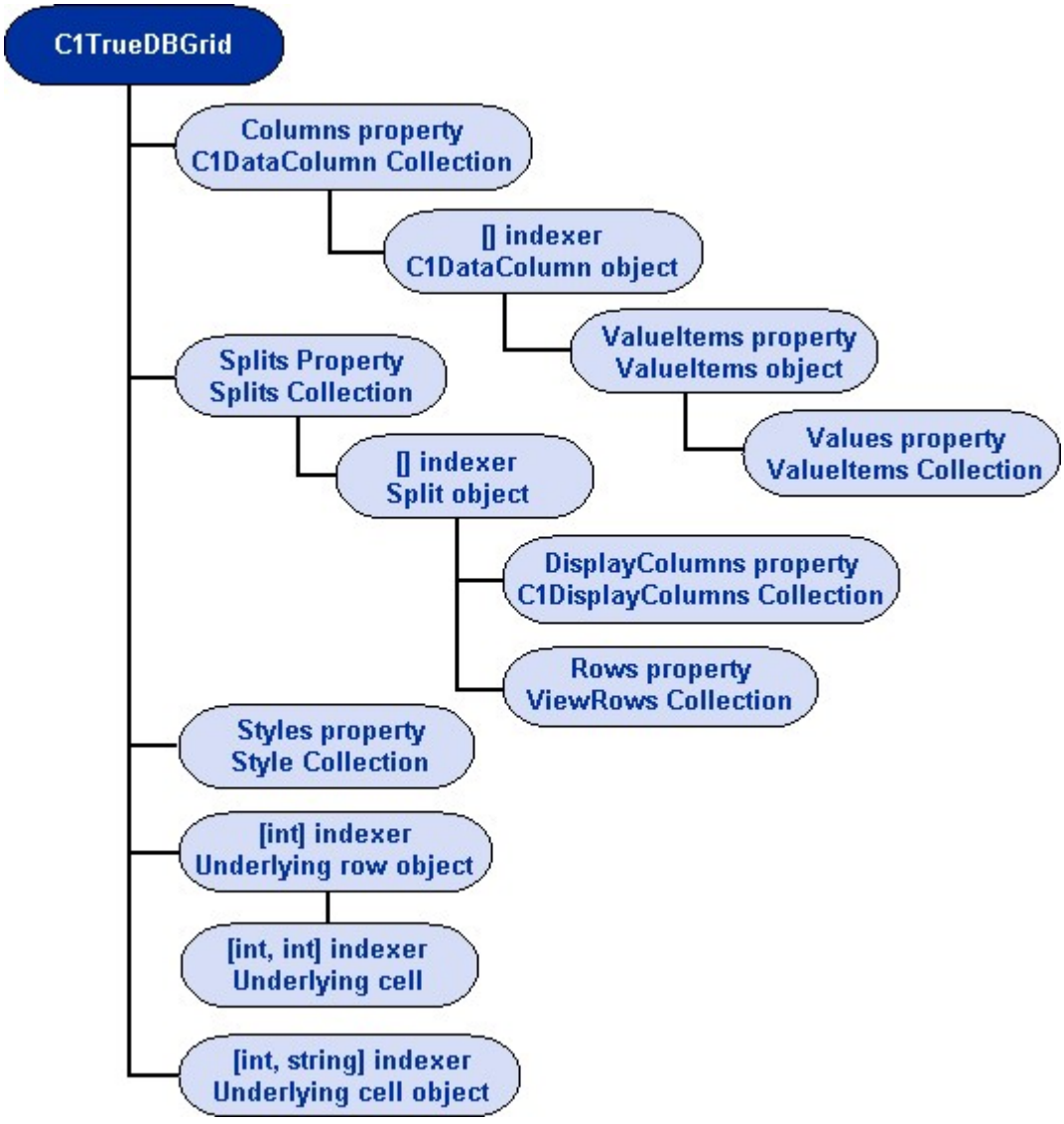
对象模型

WinForms版版True DBGrid使用最新的.NET技术来研发，WinForms版版True DBGrid控件以及它的可编程控件均为根据Microsoft规范设计的.NET对象，如果您已经熟悉Microsoft .NET对象和集合模型，您将很容易上手WinForms版版True DBGrid for。
如果您是Visual Studio的新用户，请阅读使用对象与集合使用对象与集合 (Section 5.18)，它介绍了如何在代码中使用WinForms版版True DBGrid 对象，单个对象可以设计执行不同的任务， 用来操纵他们的技术都是相同的，一旦您掌握了这些通用的编程结构，Visual Studio控件的使用 将会变得更加轻松和直观。
不管您是否有相关的经验，请参阅以下章节，它提供的所有WinForms版版True DBGrid对象和集合的略缩概要。

WinForms版True DBGrid对象与集合

WinForms版版True DBGrid 拥有丰富的对象模型，并包含以下元素：



WinForms版版True DBGrid 提供丰富的属性集、方法以及事件，可以方便您开发复杂的数据库应用程序，通过对True DBGrid对象模型的组织您可以非常轻松控制一个大的功能集。对象和集合包含可视实体，如在设计器或者代码中可以被自定义的列，对象和集合还包含抽象的实体，如仅在代码中可用的数组和书签。
通过加入到项目中，这两个控件在.NET工具箱中是可用的：

Control	Description
C1TrueDBGrid	WinForms版版True DBGrid 网格控件。

C1TrueDBDropDown	WinForms版版True DBGrid 下拉栏控件。

WinForms版版True DBGrid 的命名空间也包含以下对象的定义：

Object	Description
C1DataColumn	代表网格中一个数据列。
C1DisplayColumn	代表相关拆分的一个数据列。
GridLines	代表网格中分割项的网格线。
HBar	代表水平滚动条及它的属性。
