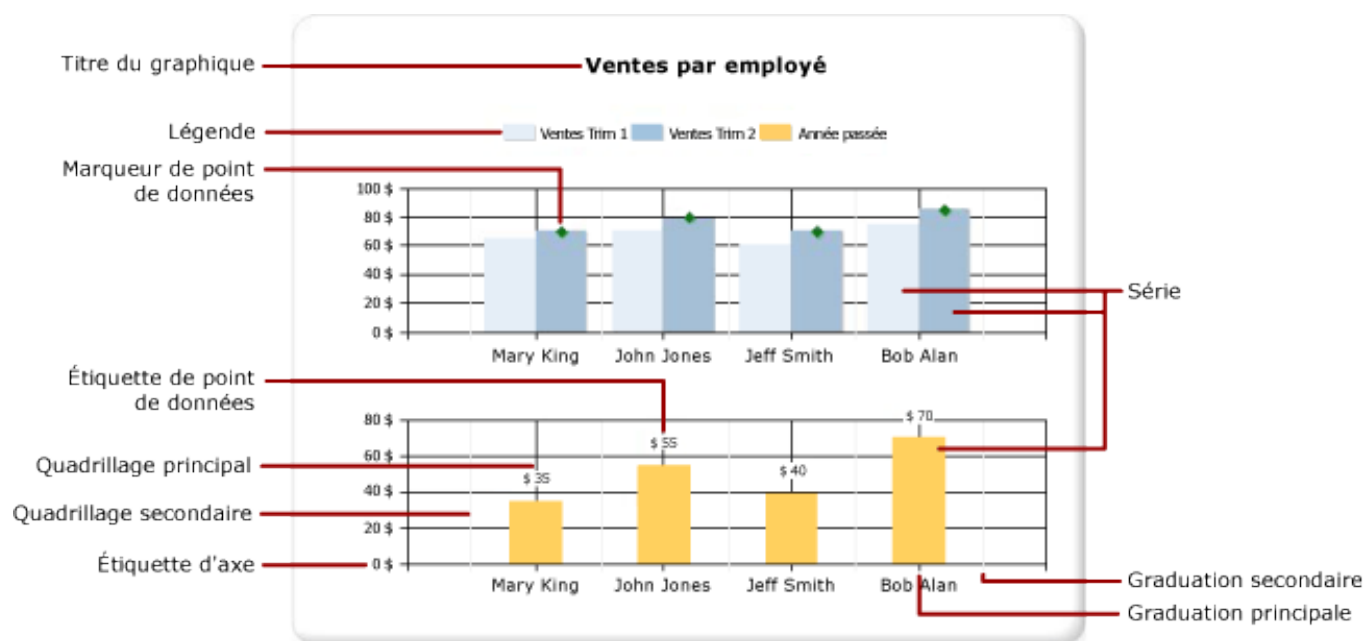
- [Microsoft Chart Controls for Microsoft .NET Framework 3.5](#) : contrôles de génération de graphiques pour les applications Web et Windows.
- [Microsoft Chart Controls Add-on for Microsoft Visual Studio 2008](#) : mise à jour de Visual Studio 2008, ajout de nouveaux contrôles dans la boîte à outils dans les projets Web et Windows. Ce package peut uniquement être installé si *Microsoft Chart Controls for Microsoft .NET Framework 3.5* a été préalablement installé.
- [Microsoft Chart Controls for .NET Framework Documentation](#) (optionnel) : documentation de référence des contrôles.
- [Microsoft Chart Controls for Microsoft .NET Framework 3.5 Language Pack](#) (optionnel) : module linguistique pour les contrôles de génération de graphiques.

1.3 Pré-requis

Dans ce cours, nous développerons une application *Windows Forms* pour montrer comment s'utilise le *Microsoft Charts Controls*. Il est alors fortement conseillé que le développement d'applications Windows Forms vous soit familier. Il est aussi conseillé de maîtriser les bases fondamentales du langage C# ou VB .NET.

1.4 Terminologie

Voici un schéma permettant de nommer l'ensemble des éléments constituant un graphique :

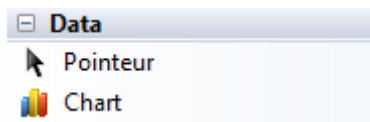


Ce graphique contient deux régions. La première contient deux séries. La seconde n'en contient qu'une seule. Une série de données représente de manière visuelle des données sous différentes formes, appelées types de graphiques.

2 Présentation du contrôle *Microsoft Chart*

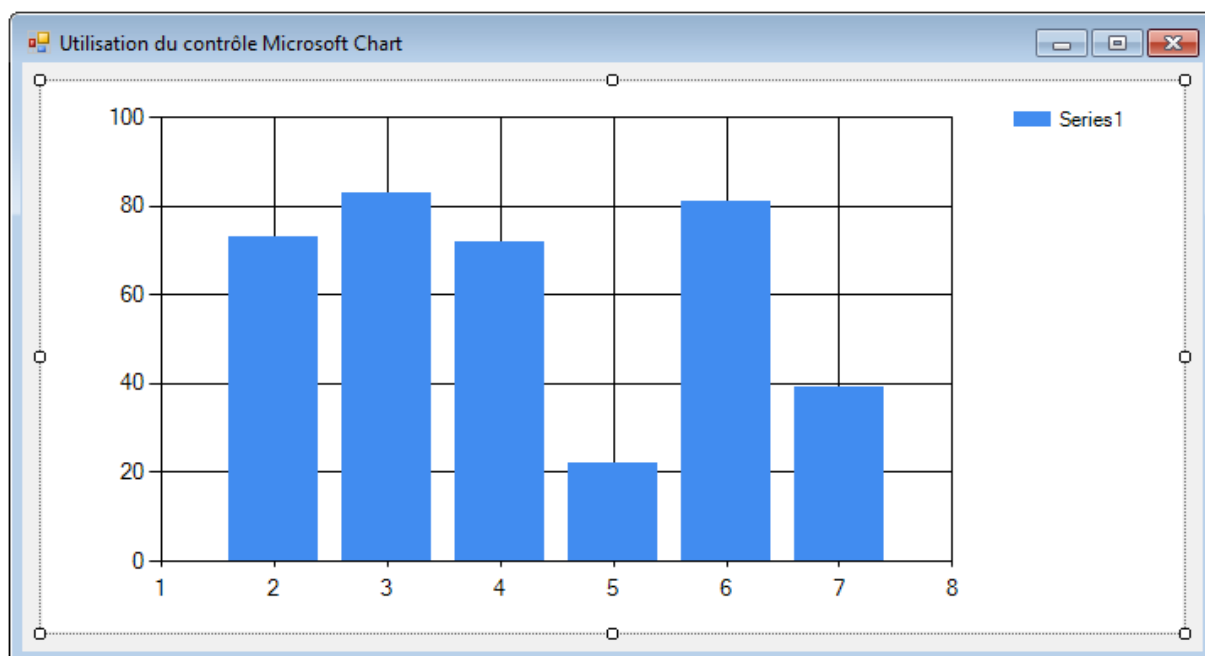
2.1 Dans les projets Windows Forms

Après la création de projets Windows Forms ou ASP .NET (application Web ou site Web), nous retrouvons dans la boîte à outils, un nouvel onglet *nommé Data*, contenant le contrôle *Chart* :











Ce contrôle permet d'afficher un graphique de données, qui peut être de type différent (histogramme, courbe, radar, nuage de point, ...).

Dans un formulaire Windows Forms, nous ajoutons ce contrôle, et nous obtenons l'interface suivante (un graphique factice) :



2.2 Les types de graphique

Le contrôle *Chart* permet d'afficher les graphiques suivants :

Visualisation	Type de graphique
	<i>Point</i>
	<i>FastPoint</i>
	<i>Bubble</i>
	<i>Line</i>
	<i>Spline</i>
	<i>StepLine</i>
	<i>FastLine</i>
	<i>Bar</i>

