TeeChart for .Net 教程

Table of Contents

[1. TeeChart Pro .Net教程 2](#_Toc361220394)

[1.1 目录 2](#_Toc361220395)

[1.1.1 教程1 - 入门 2](#_Toc361220396)

[1.1.2 教程2 - 图表显示属性 12](#_Toc361220397)

[1.1.3 教程3 － 图表页 27](#_Toc361220398)

[1.1.4 教程4 － 坐标轴控件 31](#_Toc361220399)

[1.1.5 教程5 － 图例设计 48](#_Toc361220400)

[1.1.6 教程6 － 使用数据序列 53](#_Toc361220401)

[1.1.7 教程7 － 使用函数 75](#_Toc361220402)

[1.1.8 教程8 － ADO.NET数据库访问 89](#_Toc361220403)

[1.1.9 教程9 － ASP.NET应用程序 99](#_Toc361220404)

[1.1.10 教程10 - ASP.NET 实例的初步学习 114](#_Toc361220405)

[1.1.11 教程11- 缩放和滚动 117](#_Toc361220406)

[1.1.12 教程12 - 导出和导入图表 120](#_Toc361220407)

[1.1.13 教程13 - 在图表面板上自定义绘图 132](#_Toc361220408)

[1.1.14 教程14 - 打印图表 144](#_Toc361220409)

[1.1.15 教程15 - 3D图表 147](#_Toc361220410)

[1.1.16 教程16 - TeeChart附加面板组件 150](#_Toc361220411)

[1.1.17 教程17 - 设计时 运行时及授权规定 153](#_Toc361220412)

[1.1.18 教程18 - TeeChart 工具集 162](#_Toc361220413)

[1.1.19 教程19 - 自定义序列化 225](#_Toc361220414)

# 1. TeeChart Pro .Net教程



[*慧都科技－Steema中文版中国地区总代*](http://www.evget.com)

[*联系我们*](mailto:suport@evget.com)

*© 2002-2009 Steema Software SL. All rights reserved.*

## 1.1 目录

**教程**

[教程1 － 入门](#Bookmark2)(See 1.1.1)   
[教程2 － 图表显示属性](#Bookmark3)(See 1.1.2)   
[教程3 － 图表页](#Bookmark4)(See 1.1.3)   
[教程4 － 坐标轴控件](#Bookmark5)(See 1.1.4)   
[教程5 － 图例设计](#Bookmark6)(See 1.1.5)   
[教程6 － 数据序列的运用](#Bookmark7)(See 1.1.6)   
[教程7 － 函数的运用](#Bookmark8)(See 1.1.7)   
[教程8 － ADO.NET 数据库访问](#Bookmark9)(See 1.1.8)   
[教程9 －](#Bookmark9) (See 1.1.8)[ASP.NET 应用程序](#Bookmark10)(See 1.1.9)   
[教程10 －](#Bookmark9) (See 1.1.8)[ASP.NET 实例演练](#Bookmark11)(See 1.1.10)   
[教程11 －](#Bookmark9) (See 1.1.8)[缩放和滚动](#Bookmark12)(See 1.1.11)   
[教程12](#Bookmark9)(See 1.1.8) [－ 导入和导出图表](#Bookmark13)(See 1.1.12)   
[教程13](#Bookmark9)(See 1.1.8) [－ 在图表面板上自定义绘图](#Bookmark14)(See 1.1.13)   
[教程14](#Bookmark9)(See 1.1.8) [－ 打印图表](#Bookmark15)(See 1.1.14)   
[教程15](#Bookmark9) (See 1.1.8)[－ 3D图表](#Bookmark16) (See 1.1.15)   
[教程16](#Bookmark9)(See 1.1.8) [－](#Bookmark17) (See 1.1.16)[TeeChart](#Bookmark14)(See 1.1.13)[TeeChart附加面板组件](#Bookmark17)(See 1.1.16)   
[教程17](#Bookmark9)(See 1.1.8) [－ 设计时，运行时和授权规定](#Bookmark18)(See 1.1.17)   
[教程18](#Bookmark9)(See 1.1.8) [－ TeeChart工具集](#Bookmark19)(See 1.1.18)

[*联系我们*](mailto:info@steema.com)

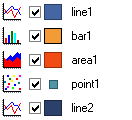
*© 2002-2009 Steema Software SL. All rights reserved.*