PrintPreviewWinSettings	封装了打印预览窗阔以及它的属性。
PrintInfo	封装了页面设置和打印作业设置。
Split	代表相邻列的一个分组作为一个滚动单元。
Style	封装了字体，颜色，图像及格式信息。
ValueItems	封装了Values集合与ValueItem属性。
ValueItem	允许列输入值和可选择翻译。
VBar	代表垂直滚动条和它的属性。

集合是一个对象用于分组相似的数据项，如网格列或者样式。一般情况下，WinForms版True DBGrid中的相似项分组可以实现为一个集合，因为集合是一个对象，它可以像其他对象一样在代码中实现控制，WinForms中中True DBGrid 包含以下集合：

Collection	Description
C1DataColumnCollection	包含0或者网格中多个C1DataColumn对象。
C1DisplayColumnCollection	包含0或者网格中多个C1DisplayColumn对象。
GroupedColumnCollection	包含0或者分组区域中多个C1DataColumn对象。
SelectedRowCollection	包含0或者选择的列索引。
SelectedColumnCollection	包含0或者代表所选择列的多个C1DataColumn对象。
SplitCollection	包含网格中一个或者多个Split 对象。
GridStyleCollection	包含一个网格中内置的或者用户定义的样式对象。
ValueItemCollection	包含0或者一个列中多个ValueItem对象.

以下章节提供了WinForm版版True DBGrid对象与集合的简要概述。

C1TrueDBGrid类

C1TrueDBGrid控件是WinForms版版True DBGrid 的主要对象，使用它的C1DataColumnCollection 和C1DisplayColumnCollection 对象可以创建，访问和更改用于映射网格的物理列与底层的数据库字段列对象，使用它的SplitCollection对象，网格可以被划分为多个水平或者垂直的窗格，为相同的数据源提供不同的视图。