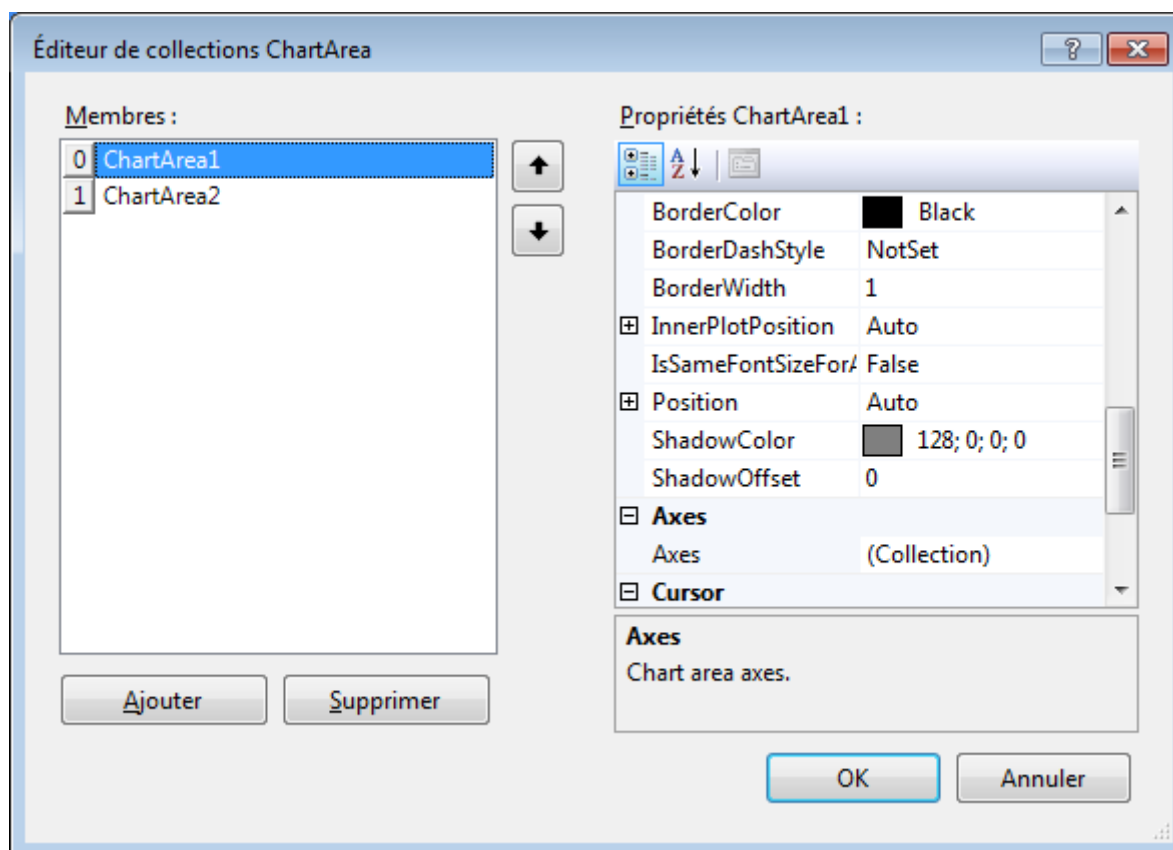


	<i>StackedBar</i>
	<i>StackedBar100</i>
	<i>Column</i>
	<i>StackedColumn</i>
	<i>StackedColumn100</i>
	<i>Area</i>
	<i>SplineArea</i>
	<i>StackedArea</i>
	<i>StackedArea100</i>
	<i>Pie</i>
	<i>Doughnut</i>
	<i>Stock</i>
	<i>Candlestick</i>
	<i>Range</i>
	<i>SplineRange</i>
	<i>RangeBar</i>
	<i>RangeColumn</i>
	<i>Radar</i>
	<i>Polar</i>
	<i>ErrorBar</i>
	<i>BoxPlot</i>
	<i>Renko</i>
	<i>ThreeLineBreak</i>
	<i>Kagi</i>
	<i>PointAndFigure</i>
	<i>Funnel</i>
	<i>Pyramid</i>

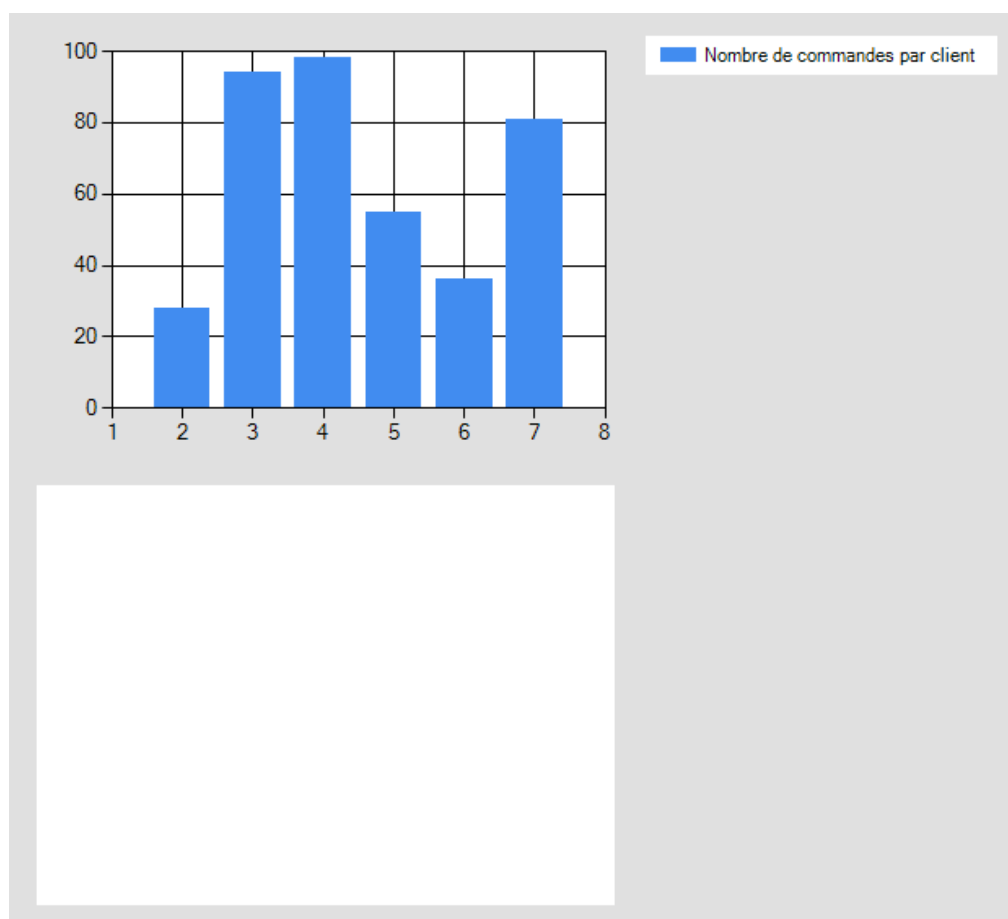
L'énumération `System.Windows.Forms.DataVisualization.Charting.SeriesChartType` permet de définir ces types de graphiques. *Column* est le type de graphique par défaut.

2.3 Les régions

Un contrôle *Chart* possède une région par défaut. Elle peut aussi en posséder plusieurs, définissables au travers de la propriété *ChartAreas*. Cette propriété est configurable au travers d'un assistant :



Chaque région possède des propriétés permettant de définir son comportement et son apparence. Dans l'exemple ci-dessus, nous définissons 2 régions. Nous obtenons alors le contrôle *Chart* suivant :



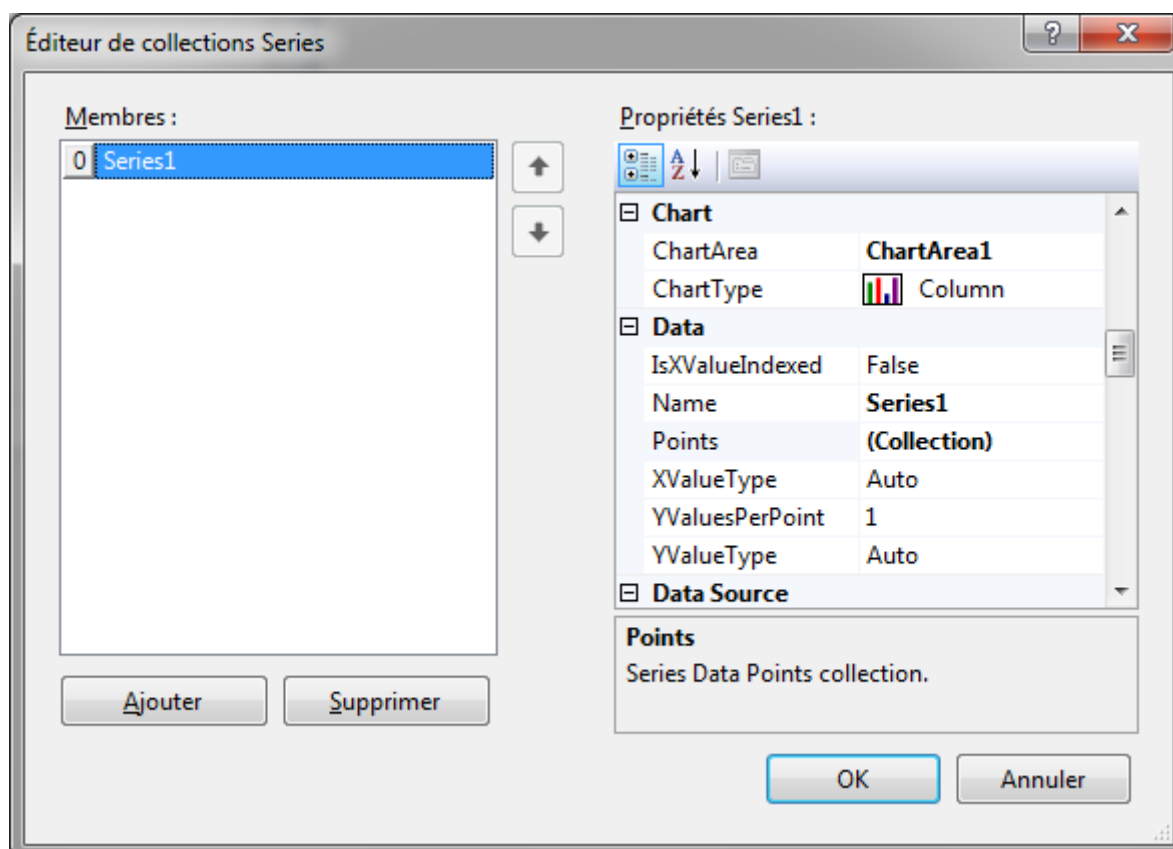
Remarque : la couleur de fond du contrôle *Chart* a été modifiée (couleur grise), afin de pouvoir visualiser les régions ajoutées.

Une région contient une plusieurs séries de données. Elles peuvent avoir des formes différentes (cf : voir les types de graphique disponibles, présentés dans ce cours). Toutefois, les types de graphique des séries devront être compatibles.

2.4 Les séries

Une série est une liste de données utilisées pour concevoir un graphique. Le contrôle *Chart* permet d'afficher une ou plusieurs séries dans une région. Chaque série est alimentée par une source de données.