### 1.1.1 教程1 - 入门

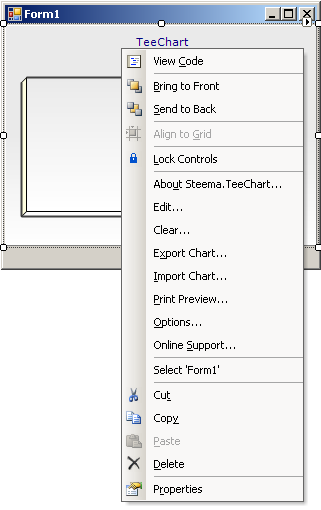
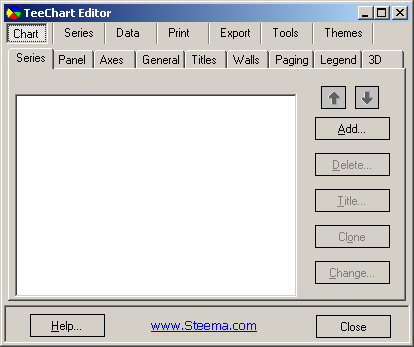
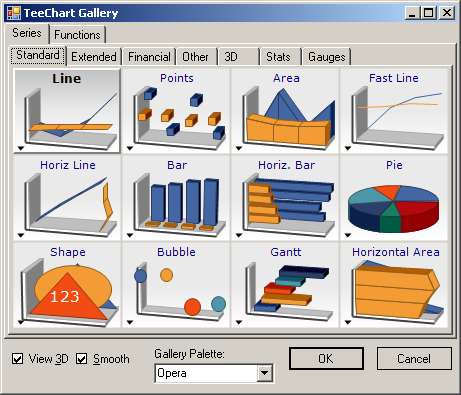
**教程01 － 入门**

**入门 － 创建图表与填充数据序列  
  
目录**   
[**简介**](#Bookmark2_Introduction)  
[**创建一个图表**](#Bookmark2_BuildChart)  
[**将TeeChart添加到一个窗体上**](#Bookmark2_IncTChart)   
     [图表编辑器](#Bookmark2_Editor)   
     [图表数据序列集](#Bookmark2_Gallery)   
  
[**填充新的数据序列**](#Bookmark2_PopSeries)   
     [ADO.NET](#Bookmark2_ADO)   
     [使用代码进行数据输入](#Bookmark2_CodedData)   
  
[**选择一个满足您需要的序列类型**](#Bookmark2_SelSeries)

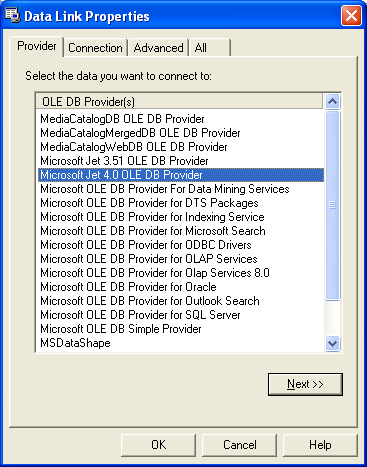
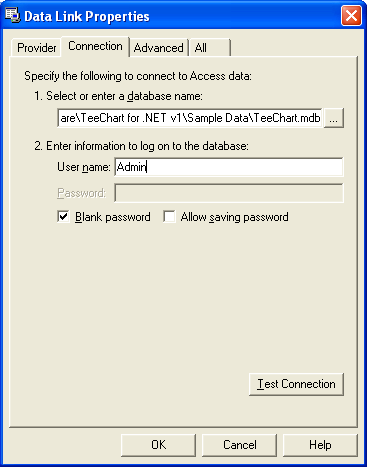
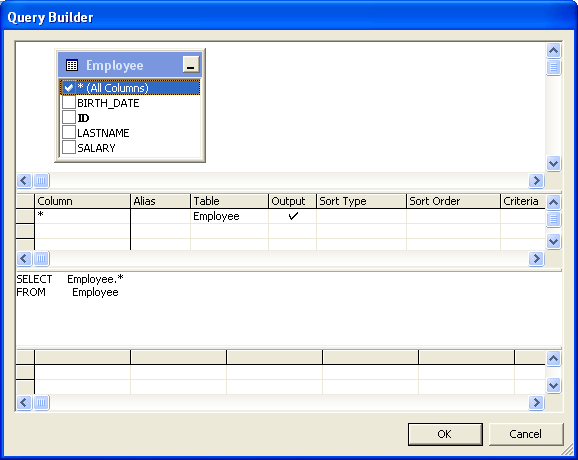
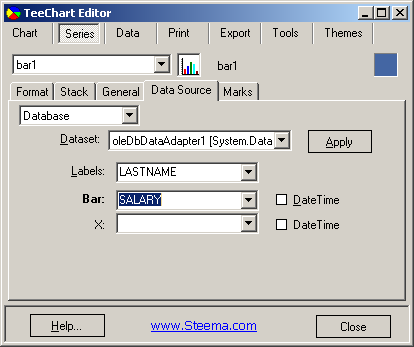
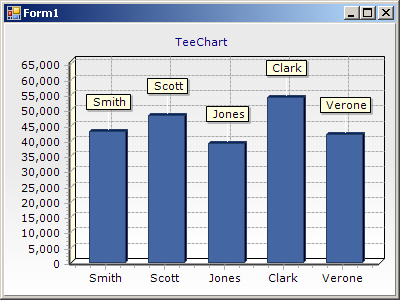
**说明**