C1TrueDBDropDown类

C1TrueDBDropDown 控件是C1TrueDBGrid 控件的一个子集，用于网格列的多列的下拉列表，C1TrueDBDropDown 控件不能作为一个独立控件。在设计器中，将一个C1TrueDBDropDown控件放置于窗体中，同放置一个C1TrueDBGrid控件一样，然而，下拉控件在运行时是不可见的，直到添加C1TrueDBGrid控件的一个C1DataColumn 对象。
为了使用下拉控件，需要在设计器或者代码中设置网格列的DropDown 属性为一个C1TrueDBDropDown控件的名称，在运行时，当用户点击单元格内的按钮来选取一个列时，C1TrueDBDropDown控件将出现在网格当前单元格的下方，如果用户从下拉控件中选取了一个项，网格当前的单元格将会被更新，C1TrueDBDropDown 控件也支持渐进式搜索。

C1DataColumnCollection类

C1TrueDBGrid控件和C1TrueDBDropDown控件均包含一个C1DataColumnCollection 对象和C1DataColumn 对象，该集合包含在C1TrueDBGrid对象之下，并可以通过C1TrueDBGrid Designer更改，同时它也可以通过WinForms 版版True DBGrid的Columns 属性来访问。

C1DataColumn对象

在C1TrueDBGrid 或 C1TrueDBDropDown 控件中的每一列由两个列对象，一个全局及一个特定拆分来表示，所有与数据访问和格式化相关属性都被包含在C1DataColumn 对象中，C1DataColumn对象中的属性均为全局的，C1DataColumn 属性的改变可以对所有列值，甚至跨拆分进行改变，C1DataColumn对象可以按照如下访问：
To write code in Visual Basic

Visual Basic
Me.C1TrueDBGrid1.Columns(0).Caption = "Region"

To write code in C#

C#
this.c1TrueDBGrid1.Columns[0].Caption = "Region";

C1DisplayColumnCollection类

C1TrueDBGrid控件和C1TrueDBDropDown控件均包含一个C1DisplayColumnCollection 对象和C1DisplayColumn对象，该集合包含在Split对象之下，并可以通过Split的DisplayColumns使用，此外，集合可以通
过C1DisplayColumnCollection Editor在.NET中被更改。更多详细信息，请参阅使用使用C1DisplayColumnCollection编辑编辑器器 (Section 6.3)。

C1DisplayColumn对象

在网格中的每一个拆分都包含至少一个C1DisplayColumn对象，关于列显示的所有属性都包含在这个对象中，不
像C1DataColumn属性，C1DisplayColumn 对象的所有属性都是特定拆分的，改变C1DisplayColumn 的属性将改变特定拆分中特定列的值，对象可以通过以下方式访问：

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.Splits(0,0).DisplayColumns(0).Style.ForeColor =
System.Drawing.Color.Blue
```

To write code in C#

```
C#
this.c1TrueDBGrid1.Splits[0,0].DisplayColumns[0].Style.ForeColor =
System.Drawing.Color.Blue;
```

GroupedColumnCollection类

当DataView 属性被设置为DataViewEnum.GroupBy，分组区域将被创建在网格上，该集合对象代表分组区域中的列 (C1DataColumn 对象)。当列被拖进或拖出分组区域，在集合中相应的列将被添加或删除。

SplitCollection类

C1TrueDBGrid控件和含一个SplitCollection集合，其也包含和Split对象，网格默认拆分为一个，但也可以被拆分为多个，该集合可以通过使用C1TrueDBGrid的Split属性来访问，此外该集合也可以通过在.NET中的Split Collection Editor来更改，请参阅使用集合拆分编辑器使用集合拆分编辑器 (Section 6.2)获取更多信息。

Split 对象

WinForms版版True DBGrid支持类似于Excel的拆分，允许将网格划分为垂直和水平窗格以支持用户对数据资源不同视图，Split 对象代表一个拆分，它包含相邻列的一个分组并作为一个滚动单元。
当一个C1TrueDBGrid 控件被创建，它默认包含了一个Split对象，Split 对象的所有属性均独立应用于C1TrueDBGrid控件，因此不需要关心拆分，除非需要创建一个固定的非滚动的列，对象可以按照以下方式访问：
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.Splits(0).Caption = "Split00"
```

To write code in C#

```
C#
this.c1TrueDBGrid1.Splits[0].Caption = "Split00";
```