Les séries d'un contrôle *Chart* peuvent être définies au travers de la propriété *Series*. Ce contrôle propose un assistant permettant de les définir et les configurer :

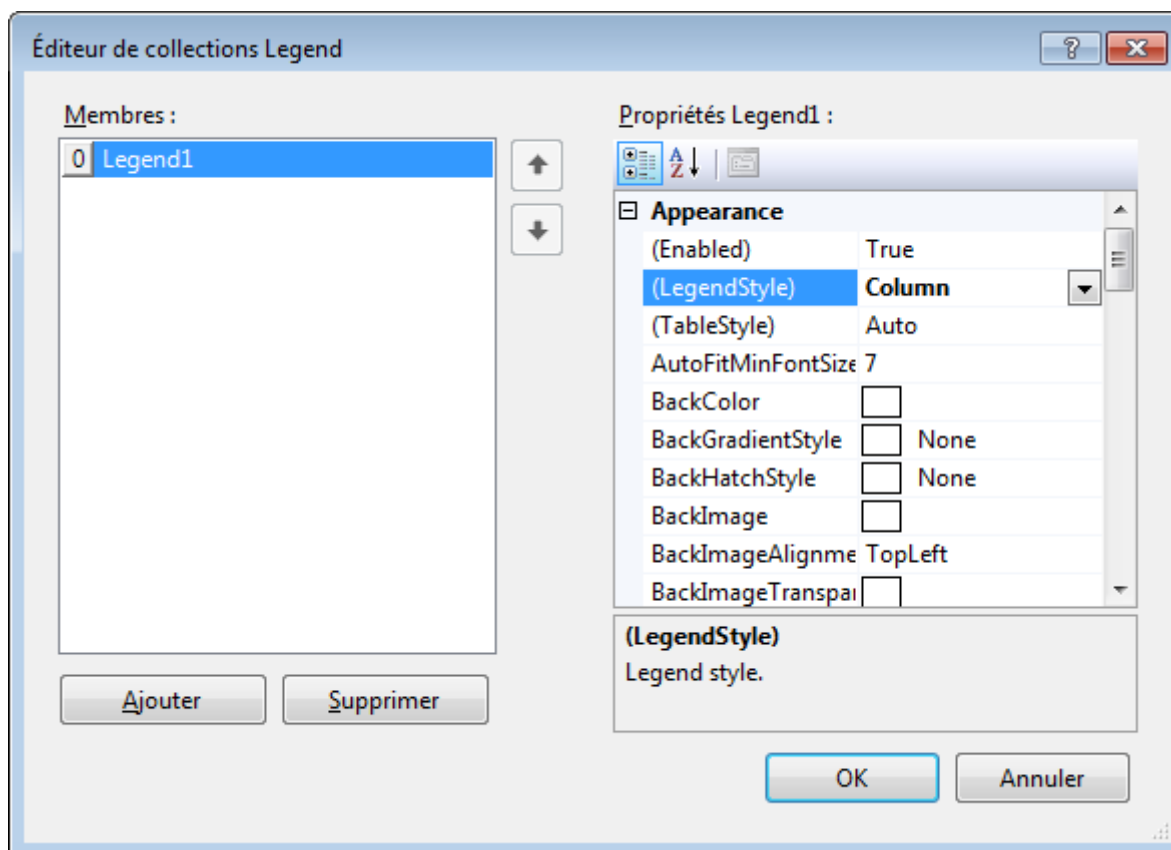


Voici quelques propriétés permettant de configurer une série :

Propriété	Description
<i>XValueMember</i>	Propriété des objets de la source de données utilisée pour l'axe des abscisses.
<i>YValueMembers</i>	Propriété des objets de la source de données utilisée pour l'axe des ordonnées.
<i>ChartArea</i>	Région dans laquelle la série doit être affichée. Une région peut contenir une ou plusieurs séries de données.
<i>ChartType</i>	Type de représentation de données (type de graphique)

2.5 Les légendes

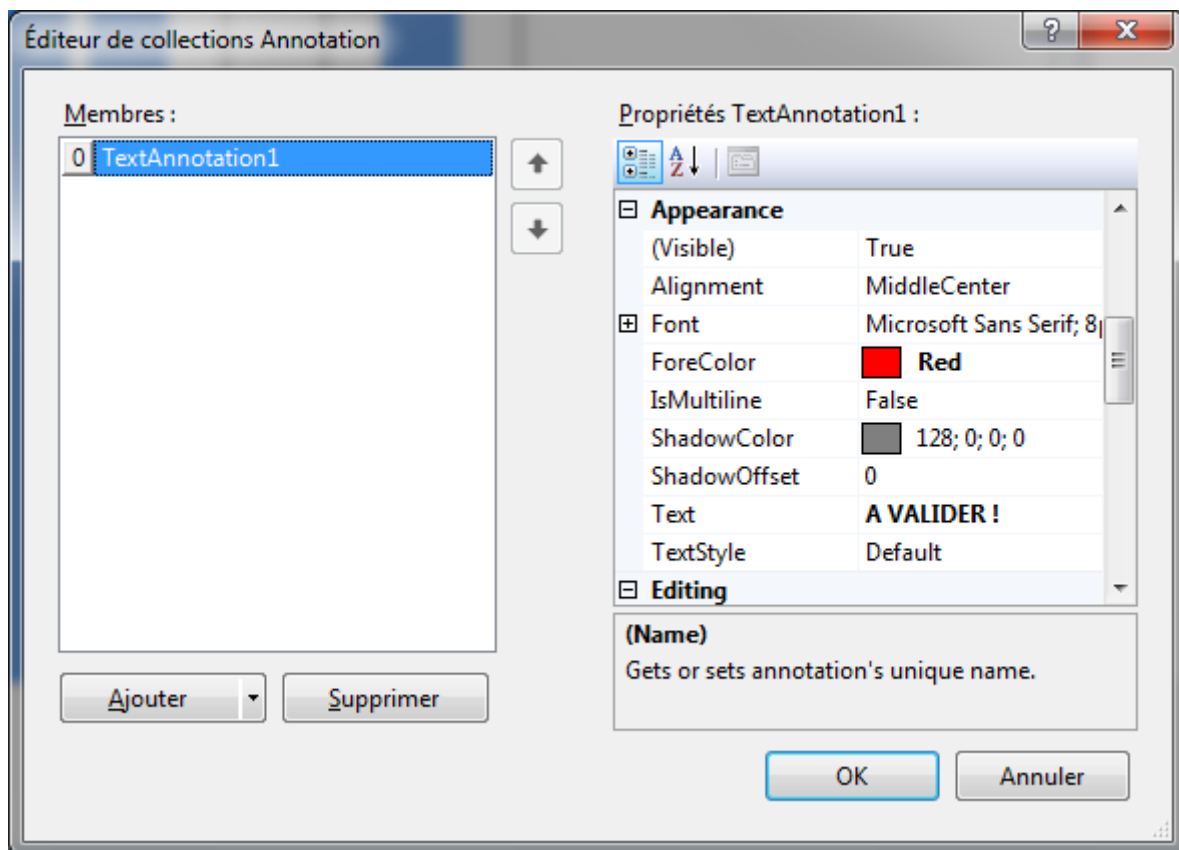
Le contrôle *Chart* possède une propriété *Legends*, permettant de définir des légendes. Les légendes permettent de décrire sous forme d'un libellé, les séries de données. Ce contrôle présente aussi un assistant permettant de les définir et de les configurer :



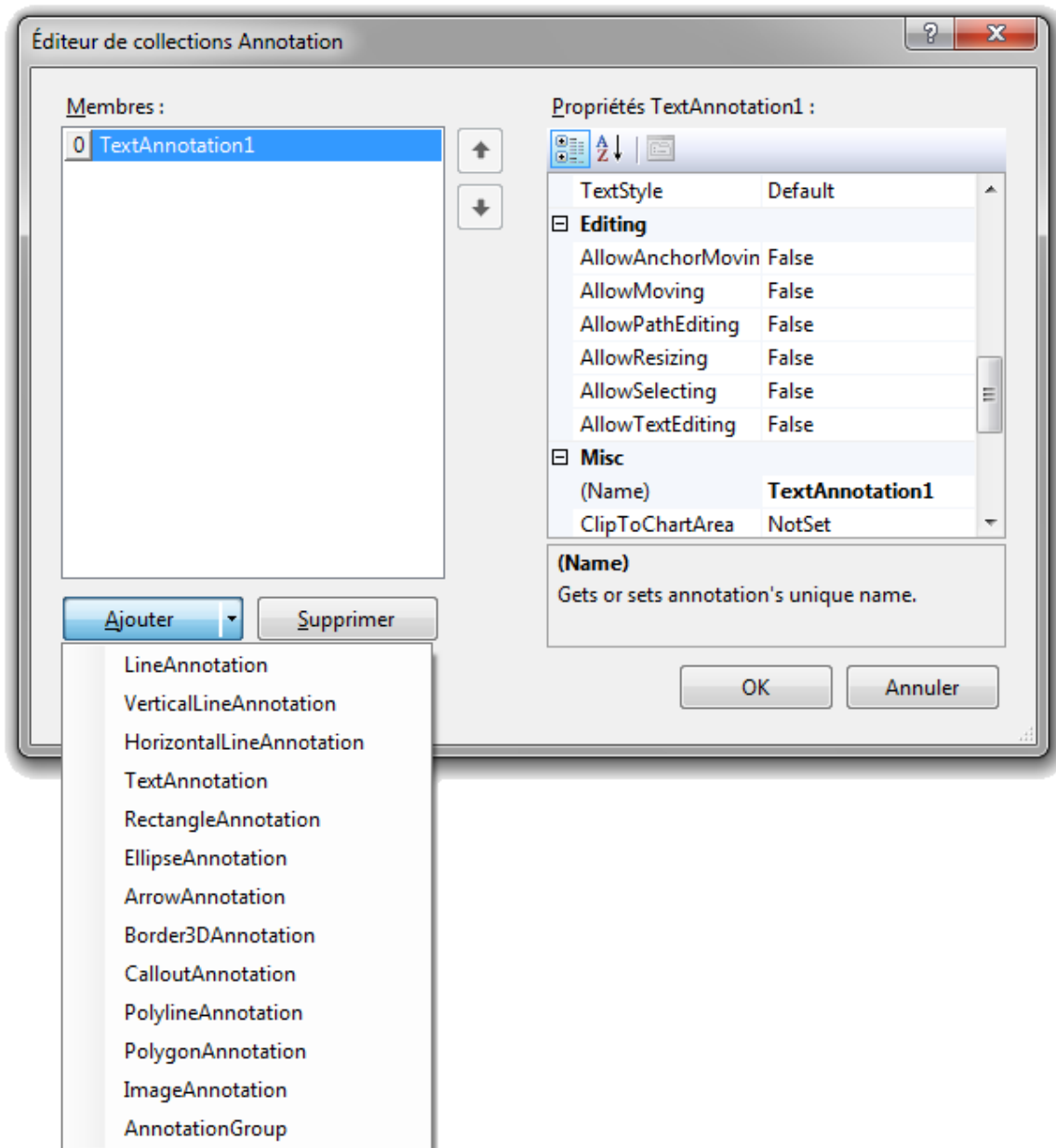
Par défaut, tout contrôle *Chart* possède une légende, affichée dans la région par défaut.