使用TeeChart编辑器及少量或完全无需用代码就可以创建和实现基本的图表。TeeChart编辑器是一个二合一的编辑器，图表能与完全不同的数据序列相关联。您可以在不插入一个数据序列的情况下来定义图表的外观，名称，图例的特性和3D样式。您可以无需重新定义整个图表的外观样式，就可在运行时任意的添加和删除不同类型的数据序列。   
  
  
  
图表的坐标轴是图表和数据序列的接口，这些组件可以协同工作。一个数据序列，它的值将影响图表坐标轴的标签特征。坐标轴的外观样式，颜色，网格密度特性和标签字体都可以在插入数据序列之前为图表定义好。TeeChart编辑器将对大多数的值作出一个最好的估算，然后来预定义您所需要的图表和数据序列。  
  
**通过图表编辑器添加多种序列类型:**  TeeChart的“特殊的数据序列图表”设计模式允许你混合使用多种类型的系列，而不限于选择一组预定义的图表格式。  
  
  
  
**互联网！**TeeChart Pro .Net完全兼容ASP.NET，在设计时，它能够放在一个ASP.NET WebForm上，并且在设计时能以一个完全所见即所得的方式来编辑图表。在运行时，您可以选择下面列出的图像格式进行输出：JPEG,GIF,PNG,TIFF,BMP和EMF。通过 [互联网应用教程](#Bookmark10)(See 1.1.9) 获得更多信息。   
  
此教程中强调的是轻松地从零开始建立一个图表的步骤，当您学习完后面的教程就可以修改和进一步改善图表外观和功能。我们开始吧！

**创建图表**

**[将TeeChart添加到窗体中](" \l "Bookmark2_IncTChart)**  
通过使用TeeChart默认的安装程序，您已经将TeeChart默认图标成功添加到Visual Studio .Net设计环境的工具箱中。开始体验！  
  
TeeChart图表的图标是用一个圆型（饼图）图来表示的。  
  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\button1.png  
  
选择并点击工具栏中的TeeChart图标，并将其拖放到窗体上。您将看到类似一个面板的TeeChart，并伴有一些基本的图表特征，如轴和标题等。但TeeChart绝不仅仅就是一个面板；右击这个新的TeeChart您会看到一个菜单，选择Edit...命令可查看TeeChart编辑器对话框。  
  
  
  
关于TeeChart其他的Pallete组件，请点击 [教程16](#Bookmark17)(See 1.1.16)。  
  
**TeeChart编辑器**   
TeeChart编辑器包括最常用的方法和图表的属性。  
  
  
  
添加一个新的数据序列，点击Add...按钮从弹出的数据序列集中选择您想要的数据序列类型。如果选择好数据序列类型之后您又想改变它，您可以这样做：返回到第一个编辑器对话框并选择Change...按钮（注意：可能会发生少数异常，因为数据序列的数据内容不兼容）。  
  
**[图表数据序列分类](" \l "Bookmark2_Gallery)**  
  
  
这个图表序列包含五个标签页：:   
  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png 标准类  
标准数据序列类型是使用最频繁的一类。  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png 统计类  
统计数据序列包括更专业的统计规划或绘图应用序列类型。  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png 金融类   
金融数据序列包括更专业的财务金融应用序列类型。  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png 扩展类   
可扩展数据序列类型有一些例子包括定制数据序列类型。  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png 3D类   
3D数据序列类型需要三个（x，Y，Z轴）变量，而不是两个（X，Y）。  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\blbullet.png其它  
不同的数据序列类型。  
  
选择一个数据序列类型。在这个例子中我们选择柱状图数据序列。这个集合关闭后，新的数据序列将在第一个图表编辑器屏幕上显示出来。

**[填充新的数据序列](" \l "Bookmark2_PopSeries)**

选择图表编辑器对话框的Series选项来查看数据序列的属性配置。 在Series属性标签下的第四个选项是数据源标签。  
  
**ADO.NET**如果您使用的数据源是ADO.NET，那么您将要在图表编辑器中配置您的数据源路径。  
  
在这个例子中，我们将连接BarSeries到TeeChart.mdb Access数据库。 首先，从VS.NET工具箱数据标签，拖拽一个OleDbDataAdapter到一个有TeeChart for.Net项目的WinForm上。在数据适配器配置向导中点击“Next”，并在下一个页面点击“新建联接”按钮。然后在新窗口中点击“Provider”标签，选择“Microsoft Jet OLE DB Provider”，如下图所示：  
  
（ **\*请注意 － 不同框架的向导外观可能稍有不同**）  
  
  
  
现在，请在同一窗口中点击“Connection”标签，然后从TeeChart .NET根目录下的数据样本文件选择TeeChartTeeChart.mdb文件：  
  
  
  
点击“Test Connection”按钮来测试联接，然后再点击同一窗口中的OK按钮。 现在点击数据适配器向导窗口中的“Next”按钮，从出现的窗口中选择“Use SQL Statements”，再次点击“Next”按钮。现在请点击“Query Builder”按钮并在"Employee" 表中添加查询。 关闭“Add Table”窗口， 然后选择“\*（所有列）”复选框：  
  