GridStyleCollection类

C1TrueDBGrid和C1TrueDBDropDown 控件均在GridStyleCollection 对象中内置保存并用户自定义Style对象，可以在代码中通过名称访问集合成员，并应用他们到网格、列或拆分以控制控件对象的外观，该集合可以在WinForms
版版True DBGrid控件中使用Styles 属性来访问，此外，该集合和它的成员可以通过C1TrueDBGrid Style Editor 在.NET 中更改。

Style对象

Style 对象封装了C1TrueDBGrid, C1TrueDBDropDown, Split, 或C1DisplayColumn对象的字体，颜色，图片以及格式信息，Style 对象是一个非常灵活且强大的工具，它提供了类似于Excel和Word的格式能力以控制网格显示的外观。
当创建了一个C1TrueDBGrid 或 C1TrueDBDropDown控件，它包含了十个内置的样式，您也可以在设计器或代码中更改内置样式或添加自定义样式，此外，通过使用Style对象将格式信息传递给每个单元格或者单元行，两个控件都可以实现几个可选的事件，对象可以通过以下方法访问额：
To write code in Visual Basic

Visual Basic
Me.C1TrueDBGrid1.Styles("Normal").Backcolor = System.Drawing.Color.Gray

To write code in C#

C#
this.c1TrueDBGrid1.Styles["Normal"].Backcolor = System.Drawing.Color.Gray;

ValueItems类

ValueItems 对象包含一个集合和一组属性，可以在网格中创建数据库值的交替显示，它可以为给定的C1DataColumn对象指定一个允许输入的值，或者它可以用于将原始数据转换为备用文本或者图像显示(例如，
用Balance Due 和 Paid in Full 替代数值数据0和1)。ValueItems 对象包含显示属性和ValueItem 对象的集合，该对象通过以下方式访问：
To write code in Visual Basic

Visual Basic
Me.C1TrueDBGrid.Columns(0).ValueItems.MaxComboItems = 5

To write code in C#

C#
this.c1TrueDBGrid.Columns[0].ValueItems.MaxComboItems = 5;

ValueItemCollection类

在C1TrueDBGrid 或 C1TrueDBDropDown 控件中的每个C1DataColumn 对象将它的值/值对保存到对象中，并命名为ValueItem 对象，ValueItemCollection 对象这些值对的集合，该集合可以通过ValueItems对象的Values 属性访问，例如，想要在集合中选择第一个ValueItem，其代码如下：
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid.Columns(0).ValueItems.Values(0).DisplayValue = "Canada"
```

To write code in C#