2.6 Les annotations

Il est possible d'ajouter des annotations dans les graphiques. Tout contrôle *Chart* possède une propriété *Annotations* permettant de les définir. Cette propriété est « configurable » au travers d'un assistant :



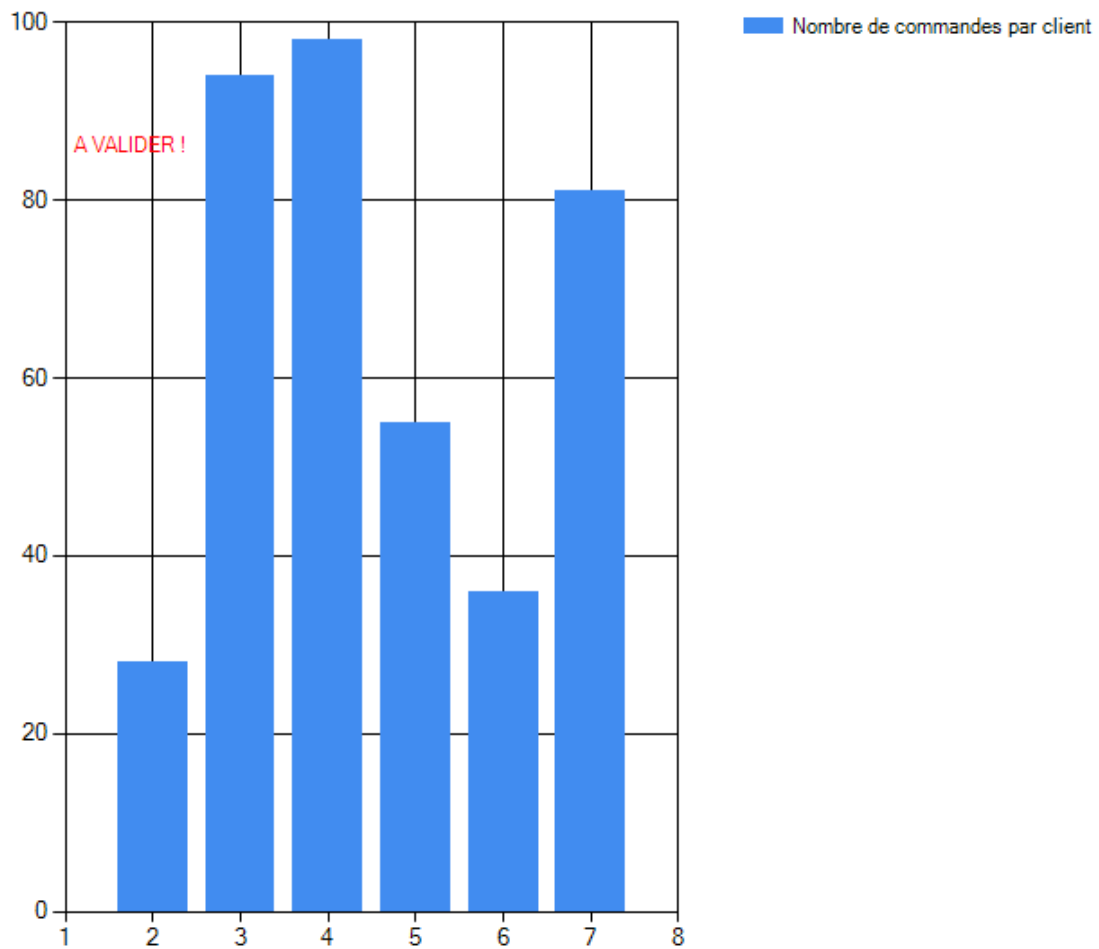
Il existe différents types d'annotations, accessibles au travers de la petite flèche accolée au bouton *Ajouter*, permettant d'afficher différentes formes d'annotations :



Par exemple, nous pouvons ajouter une annotation de type *TextAnnotation*, permettant d'indiquer que les données du graphique sont à valider. Nous modifions alors les propriétés suivantes :

- *Text* en la valorisation avec le texte « A VALIDER ! »
- *ForeColor* à « Red »
- *X* à 10 (position de l'annotation sur l'axe des abscisses)
- *Y* à 15 (position de l'annotation sur l'axe des ordonnées)

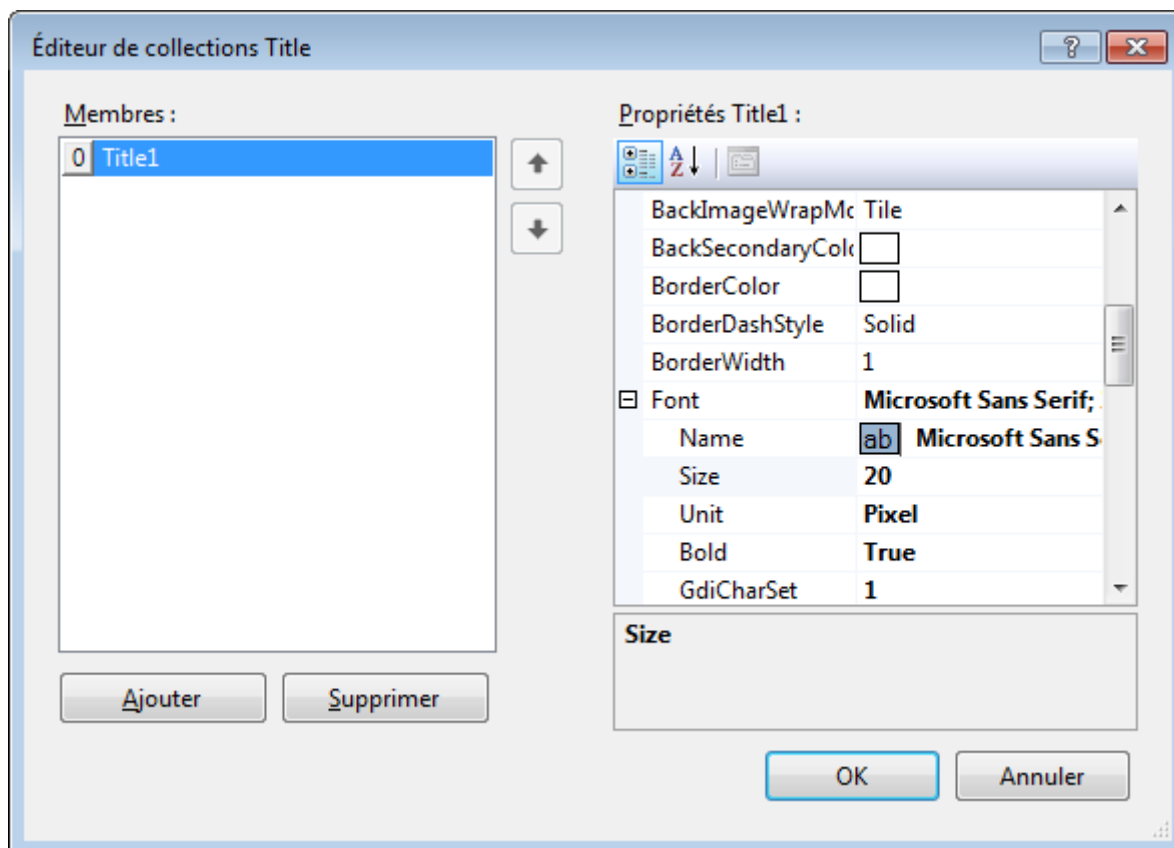
Nous obtenons le résultat suivant :



Pour affecter l'annotation à une région particulière, il suffit de définir la propriété *ClipToChartArea*.

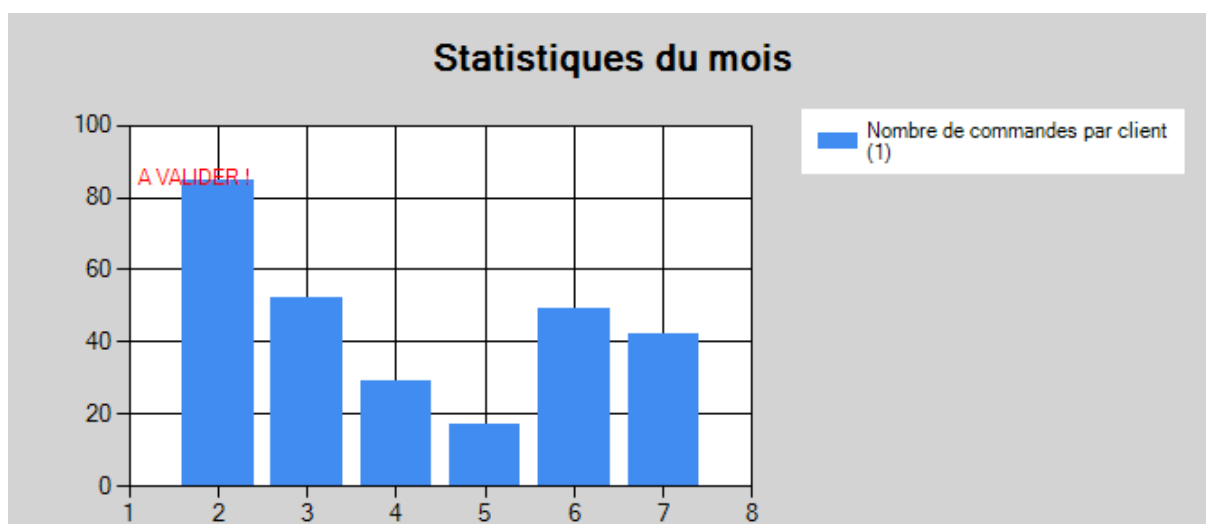
2.7 Les titres

Un contrôle *Chart* possède une collection de titres, définis au travers de la propriété *Titles*. Un assistant permet de définir ces titres :



Les titres sont affichés dans la zone des titres, située par défaut en haut du diagramme. Chaque titre possède des propriétés permettant de définir son apparence et sa position dans la zone de titre ou par rapport à une aire de graphique. La propriété *Text* permet de définir le libellé du titre.