  
  
现在，确定是这个窗口，点击“下一步”，然后选择“完成”。 现在打开TeeChar编辑器, 将BarSeries和导航添加到该数据序列标签的数据源标签。 从组合框和数据设置组合框中分别选择“Database”和oleDbDataAdapter1 [System.Data.OleDb.OleDbDataAdapter] 。设置标签为“LASTNAME”并设置Y为“SALARY”，如下图所示：  
    
  
  
现在点击“Apply”按钮并运行该窗体，您现在应该有类似下面的内容：  
  
  
  
更多信息，请看 [教程08 - ADO.NET 数据库访问](#Bookmark9)(See 1.1.8)   
  
  
**[通过代码输入数据](" \l "Bookmark2_CodedData)**对于数据输入，您需要编写一些代码。本节将告诉您所需的步骤，用于创建一个使用编码输入的图表。  
  
在设计时，您先前用图表编辑器添加的数据序列会显示随机的填充数据。但是在运行时，该数据序列将是空的数据，除非您手动填充它。从工具箱中选择一个按钮，并放置在您的窗体中。在Visual Studio .Net中，双击该按钮进入代码编辑器 （在不同的编程环境，该方法可能会有所不同）。  
  
在代码编辑器中键入以下内容：

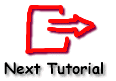
[C#.Net]   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            bar1.Add(3,"Pears",Color.Red);   
            bar1.Add(4,"Apples",Color.Blue);   
            bar1.Add(2,"Oranges",Color.Green);   
        }

[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Bar1.Add(3, "Pears", Color.Red)   
        Bar1.Add(4, "Apples", Color.Blue)   
        Bar1.Add(2, "Oranges", Color.Green)   
End Sub

运行该项目并按下按钮，在您的图表中将显示三个新的柱形图。 就是这样，一个简单的图表就完成了！  
  
在您的图表中，第一个柱形图数据序列被称作Bar1，第二个称作Bar2等... 因此，如果您的图表中有多种数据序列，您都可以使用类似的名字来修改相关的序列名称。  
  
该数据序列的“Add”方法共有25个重载，其中第7个接受3个变量：值，文本和颜色。Add方法从而取得与标签轴间距相等的值 （这里指的是X轴）。如果您的数据只包含两个变量的话，那么您就要使用“Add”的另一个重载方法。 添加另一个按钮到您的项目，并将下面的代码键入到里面。 运行该项目，键入的代码如下：

[C#.Net]   
private void button2\_Click(object sender, System.EventArgs e)   
        {   
            Random rnd = new Random();   
            if(bar1.Count > 0)   
            {   
                //Increment X Axis value and add a new random point   
                bar1.Add((bar1.XValues.Last + rnd.Next(10)), ((bar1.YValues.Last/(bar1.YValues.Last - 1)) + rnd.Next(10)),"Lemons",Color.Yellow);   
            }   
            else   
            {   
                //Add a new random point   
                bar1.Add(1,rnd.Next(10),"Lemons",Color.Yellow);   
            }   
        }   
   
[VB.Net]   
Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click   
        Dim rnd As Random   
        If (Bar1.Count > 0) Then   
            'Increment X Axis value and add a new random point   
            Bar1.Add((Bar1.XValues.Last + rnd.Next(10)), ((Bar1.YValues.Last / (Bar1.YValues.Last - 1)) + rnd.Next(10)), "Lemons", Color.Yellow)   
        Else   
            'Add a new random point   
            Bar1.Add(1, rnd.Next(10), "Lemons", Color.Yellow)   
        End If   
End Sub

**[选择一个满足您需要的数据序列类型](" \l "Bookmark2_SelSeries)**

最后的实例代码可生成新的X、Y值。 在X轴点与点之间的距离并不固定，在某些情况下这将导致柱形图重叠在一起，这种情况对少部分程序是可取的，但不适用于所有的应用程序。使用图表编辑器将数据序列类型修改为（在设计时，进入图表编辑器，然后点击修改按钮进行修改，同时，您还需要修改代码行中关于Bar1的所有引用）线型序列从而显示另外一种使用两个变量图形化显示数据的方法 。  
  
如果您的数据超过2个变量，还有其他数据序列类型适合用于显示这些数据。您可以从图表库中选择您所想要的数据序列类型，同时您可以阅读TeeChar帮助文档来获得更多的有用信息。   
  
  
[](#Bookmark3)(See 1.1.2)

[*联系我们*](mailto:info@steema.com)

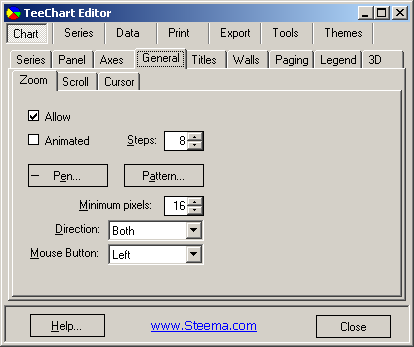
*© 2002-2009 Steema Software SL. All rights reserved.*