```
C#
this.c1TrueDBGrid.Columns[0].ValueItems.Values[0].DisplayValue = "Canada";
```

ValueItem类

ValueItem 对象由两个属性组成：DisplayValue和Value，Value 属性指定了数据库中基础值，DisplayValue 属性指定在网格中值的显示，这些对象可以包含在ValueItemCollection 对象中，并在.NET中的ValueItemCollection Editor编辑，该编辑器在ValueItems 对象中的C1TrueDBGrid Designer 下编辑，更多详细信息请参阅使用使用
ValueItemCollection 编辑器编辑器 (Section 6.4)。

PrintInfo类

PrintInfo 对象用于指定页面布局和打印工作字符如输出设备的名字，间距设置，页眉与页脚以及打印的份数。
C1TrueDBGrid控件的PrintInfo 属性返回更改打印工作的对象。
PrintInfo 对象是持久的，这就意味着一个打印布局可以在设计时定义，运行时可以在代码中重新被调用。

PrintPreviewWinSettings类

PrintPreviewWinSettings 对象提供了网格打印预览窗口的相关属性，通过该对象，页眉页脚和其他可视元素均可以在预览窗口中被设置，通过C1TrueDBGrid 控件的PreviewInfo 属性来访问。

HBar类

HBar 对象用于指定水平滚动条的属性，通过使用HScrollBar 属性，开发人员可以指定滚动条的高度，以及是否能够自动显示。

VBar类

VBar 对象用于指定垂直滚动条的属性，通过使用VScrollBar 属性，开发人员可以指定滚动条的高度，以及是否能够自动显示。

GridLines类

GridLines 对象用于指定ColumnDivider及RowDivider 属性的特性，通过使用GridLines对象可以在运行或者设计时对行或者列线的颜色与样式更改。

GridBorders类

GridBorders 对象用于指定Style的Borders属性特性，该属性设置单元格的列边框，通过使用对象，开发人员可以指定每一个单元格边框的宽度和单元格边框的颜色。

SelectedRowCollection类

当用户在运行时选择并高亮C1TrueDBGrid控件的一行或多行，所选择行的行索引被保存在SelectedRowCollection对象，在代码中，集合的Item属性和IndexOf方法用于决定哪一行被选择，也可以使用它的Add和RemoveAt方法编程实现选择或者取消选择记录。

SelectedColumnCollection类

当用户在运行时选择并高亮C1TrueDBGrid控件的一列或多列，所选择列的列索引被保存在SelectedRowCollection对象，在代码中，集合的Item属性和IndexOf方法用于决定哪一列被选择，也可以使用它的Add和RemoveAt方法编程实现选择或者取消选择记录。

使用对象与集合

本节将描述如何在代码中更有效率的使用对象与集合，尽管这个概念已经在WinForms版版True DBGrid对象与集合对象与集合 (Section 5.1)中阐述了，其相同的工作原理可以应用在所有的Visual Studio对象和集合。
当WinForms版版True DBGrid控件被置于一个窗体中，C1TrueDBGrid 对象就会被创建，在Visual Studio中创建的C1TrueDBGrid 对象默认命名为C1TrueDBGrid1, C1TrueDBGrid2 以此类推，控件名称可以在设计时通过属性窗口改变。

使用集合

C1TrueDBGrid 对象拥有8个独立的集合用以支配不同的对象，每一个集合关联C1TrueDBGrid 内的属性并返回集合对象，这可以避免当需要在代码中使用网格时，开发人员要输入整个集合名，以下表格显示了集合与属性的对应关系：

Collection	Associated Property
C1DataColumnCollection	Columns属性

C1DisplayColumnCollection	DisplayColumns属性
GridStyleCollection	Styles属性
SelectedColumnCollection	SelectedCols属性
SelectedRowCollection	SelectedRows属性
SplitCollection	Splits属性
ValueItemCollection	Values属性

默认情况下，SplitCollection对象包含一个Split 对象，GridStyleCollection 对象包含十种默认的Style 对象：Normal, Heading, Footing, Selected, Caption, HighlightRow, EvenRow, OddRow, RecordSelector以及以及FilterBar。
使用基于0的索引引用在集合中的对象，按照下述方法读取或设置Split 对象的属性：
To write code in Visual Basic

Visual Basic

```
' 读取一个Split对象属性。
variable = Me.C1TrueDBGrid1.Splits(0).Property


' 设置一个Split对象属性。
Me.C1TrueDBGrid1.Splits(0).Property = variable
```

To write code in C#

C#

```
// 读取一个Split对象属性。
variable = this.c1TrueDBGrid1.Splits[0].Property;


// 设置一个Split对象属性。
this.c1TrueDBGrid1.Splits[0].Property = variable;
```

使用集合的Item 方法创建集合中对象的一个引用，以下代码创建网格默认Split对象一个引用：
To write code in Visual Basic

Visual Basic

```
' 声明Split0为一个Split对象。
Dim Split0 As C1.Win.C1TrueDBGrid.Split


' 设置Split0为集合中第一个Split的引用。

Split0 = Me.C1TrueDBGrid1.Splits(0)
```

To write code in C#

C#

```
// Declare Split0 as Split object.
C1.Win.C1TrueDBGrid.Split Split0;
// Set Split0 to reference the first Split in the collection.