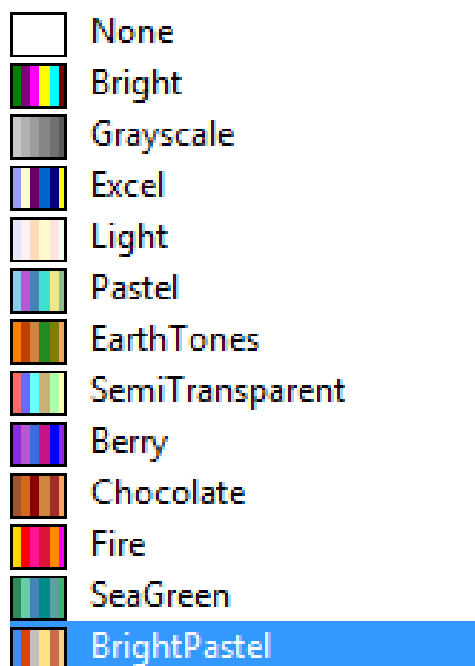
Voici un exemple de titre :



2.8 Design du graphique

2.8.1 Choix de la palette de couleurs

Le contrôle Chart permet au travers de sa propriété *Palette*, de définir la palette de couleurs à utiliser dans le graphique, pour définir les couleurs des graphiques eux-mêmes, pour les titres et légendes... Lorsque plusieurs séries sont ajoutées au graphique, le graphique assigne une couleur à chaque série, dans l'ordre dans lequel les couleurs ont été définies dans la palette. Voici les différentes palettes de couleurs disponibles :

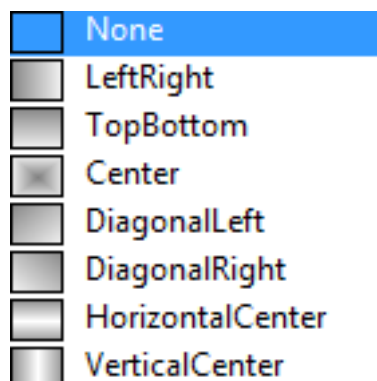


BrightPastel est la palette de couleurs par défaut.

2.8.2 Modification de la couleur de fond

Il est possible de modifier le fond du graphique via l'utilisation de couleurs, en utilisant les propriétés :

- *BackColor* : désignation de la couleur de fond ;
- *BackGradientStyle* : sens du dégradé appliqué au fond (de gauche à droite, de base en haut, en diagonale...). Les deux couleurs utilisées pour le dégradé sont définies par les propriétés *BackColor* et *BackSecondaryColor*. Voici les sens de dégradés possibles :



- *BackImage* : désignation de l'image de fond ; la propriété *BackgroundImageLayout* permet de définir le mode de disposition de l'image :

None

Tile

Center

Stretch

Zoom

Et la propriété *BackImageAlignment* permet de définir la position de l'image :

TopLeft

Top

TopRight

Right

BottomRight

Bottom

BottomLeft

Left

Center

2.8.3 Modification des bordures

Il est possible de définir une bordure pour le graphique, en utilisant les propriétés suivantes :

- *BorderColor* : couleur de la bordure ;
- *BorderlineDashStyle* : style de la bordure. Sa valeur par défaut est *NotSet*, qui indique que le graphique n'a pas de bordure. Voici les valeurs disponibles :

NotSet

Dash

DashDot

DashDotDot

Dot

Solid

- *BorderlineWidth*: épaisseur de la bordure.

2.9 Définition d'une source de données

Le contrôle *Chart* accepte une source de données, contenant les données permettant d'alimenter les séries de données du graphique. Cette source de données peut être une grappe d'objets, une *DataTable*, ...

Pour définir cette source de données, nous valoriserons la propriété *DataSource*. Lors de la modification de la source de données remplaçant une source de données existante, nous devons appliquer la méthode *DataBind()* afin demander au contrôle *Chart* de prendre en compte la nouvelle source de données.

3 Mise en œuvre du contrôle

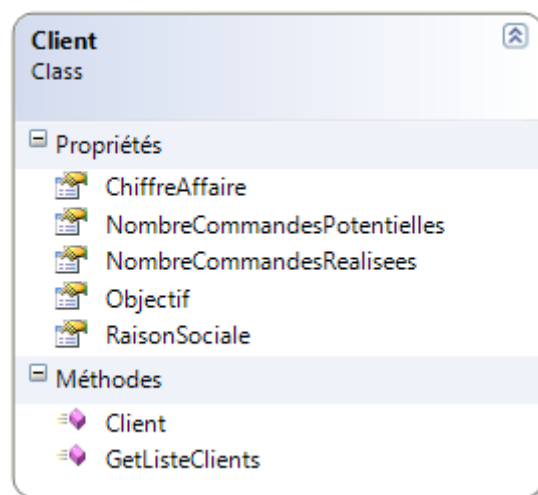
3.1 Présentation

Nous allons mettre en œuvre le contrôle *Chart* dans des formulaires Windows Forms. Nous allons donc créer deux formulaires Windows :

- Un contenant un contrôle *Chart*, qui affichera un seul graphique (avec une série de données). Ce formulaire permettra de visualiser les différents types de graphiques proposés par le contrôle *Chart*.
- Un autre contenant un contrôle *Chart*, avec plusieurs régions et plusieurs séries.

3.2 Définition de la source de données

La source de données de notre application sera une grappe d'objets, créée à partir de la classe suivante :



Un client est caractérisé par :

- Une raison sociale.
- Un chiffre d'affaire.
- Un nombre de commandes potentielles pouvant être réalisées.
- Un nombre de commandes réalisées.
- Un objectif correspondant au nombre de commandes à réaliser.

La méthode *GetListeClients* est une méthode statique permettant d'obtenir une liste de clients, constituant un bouchon de données, qui sera utilisée comme source de données.