### 1.1.2 教程2 - 图表显示属性

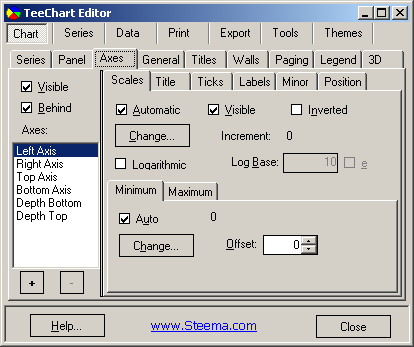
**教程2 － 图表显示属性**

图表显示特征是指那些影响图表整体外观的因素。它们包括那些定义图表背景颜色，标题，位置，页边，粗细和斜角，背景图片，框架和轴可见，画笔颜色和宽度，3D，立面等属性和方法。您可以在设计时或运行时可通过图表编辑器使用这些属性，并且您还可以在运行时以编程的方式来修改这些属性。图表的整体外观特征分为以下类别：   
  
[**总体**](#Bookmark3_General)   
[**坐标轴**](#Bookmark3_Axis)   
[**标题**](#Bookmark3_Title)   
[**图例**](#Bookmark3_Legend)   
[**面板**](#Bookmark3_Panel)   
[**分页**](#Bookmark3_Paging)   
[**立面**](#Bookmark3_Walls)   
[**3D**](#Bookmark3_ThreeDOptions)   
  
如果您打开图表编辑器和浏览图表编辑页就会发现这些组别。 其它的属性将影响您图表的外观， 其中包括数据序列的颜色和特有的标记，个别坐标轴和网格显示的属性和标签。这些将在后面的教程中进行专门的介绍。  
  
**图表显示属性入门**添加一个新的图表到窗体中，您会看到它将显示为一个带有灰色背景的3D面板，2个立面（底部和左边），框架（定义图表plottable区域的框架）和一个默认的图表标题页眉“TeeChar” 。   
右击图表并从图表编辑器的菜单中选择“Edit...”。第一页是空的数据序列页面，在本教程中我们暂不关心。选择第三个“General”标签来查看图表总体属性页面。

**图表总体属性**

  
  
总体选项包括以下三部分：   
  
缩放   
滚动   
指针   
  
缩放和滚动将在后面的教程中详细介绍。

**坐标轴属性**

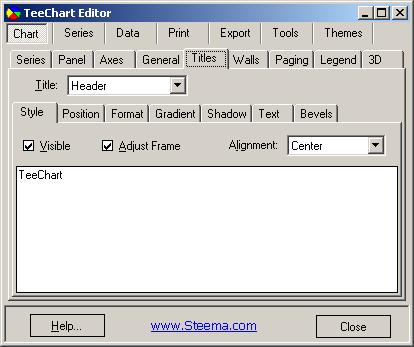
坐标轴控件是一个特殊的领域并且相当的复杂，因此我们将在后面的教程中进行深入讲解。 在这里我们只介绍基本轴和框架显示内容。  
第二个图表页面有定义轴和框架特征的属性。  
  
  
  
图表中包含6个标准轴。 左边，顶部，右边，底部， 底部纵深和顶部纵深。图表框架显示为图表plottable区域和无数据（参见BackWall）对应功能的环绕形式。默认的初始化纵深是不可见的。一个数据序列被添加到图表并与这些轴（默认为左边，底部）进行关联时，其他所有的轴是可见的。在对话框上可以通过“+”和“-”两个按钮进行添加或移除自定义轴。如果想让自定义轴是可见的（关于任何其他轴），那么一个数据序列必须与轴相关联。请参阅轴控件教程了解更多自定义轴的信息。  
使轴和框架能够显示的关键特性如下：  
  
**可见性**   
此复选框可启用或禁用所有轴的显示。如果后面的立面框架是可见的，即使轴被隐藏了图表环绕也依然可见。即使没有一个区域是为轴标签所留的，也仍会显示一个更大的图表。每个轴（比例标签）的可见性会覆盖每个轴的显示特性。如果您在列表中选择的是纵深轴，您需要注意纵深标尺默认是不可见的。

[C#.Net]   
tChart1.Axes.Visible = true;   
   
[VB.Net]   
TChart1.Axes.Visible = True

坐标轴：左，右，顶部，底部，和纵深可见   
在列表框中选择您想要显示或者隐藏的轴，并可以切换刻度标签上的复选框来控制该特定轴的显示。

[C#.Net]   
tChart1.Axes.Left.Visible = true;   
   
[VB.Net]   
TChart1.Axes.Left.Visible = True

**标题属性**

TeeChart编辑器的标题页用于控制图表标题，页眉和页脚的特性。   
  
  
  