Split0 = this.c1TrueDBGrid1.Splits[0];
```

注意之前的示例中使用了命名空间限定符，推荐使用命名空间限定符已解决与其他控件的潜在命名冲突，例如，如果其他控件也用于同一个项目，并定义一个名为Split的对象，此时WinForms版版True DBGrid 命名空间限定符就是必要的，命名控件限定符也用于其他控件。
因此Item方法对于隐含集合，它会被省略：
To write code in Visual Basic

Visual Basic

```
' 声明Split0为一个Split对象。
Dim Split0 As C1.Win.C1TrueDBGrid.Split


' 设置Split0为集合中第一个Split的引用。

Split0 = Me.C1TrueDBGrid1.Splits(0)
```

To write code in C#

C#

```
// 声明Split0为Split对象。
C1.Win.C1TrueDBGrid.Split Split0;

// 设置Split0为集合中第一个Split的引用。

Split0 = this.c1TrueDBGrid1.Splits[0];
```

使用Split0来读取或设置Split 对象属性或执行它的方法：
To write code in Visual Basic

Visual Basic

```
' 读取一个Split对象属性。
variable = Split0.Property

' 设置一个Split对象属性。
Split0.Property = variable

' 执行一个Split对象方法。
Split0.Method (arg1, arg2, ...)
```

To write code in C#

C#

```
// 读取一个Split对象属性。
variable = Split0.Property;

// 设置一个Split对象属性。
Split0.Property = variable;

// 执行一个Split对象方法。
Split0.Method (arg1, arg2, ...);
```

在很多时候，您需要读取和设置多个对象的属性，例如：
To write code in Visual Basic

Visual Basic

```
' 读取一个Split对象的属性。
variable1 = Me.C1TrueDBGrid1.Splits(0,0).Property1

variable2 = Me.C1TrueDBGrid1.Splits(0,0).Property2


' 设置一个Split对象的属性。
Me.C1TrueDBGrid1.Splits(0,0).Property1 = variable1

Me.C1TrueDBGrid1.Splits(0,0).Property2 = variable2
```

To write code in C#

```
C#
// 读取一个Split对象的属性。
variable1 = this.c1TrueDBGrid1.Splits[0,0].Property1;

variable2 = this.c1TrueDBGrid1.Splits[0,0].Property2;

// 设置一个Split对象的属性。
this.c1TrueDBGrid1.Splits[0,0].Property1 = variable1;

this.c1TrueDBGrid1.Splits[0,0].Property2 = variable2;
```

该段代码效率是非常低下的，因为C1TrueDBGrid1.Splits(0,0)这个对象被访问的次数非常多，更有效的方法是对对象创建一个引用并可以反复的使用它：
To write code in Visual Basic

```
Visual Basic
' 声明Split0为一个Split对象。
Dim Split0 As C1TrueDBGrid.Split

' 设置Split0为集合中第一个Split的引用。
Split0 = Me.C1TrueDBGrid1.Splits.Item(0,0)

' 读取一个Split对象的属性。
variable1 = Split0.Property1 variable2 = Split0.Property2


' 设置一个Split对象的属性。
Split0.Property1 = variable1
Split0.Property2 = variable2
```

To write code in C#

```
C#
// 声明Split0为一个Split对象。
C1TrueDBGrid.Split Split0;

// 设置Split0为集合中第一个Split的引用。
Split0 = this.c1TrueDBGrid1.Splits[0,0];
// 读取一个Split对象的属性。
variable1 = Split0.Property1;

variable2 = Split0.Property2; // Set a Split object's properties.