Voici l'implémentation de ces classes :



```
// C# .NET

public class Client
{
    public string RaisonSociale {get; set;}
    public double ChiffreAffaire { get; set; }
    public int Objectif { get; set; }
    public int NombreCommandesRealisees { get; set; }
    public int NombreCommandesPotentielles { get; set; }

    public Client(string aRaisonSociale, double aChiffreAffaire, int
aNombreCommandesPotentielles, int aObjectif, int aNombreCommandes)
    {
        this.RaisonSociale = aRaisonSociale;
        this.ChiffreAffaire = aChiffreAffaire;
        this.NombreCommandesPotentielles = aNombreCommandesPotentielles;
        this.Objectif = aObjectif;
        this.NombreCommandesRealisees = aNombreCommandes;
    }

    public static List<Client> GetListeClients()
    {
        List<Client> oListeClients = new List<Client>();

        oListeClients.Add(new Client("Strom Industries", 19380, 12,
25000, 2));
        oListeClients.Add(new Client("ADI Numérique", 13987, 21, 12000,
4));
        oListeClients.Add(new Client("Point Formation", 21622, 8, 20500,
3));

        return oListeClients;
    }
}
```



```
' VB .NET

Public Class Client

    Private _RaisonSociale As String
    Public Property RaisonSociale() As String
        Get
            Return _RaisonSociale
        End Get
        Set(ByVal value As String)
            _RaisonSociale = value
        End Set
    End Property

    Private _ChiffreAffaire As Double
    Public Property ChiffreAffaire() As Double
        Get
            Return _ChiffreAffaire
        End Get
        Set(ByVal value As Double)
            _ChiffreAffaire = value
        End Set
    End Property

    Private _Objectif As Integer
    Public Property Objectif() As Integer
        Get
            Return _Objectif
        End Get
        Set(ByVal value As Integer)
            _Objectif = value
        End Set
    End Property

    Private _NombreCommandesRealisees As Integer
    Public Property NombreCommandesRealisees() As Integer
        Get
            Return _NombreCommandesRealisees
        End Get
        Set(ByVal value As Integer)
            _NombreCommandesRealisees = value
        End Set
    End Property

    Private _NombreCommandesPotentielles As Integer
    Public Property NombreCommandesPotentielles() As Integer
        Get
            Return _NombreCommandesPotentielles
        End Get
        Set(ByVal value As Integer)
            _NombreCommandesPotentielles = value
        End Set
    End Property

End Class
```



```
Public Sub New(ByVal aRaisonSociale As String, ByVal aChiffreAffaire
As Double, ByVal aNombreCommandesPotentielles As Integer, ByVal aObjectif
As Double, ByVal aNombreCommandes As Integer)
    Me.RaisonSociale = aRaisonSociale
    Me.ChiffreAffaire = aChiffreAffaire
    Me.NombreCommandesPotentielles = aNombreCommandesPotentielles
    Me.Objectif = aObjectif
    Me.NombreCommandesRealisees = aNombreCommandes
End Sub

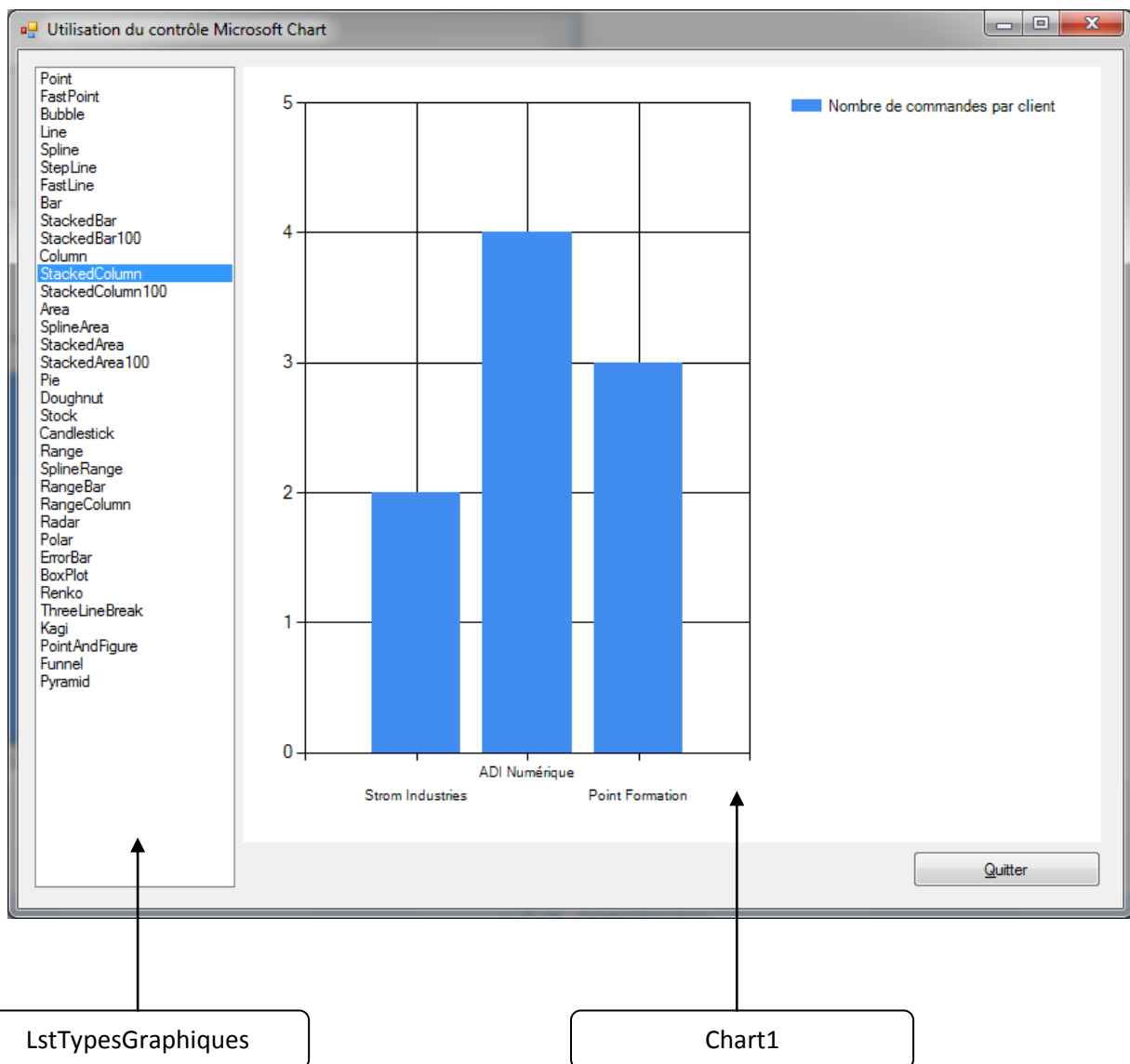
Public Shared Function GetListeClients() As List(Of Client)
    Dim oListeClients As New List(Of Client)()

    oListeClients.Add(New Client("Strom Industries", 19380, 12,
25000, 2))
    oListeClients.Add(New Client("ADI Numérique", 13987, 21, 12000,
4))
    oListeClients.Add(New Client("Point Formation", 21622, 8, 20500,
3))

    Return oListeClients
End Function
End Class
```

3.3 Visualiser les types de graphique

Nous allons réaliser un formulaire permettant de visualiser les types de graphiques. Le design de ce formulaire est le suivant :



3.3.1 Affichage de la liste des types de graphique et de la source de données

Lors du chargement du formulaire :

- Nous affichons la liste des types de graphiques proposés par le contrôle *Chart*. Pour ce faire, nous utilisons dans le formulaire un contrôle de type *BindingSource* nommé *BdsTypesGraphiques*. Nous définissons ce contrôle comme source de données du contrôle *LstTypesGraphiques*, au travers de sa propriété *DataSource*. Enfin, nous obtenons la liste des types de graphique au travers de l'énumération *System.Windows.Forms.DataVisualization.Charting.SeriesChartType*.
- Nous définissons la source de données du graphique avec la liste des clients, au travers de sa propriété *DataSource*.

```
// C#

private void FrmChart_Load(object sender, EventArgs e)
{
    // Alimentation des types de graphiques.
    BdsTypesGraphiques.DataSource =
    Enum.GetValues(typeof(System.Windows.Forms.DataVisualization.Charting.SeriesChartType));

    // Création de la source de données.
    chart1.DataSource = Client.GetListeClients();
}
```

```
' VB .NET

Private Sub FrmChart_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    ' Alimentation des types de graphiques.
    BdsTypesGraphiques.DataSource =
    [Enum].GetValues(GetType(System.Windows.Forms.DataVisualization.Charting.
SeriesChartType))

    ' Création de la source de données.
    chart1.DataSource = Client.GetListeClients()
End Sub
```

Lors de la sélection d'un type de graphique, nous mettons à jour le graphique. Pour ce faire, nous valorisons le type de graphique de la série de données du graphique, avec le type de graphique :

```
// C#

private void BdsTypesGraphiques_CurrentChanged(object sender, EventArgs
e)
{
    chart1.Series[0].ChartType =
    (System.Windows.Forms.DataVisualization.Charting.SeriesChartType)BdsTypes
Graphiques.Current;
}
```

```
' VB .NET

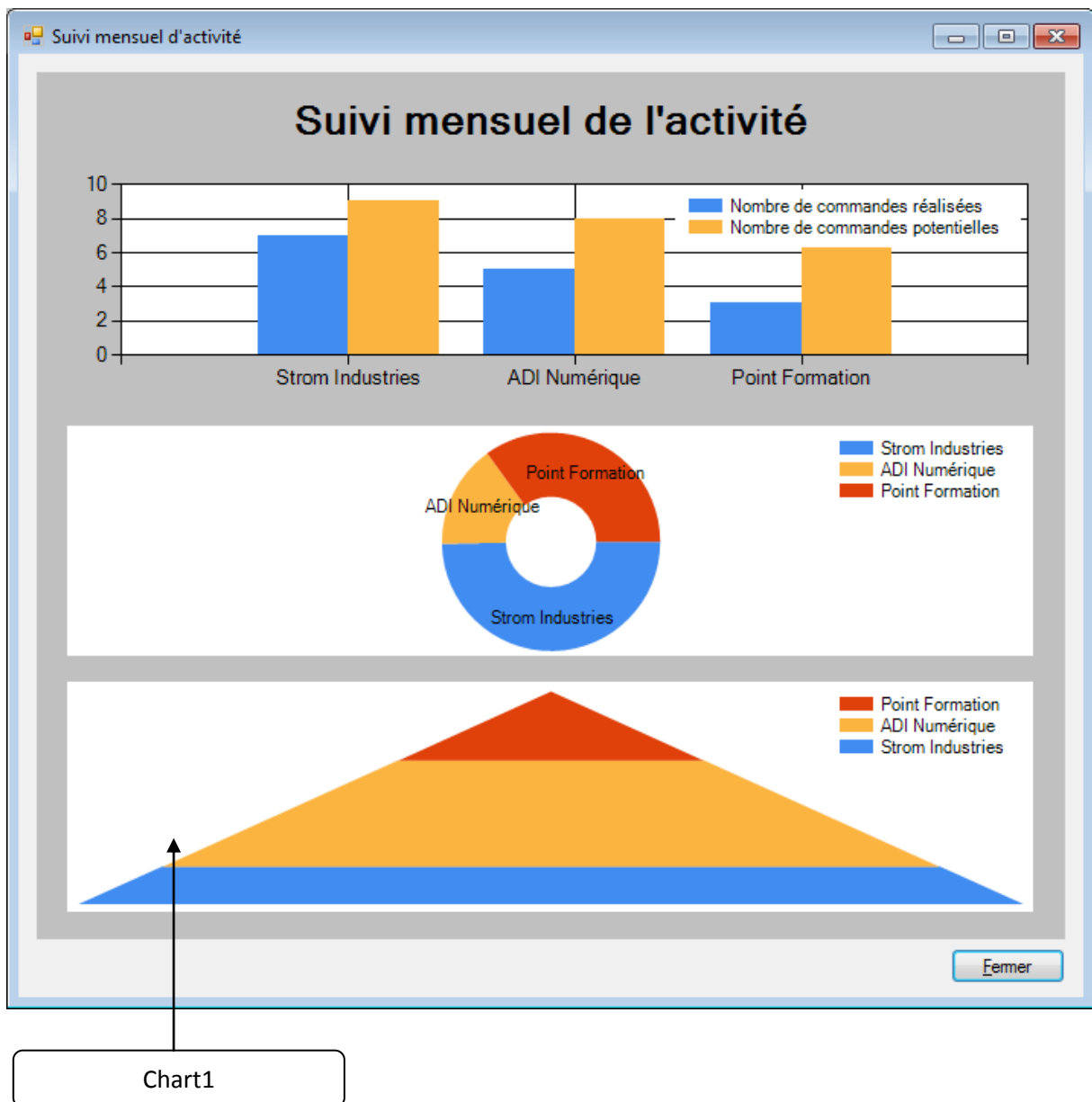
Private Sub BdsTypesGraphiques_CurrentChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
BdsTypesGraphiques.CurrentChanged
    chart1.Series(0).ChartType = CType(BdsTypesGraphiques.Current,
System.Windows.Forms.DataVisualization.Charting.SeriesChartType)
End Sub
```

3.3.2 Configuration du contrôle *Chart*

Par défaut, le contrôle *Chart* possède une région et une série de données, dont le type de graphique est *Column*. De ce fait, aucune configuration n'est à effectuer, si ce n'est de modifier le nom de la série (propriété *Name*) avec le libellé « Nombre de commandes par client ».