**文本**   
使用下拉复选框选择标题（头），子标题，页脚或子页脚。在文本框中输入所需的文本。 您可以输入多行标题。  
  
运行时：

Header类的文本属性是一个标准的.Net字符串类，您可以像下面这样进行修改，例如：

[C#.Net]   
tChart1.Header.Text = "My Header Text";   
   
[VB.Net]   
TChart1.Header.Text = "My Header Text"

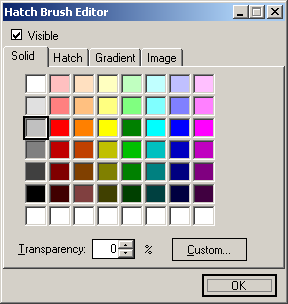
**对齐样式**   
对齐是指图表区域的页眉（或页脚）而不是指整个图表面板。图表区域是图表加上轴标签及图例的plottable区域。  
  
运行时：

[C#.Net]   
tChart1.Header.Alignment = StringAlignment.Far;   
   
[VB.Net]   
TChart1.Header.Alignment = StringAlignment.Far

**位置**   
使用位置属性可以覆盖标题或者页脚的默认位置，并且用户可以自定义一个位置属性（像素关系到图表顶部，左边）。  
  
运行时：

[C#.Net]   
tChart1.Header.CustomPosition = true;   
tChart1.Header.Top = tChart1.Header.Top + 10;   
   
[VB.Net]   
TChart1.Header.CustomPosition = True   
TChart1.Header.Top = TChart1.Header.Top + 10

**格式化：**   
包含标题框的设置，例如标题框的颜色，标题框周围框架的画笔属性， 标题框的格式刷属性[Pattern]等。  
  
例如： 渐变  
渐变指标题框或者页脚框的背景模式[填充，纹理，渐变，图片]。

  
  
默认是透明的，这意味着透明复选框未被选中前，标题框都是不可见的。  
  
运行时：

[C#.Net]   
tChart1.Header.Transparent = false;   
tChart1.Header.Brush.Style = System.Drawing.Drawing2D.HatchStyle.Cross;   
   
[VB.Net]   
TChart1.Header.Transparent = False   
TChart1.Header.Brush.Style = System.Drawing.Drawing2D.HatchStyle.Cross

**文本**   
包含标题文本格式的外观特性。   
  
例如： 字体   
字体按钮将允许使用字体对话框窗口，在对话框中允许选择Windows的字体和定义（斜体，粗体等）样式和颜色。   
  
运行时：

[C#.Net]   
tChart1.Header.Font.Name = "Verdana";   
tChart1.Header.Font.Bold = true;   
   
[VB.Net]   
TChart1.Header.Font.Name = "Verdana"   
TChart1.Header.Font.Bold = True

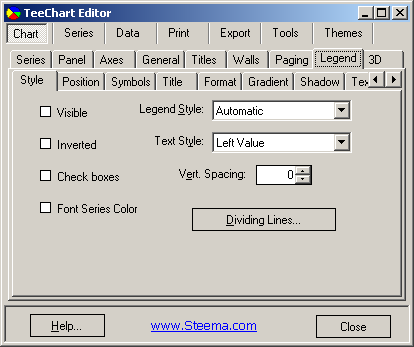
**阴影**   
页眉或页脚框阴影显示的属性。你可以定义颜色和图表格式刷 [填充，纹理，渐变或图像]。  
  
运行时：

[C#.Net]   
tChart1.Header.Shadow.Visible = true;   
tChart1.Header.Shadow.Color = Color.Chartreuse;   
   
[VB.Net]   
TChart1.Header.Shadow.Visible = True   
TChart1.Header.Shadow.Color = Color.Chartreuse

**Bevels**定义页眉框或者页脚框的**Bevels**属性。

**图例属性**

图例的内容是一个特殊的主题，我们将在后面的教程中详细介绍。TeeChart编辑器的图例页面让您可以定义图例的外观。非常重要的  
初始化步骤是控制图例对齐方式和可见属性，颜色，字体，框架，阴影等。

  
  
**样式**   
打开/关闭图例显示和设置图例显示内容的特性（包含复选框等）。  
  
运行时：

示例中使用Checkbox来控制图例的显示：

[C#.Net]   
private void checkBox1\_CheckedChanged(object sender, System.EventArgs e)   
        {   
            if(checkBox1.Checked)   
            {tChart1.Legend.Visible = true;}   
            else   
            {tChart1.Legend.Visible = false;}   
        }   
   
[VB.Net]   
Private Sub CheckBox1\_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged   
        If CheckBox1.Checked Then   
            TChart1.Legend.Visible = True   
        Else   
            TChart1.Legend.Visible = False   
        End If   
End Sub

%分界线   
用来在两个图例框之间绘制中线。图表画笔属性是非常有用的。  
  
运行时：

[C#.Net]   
tChart1.Legend.DividingLines.Visible = true;   
tChart1.Legend.DividingLines.Color = Color.Blue;   
   