Split0.Property1 = variable1;
Split0.Property2 = variable2;
```

该代码更有效并且更容易读取，如果在Visual Studio应用中频繁访问集合对象，通过遵守以上规则将会显著提高代码的性能。
同样的，此项技术应用于Visual Studio中True DBGrid的其他对象与集合，这里对于网格十分重要的对象还包括C1DataColumn 和 C1DataColu

mnCollection 对象 (也包括C1DisplayColumn 对象):
To write code in Visual Basic

Visual Basic

```
' 声明Cols为一个Columns集合对象，此时将它设置为C1TrueDBGrid1的C1DataColumnCollection对象的引用。

Dim Cols As C1.Win.C1TrueDBGrid.C1DataColumnCollection
Cols = Me.C1TrueDBGrid1.Columns

' 声明Col0为一个C1DataColumn对象，此时将它设置为集合中第一个Column对象的引用。
Dim Col0 As New C1.Win.C1TrueDBGrid.C1DataColumn
```

```
Col0 = Cols(0)

' 读取并设置C1DataColumn对象的Property1。
variable1 = Col0.Property1 Col0.Property1 = variable1

' 执行C1DataColumn对象的Method1 (声明为一个Sub)。
Col0.Method1 (arg1, arg2, ...)

' 执行C1DataColumn对象的Method2 (声明为一个Function)。
variable2 = Col0.Method2(arg1)
```

To write code in C#

C#

```
// 声明Cols为一个Columns集合对象，此时将它设置为C1TrueDBGrid1的C1DataColumnCollection对象。

C1.Win.C1TrueDBGrid.C1DataColumnCollection Cols;
Cols = this.c1TrueDBGrid1.Columns;

// 声明Col0为一个C1DataColumn对象，此时将它设置为集合中第一个Column对象的引用。
C1.Win.C1TrueDBGrid.C1DataColumn Col0 = new C1TrueDBGrid.DataColumn();
Col0 = Cols[0];

// 读取并设置C1DataColumn对象的Property1。
variable1 = Col0.Property1; Col0.Property1 = variable1;

// 执行C1DataColumn对象的Method1 (声明为一个Sub)
Col0.Method1 (arg1, arg2, ...);

// 执行C1DataColumn对象的Method2 (声明为一个Function)。
variable2 = Col0.Method2(arg1);
```

Visual Basic
也提供高效的声明方式，可以不需要精确的分配变量就能设置对象的多个属性，例如下述代码设置网格中第一列多个属性(注意集合是从0开始的):
To write code in Visual Basic

Visual Basic

```
With Me.C1TrueDBGrid1.Columns(0)

.Property1 = variable1
.Property2 = variable2
End With
```

To write code in C#

C#

```
this.c1TrueDBGrid1.Columns[0].Property1 = variable1;

this.c1TrueDBGrid1.Columns[0].Property2 = variable2;
```

添加成员

为了创建并添加一个对象到一个集合，需要使用集合的 Add 方法，该方法将对象作为它的唯一参数，例如：通过添加一个ValueItem对象到ValueItemCollection对象中创建包含多个值项的列：
To write code in Visual Basic

```
Visual Basic

' 创建一个ValueItem对象。
Dim v As C1TrueDBGrid.ValueItem = new C1TrueDbGrid.ValueItem()

Me.C1TrueDBGrid1.Columns(0).ValueItems.Values.Add(v)
```

To write code in C#

```
C#

// 创建一个ValueItem对象。
C1TrueDBGrid.ValueItem v = new C1TrueDBGrid.ValueItem();

this.c1TrueDBGrid1.Columns[0].ValueItems.Values.Add(v);
```

该代码添加了一个ValueItem 对象到C1TrueDBGrid1的ValueItemCollection中，或者创建了一个从索引1开始的ValueItem 对象插入方法：
To write code in Visual Basic

```
Visual Basic

' 创建了一个从索引1开始的Split对象。
Dim S As C1TrueDBGrid.ValueItem
Me.C1TrueDBGrid1.Columns(0).ValueItems.Values.Insert(1, S)
```

To write code in C#

```
C#

//创建了一个从索引开始的Split对象。
C1TrueDBGrid.ValueItem S; this.c1TrueDBGrid1.Columns[0].ValueItems.Values.Insert(1, S);
```

唯一不能使用Add 或 RemoveAt 方法添加或删除的对象是Split对象，Split对象的 InsertHorizontalSplit / RemoveHorizontalSplit和 InsertVerticalSplit / RemoveVerticalSplit 方法必须正确使用添加或移除Splits，这些方法可以在设计时右键点击网格的上下文菜单使用。

移除成员

不管集合如何实现Add或Insert方法，删除项的语法都是相同的，若要从集合中移除现有项，请使用RemoveAt 方法:
To write code in Visual Basic