3.4 Suivi mensuel d'activité

Nous allons créer un graphique permettant de réaliser un suivi mensuel d'activité :



Ce graphique contient trois régions :

- La première affiche au travers d'histogrammes, le nombre de commandes réalisées par client, avec les objectifs à atteindre.
- La seconde affiche dans un « donut », la proportion des chiffres d'affaire des clients.
- La dernière affiche dans une pyramide le potentiel de commandes par client.

Chacune de ces régions possède sa propre légende.

Le fond du graphique est de couleur grise. Ainsi, nous valorisons la propriété *BackColor* du contrôle *Chart* à *Silver*.

3.5 Définition de la source de données

Lors du chargement du formulaire :

- Nous affichons la liste des types de graphiques proposées par le contrôle *Chart*.
- Nous définissons la source de données du graphique avec la liste des clients.

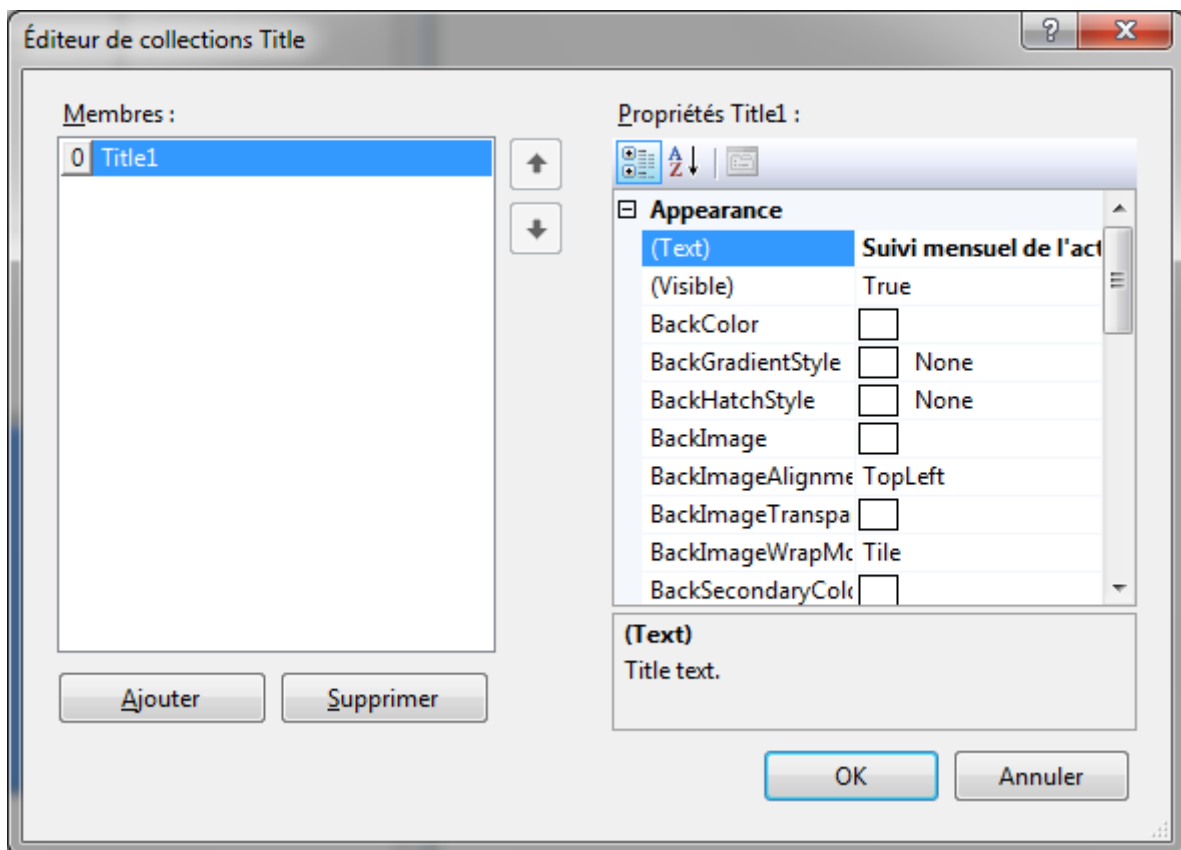
```
// C#  
  
private void FrmSuiviMensuelActivite_Load(object sender, EventArgs e)  
{  
    // Création de la source de données.  
    chart1.DataSource = Client.GetListeClients();  
    chart1.DataBind();  
}
```

```
' VB .NET  
  
Private Sub FrmSuiviMensuelActivite_Load(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles MyBase.Load  
    ' Création de la source de données.  
    chart1.DataSource = Client.GetListeClients()  
    chart1.DataBind()  
End Sub
```

3.5.1 Configuration du contrôle *Chart*

3.5.1.1 Ajout d'un titre

Notre graphique ne contient qu'un seul titre. Pour l'ajouter, nous utilisons l'assistant de création de titres accessibles au travers de la propriété *Titles* du contrôle *Chart* :



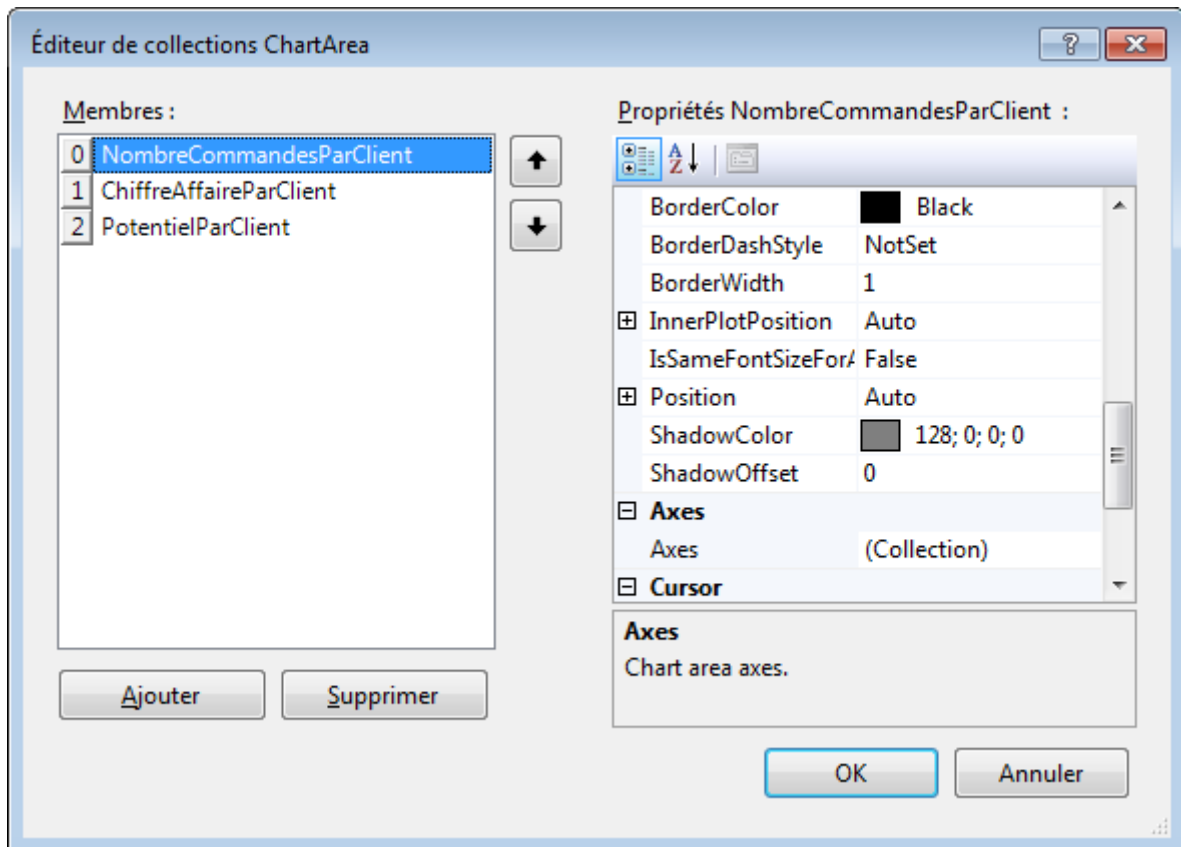
Nous valorisons les propriétés suivantes :

- *Text* avec le libellé « Suivi mensuel de l'activité »
- *Size* de la propriété de regroupement *Font* à 20.
- *Bold* de la propriété de regroupement *Font* à *True*.