[VB.Net]   
TChart1.Legend.DividingLines.Visible = True   
TChart1.Legend.DividingLines.Color = Color.Blue

**位置**   
设置默认显示位置或激活自定义的图例配置。 TChart会改变图例的形状，以适应所在的位置。如果图例被设置在图表的一侧（左或右），默认情况下，将从上到下进行排列。如果图例低于或高于图表，那么图表内容将并排放置。默认的可能会使用调整图表选项和（或）自定义配置来覆盖。您可以查看更多关于自定义图例的教程。

例如：调整图表大小   
调整图表大小是指当图例绘制时图表是否离开图例的范围。 这使得图例将始终在图表区域内。  
  
运行时：

[C#.Net]   
tChart1.Legend.ResizeChart = true;   
   
[VB.Net]   
TChart1.Legend.ResizeChart = True

%顶部位置  
定义图例框从图表面板顶部中进行移植。  
  
运行时：

[C#.Net]   
tChart1.Legend.CustomPosition = true;   
tChart1.Legend.Top = 5;   
   
[VB.Net]   
TChart1.Legend.CustomPosition = True   
TChart1.Legend.Top = 5

**标记**   
定义图例标记的大小和格式。  
  
%颜色 宽度   
定义图例中颜色框的宽度。

运行时：

[C#.Net]   
tChart1.Legend.Symbol.WidthUnits = LegendSymbolSize.Pixels;   
tChart1.Legend.Symbol.Width = 30;   
   
[VB.Net]   
TChart1.Legend.Symbol.WidthUnits = Steema.TeeChart.LegendSymbolSize.Pixels   
TChart1.Legend.Symbol.Width = 30

**格式化**   
包含图例框的设置，例如颜色，框架（ChartPen属性），模式[ChartBrush属性] ，球形框架和透明复选框。   
  
**文本**   
图例文本的内容和阴影的文本特性。  
  
运行时：

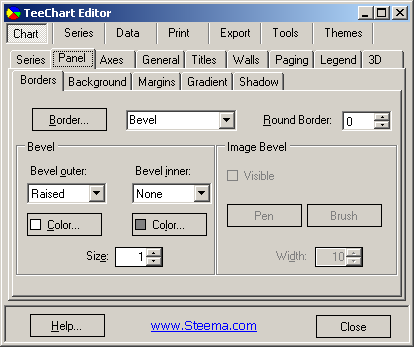
[C#.Net]   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            Legend legend1;   
            legend1 = tChart1.Legend;   
            legend1.Color = Color.Blue;   
            legend1.Font.Name = "Times New Roman";   
            legend1.Font.Color = Color.Yellow;   
        }   
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        With TChart1.Legend   
            .Color = Color.Blue   
            .Font.Name = "Times New Roman"   
            .Font.Color = Color.Yellow   
        End With   
End Sub

**渐变**   
激活/禁用图例中渐变背景并选择其方向和颜色。  
  
**阴影**   
图例框阴影的显示属性。 您可以定义颜色，模式（ChartBrush）大小和透明。  
  
运行时：

[C#.Net]   
tChart1.Legend.Shadow.Color = Color.Cyan;   
tChart1.Legend.Shadow.Width = 4;   
   
[VB.Net]   
TChart1.Legend.Shadow.Color = Color.Cyan   
TChart1.Legend.Shadow.Width = 4

**斜角**颜色和内，外斜角大小的属性。

**面板属性**

TeeChart面板页面允许您设置参数来大大改善图表的外观。面板渐变效果在支持真彩的显示器效果是更佳的。  
  
现在您已经熟悉了TeeChart的属性，所以在这里我们不会涵盖所有的属性。面板属性和方法，您可以通过访问面板接口：  
  
  
  
重点领域是：  
  
**边框**   
您可以独立地定义边框或将它与Bevel属性一起使用。可对几乎所有的3D组合进行混合Bevel、边框以及熟练控制宽度的操作。  
  
例如：Bevel Inner/Bevel Outer/Border Width  
这些属性将在图表面板的边框生成多种3D效果。在设计时修改它们，看看它们如何影响图表边框的Bevels。   
  
**背景**   
例如：面板颜色   
面板将用所选的颜色绘制整个图表的背景。如果您选择查看渐变背景或图像，他们将隐藏面板颜色。如果您设置了背景的颜色，它将用图表框架的颜色取代面板的颜色。  
  
例如：背景图片  
您可以选择一些位图文件来当作背景图片。使用Hatch Brush编辑器（从模式按钮），有五种不同的Wrap Modes可用来选择背景图片。  
运行时：  
在运行时，使用ChartBrush.LoadImage向图表添加一个背景图片。

[C#.Net]   
tChart1.Panel.Brush.WrapMode = System.Drawing.Drawing2D.WrapMode.TileFlipXY;   
tChart1.Panel.Brush.LoadImage(@"C:\...\MyImage.jpg");   
   