```
Visual Basic
' 从索引1开始移除Split对象。
Me.C1TrueDBGrid1.Columns(0).ValueItems.Values.RemoveAt(1)
```

To write code in C#

```
C#
// 从索引1开始移除Split对象。
this.c1TrueDBGrid1.Columns[0].ValueItems.Values.RemoveAt(1);
```

当该语句被执行后，集合中的所有索引值大于1的splits都会下调1并填充移除split的空间，注意到RemoveAt 方法的参数是待移除成员的位置。 5.18.1.3 使用Count属性
使用集合的Count属性决定集合中对象的编号：
To write code in Visual Basic

```
Visual Basic
' 设置一个等于C1TrueDBGrid1中Split编号的变量。

variable = Me.C1TrueDBGrid1.Splits.Count
```

To write code in C#

```
C#
// 设置一个等于C1TrueDBGrid1中Split编号的变量。

variable = this.c1TrueDBGrid1.Splits.Count;
```

按照下例中使用Count 属性遍历集合中的所有对象，可以打印网格中每个C1DataColumn 对象的Caption字串：
To write code in Visual Basic

```
Visual Basic
For n = 0 To Me.C1TrueDBGrid1.Columns.Count - 1
Debug.WriteLine(Me.C1TrueDBGrid1.Columns.Caption)

Next n
```

To write code in C#

```
C#
for (n = 0; n < this.c1TrueDBGrid1.Columns.Count; n++) {

Console.WriteLine(this.c1TrueDBGrid1.Columns[n].Caption);


}
```

Count 属性也可以用于追加和移除列：

To write code in Visual Basic

```
Visual Basic
```

```
' 决定列数。
Dim NumCols As Integer
NumCols = Me.C1TrueDBGrid1.Columns.Count

' 在Columns集合后面追加一个列。
Dim C As C1TrueDBGrid.C1DataColumn = New C1TrueDBGrid.C1DataColumn()
Me.C1TrueDBGrid1.Columns.Insert(NumCols, C)
' 让新的列可见，默认情况下在运行时创建的列是不可见的。
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(C).Visible = True
' 网格中的所有列将通过循环全部移除。
While Me.C1TrueDBGrid1.Columns.Count
Me.C1TrueDBGrid1.Columns.RemoveAt(0)
End While
```

To write code in C#

```
C#
```

```
// 决定列数。
int NumCols; NumCols = this.c1TrueDBGrid1.Columns.Count;

// 在Columns集合后面追加一个列。
C1TrueDBGrid.C1DataColumn C = new C1TrueDBGrid.C1DataColumn(); this.c1TrueDBGrid1.Columns.Insert(NumCols, C);


// 让新的列可见，默认情况下在运行时创建的列是不可见的。
this.c1TrueDBGrid1.Splits[0].DisplayColumns[C].Visible = true;
// 网格中的所有列将通过循环全部移除。
while ( this.c1TrueDBGrid1.Columns.Count > 0 )

{ this.c1TrueDBGrid1.Columns.RemoveAt(0);
}
```

For Each...Next 这样高效的语句可以不必使用Count属性对集合中对象进行遍历：
To write code in Visual Basic

```
Visual Basic
```

```
Dim C As C1TrueDBGrid.C1DataColumn
For Each C In Me.C1TrueDBGrid1.Columns

Debug.WriteLine(C.Caption)
Next S
```

To write code in C#

```
C#
```

```
C1TrueDBGrid.C1DataColumn c;

foreach (c In this.c1TrueDBGrid1.Columns) {

Console.WriteLine(c);
}
```

事实上，使用For Each...Next语句是最方便的方法对集合中的对象进行遍历。