3.5.1.2 Ajout des régions

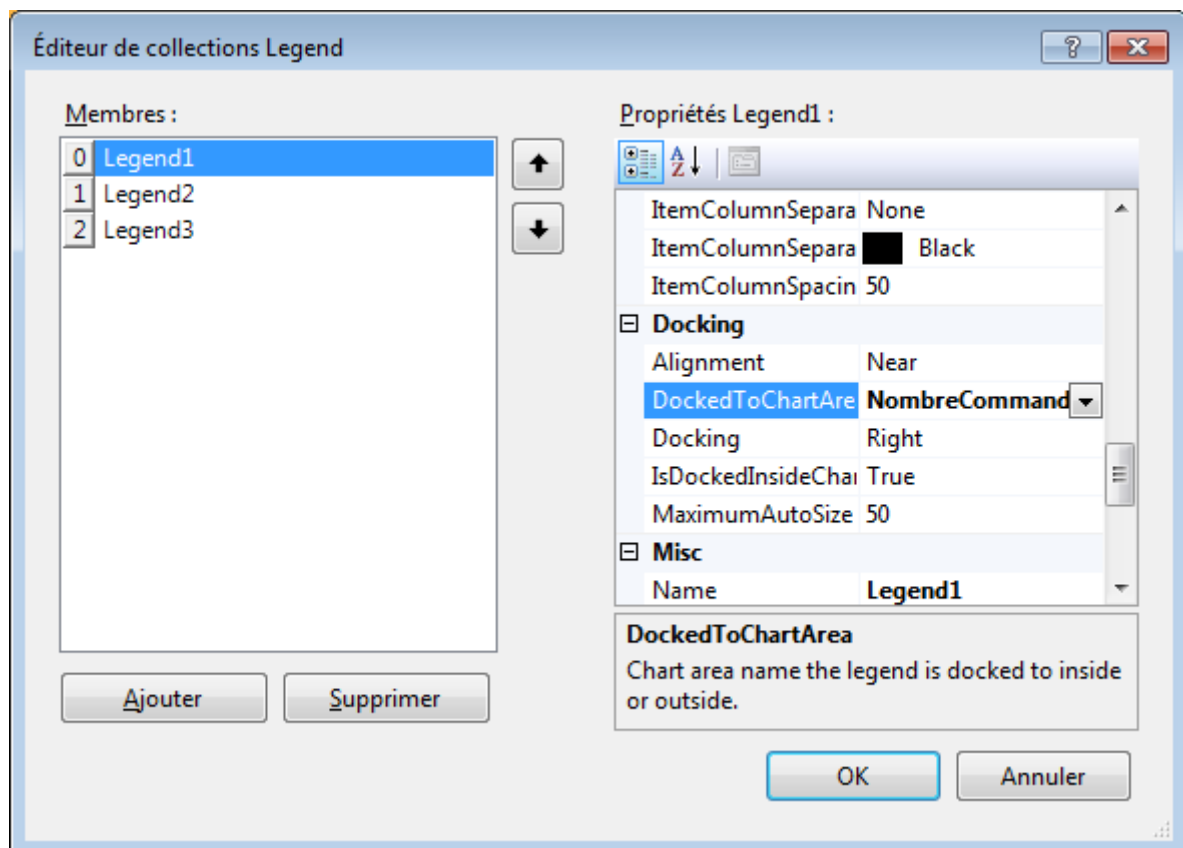
Notre graphique contient trois régions. Pour les définir, nous utilisons l'assistant accessible via la propriété *ChartAreas* du contrôle *Chart*. Nous ajoutons trois régions, et nous leur donnons les noms respectifs suivants :

- *NombreCommandesParClient* ;
- *ChiffreAffaireParClient* ;
- *PotentielParClient*.



3.5.1.3 Répartition des légendes

Chacune des régions devra posséder sa propre légende. Au travers de l'assistant de création des légendes, nous créons trois légendes. La première légende nommée *Legend1* doit être affichée dans la première région. Pour ce faire, il faut valoriser sa propriété *DockedToChartArea* avec le nom de la région. Procéder de manière analogue pour les deux autres légendes, nommées *Legend2* et *Legend3*.



3.5.1.4 Création des séries de données

Notre graphique contient quatre séries de données, réparties dans ses trois régions. La première région contient deux séries, la seconde une, tout comme la troisième. Nous affichons alors l'assistant de création des séries de données, accessible via la propriété *Series* du contrôle *Chart*. Nous ajoutons quatre séries de données :

- La première doit s'afficher dans la région *NombreCommandesParClient*. Nous valorisons ses propriétés suivantes :

Propriété	Valeur
<i>Name</i>	<i>Nombre de commandes réalisées</i>
<i>ChartArea</i>	<i>NombreCommandesParClient</i>
<i>Column</i>	<i>Column</i>
<i>Legend</i>	<i>Legend1</i>
<i>XValueMember</i>	<i>RaisonSociale</i>
<i>YValueMembers</i>	<i>NombreCommandesRealisees</i>

- La seconde doit aussi s'afficher dans la région *NombreCommandesParClient*. Nous valorisons ses propriétés suivantes :

Propriété	Valeur
<i>Name</i>	<i>Nombre de commandes potentielles</i>
<i>ChartArea</i>	<i>NombreCommandesParClient</i>
<i>Column</i>	<i>Column</i>

<i>Legend</i>	<i>Legend1</i>
<i>XValueMember</i>	<i>RaisonSociale</i>
<i>YValueMembers</i>	<i>Objectif</i>

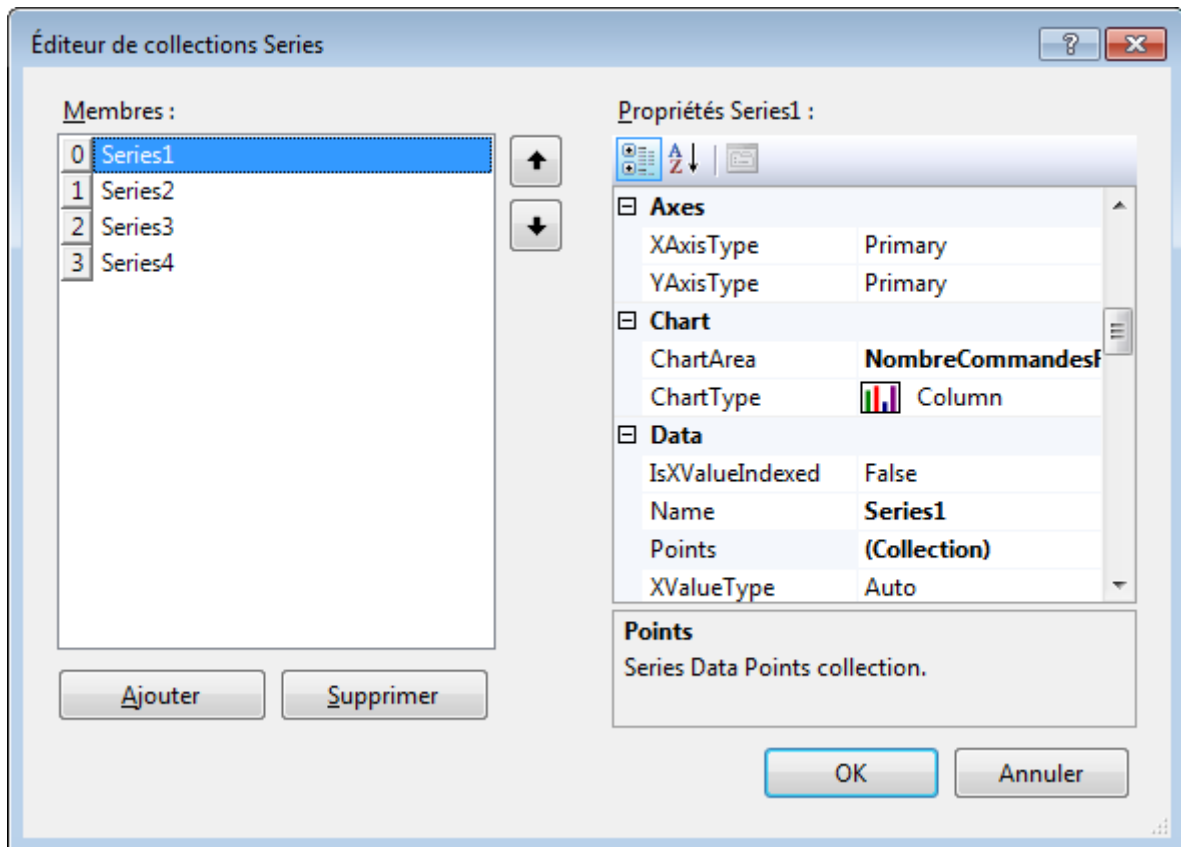
Attention, tous les types de graphiques ne peuvent être affichés au sein de la même région. Dans le cas d'une incompatibilité, une exception de type *InvalidOperationException* sera levée.

- La troisième doit s'afficher dans la région *ChiffreAffaireParClient*. Nous valorisons ses propriétés suivantes :

Propriété	Valeur
<i>ChartArea</i>	<i>ChiffreAffaireParClient</i>
<i>Column</i>	<i>Doughnut</i>
<i>Legend</i>	<i>Legend2</i>
<i>XValueMember</i>	<i>RaisonSociale</i>
<i>YValueMembers</i>	<i>ChiffreAffaire</i>

- La quatrième doit s'afficher dans la région *PotentielParClient*. Nous valorisons ses propriétés suivantes :

Propriété	Valeur
<i>ChartArea</i>	<i>PotentielParClient</i>
<i>Column</i>	<i>Pyramid</i>
<i>Legend</i>	<i>Legend3</i>
<i>XValueMember</i>	<i>RaisonSociale</i>
<i>YValueMembers</i>	<i>NombreCommandesPotentielles</i>



4 Conclusion

Ce cours vous a présenté *Microsoft Chart Control*, ainsi que ses possibilités et comment l'utiliser. Ce contrôle s'utilise aussi bien dans les applications Web et Windows Forms, où il permettra de créer dans les formulaires une meilleure présentation des données.

Pour en savoir plus sur ce contrôle, nous vous invitons à consulter la documentation de référence : [Microsoft Chart Controls for .NET Framework Documentation](#)