[VB.Net]   
TChart1.Panel.Brush.WrapMode = System.Drawing.Drawing2D.WrapMode.TileFlipXY   
TChart1.Panel.Brush.LoadImage("C:\...\MyImage.jpg")

**页边距**   
页边距定义为图表边缘和图表框架的距离，并且用图表面板整体尺寸的百分比来表示。

在设计时，在图表编辑器页边距框中移动spin框将显示所做改变的效果   
  
运行时属性是：

TChart1.Panel.MarginTop    
TChart1.Panel.MarginLeft   
TChart1.Panel.MarginRight   
TChart1.Panel.MarginBottom

您可以使用HScrollbar来修改这些属性：

[C#.Net]   
private void Form1\_Load(object sender, System.EventArgs e)   
        {   
            bar1.FillSampleValues(10);   
            hScrollBar1.Value = Convert.ToInt32(tChart1.Panel.MarginLeft);   
            hScrollBar1.Maximum = 80;   
        }   
   
private void hScrollBar1\_Scroll(object sender, System.Windows.Forms.ScrollEventArgs e)   
        {   
            tChart1.Panel.MarginLeft = Convert.ToDouble(e.NewValue);   
        }   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Bar1.FillSampleValues(10)   
        HScrollBar1.Value = TChart1.Panel.MarginLeft   
        HScrollBar1.Maximum = 80   
End Sub   
   
Private Sub HScrollBar1\_Scroll(ByVal sender As Object, ByVal e As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll   
        TChart1.Panel.MarginLeft = e.NewValue   
End Sub

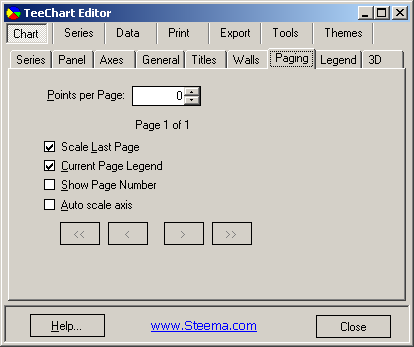
**渐变**  
要定义渐变您必须选择一个StartColor和EndColor（加上或者可选MidColor），使其成为可见的渐变效果。 渐变将覆盖整个图表面板。 渐变方向定义了StartColor，MidColor，EndColor之间颜色变化的方向。

运行时：

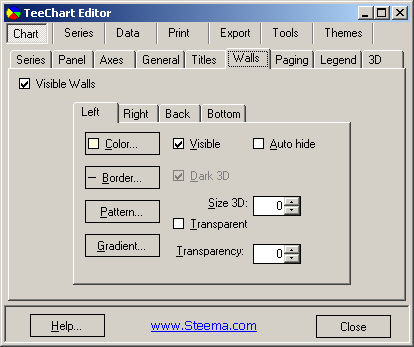
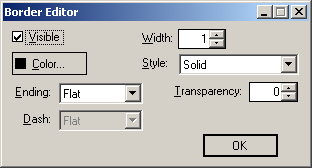
[C#.Net]   
tChart1.Panel.Gradient.Visible = true;   
tChart1.Panel.Gradient.StartColor = Color.Red;   
tChart1.Panel.Gradient.MiddleColor = Color.White;   
tChart1.Panel.Gradient.EndColor = Color.Blue;   
tChart1.Panel.Gradient.Direction = System.Drawing.Drawing2D.LinearGradientMode.ForwardDiagonal;   
   
[VB.Net]   
With TChart1.Panel   
         .Gradient.Visible = True   
         .Gradient.StartColor = Color.Red   
         .Gradient.MiddleColor = Color.White   
         .Gradient.EndColor = Color.Blue   
         .Gradient.Direction = System.Drawing.Drawing2D.LinearGradientMode.ForwardDiagonal   
End With

**阴影**定义图表面板阴影的大小，颜色，模式[ChartBrush]和透明等属性。

**页面属性**

TeeChart页允许图表被划分为每页设置数字，而且图表可以根据这些数字进行翻阅。更多信息请见页面教程。  
  


**立面属性**

在TeeChart编辑器的图表区页面对图表的立面属性进行设置。 这里有四种立面：左，右，底部和后面，它们都可能是2D或3D的。  
  
立面属性可以通过TChart1.Walls获得。  
  
  
  
**颜色**   
通过颜色按钮您将看到颜色编辑器，您可以为选定的立面选择一个颜色。  
  
**边框**   
通过边框按钮您可用ChartPen定义立面类。ChartPen与其它TeeChart项目所用到的画笔是一样的。  
  
  
  
可见属性启用/禁用边框的显示特性。 就BackWall而言，它里面的轴是可见的，这些立面的边框将会隐藏在这些轴的后面，即便您隐藏部分或所有的轴，您都只能看见启用或禁用边框的效果（例如隐藏轴线上方和右侧来观察改变立面边颜色的结果，即TChart1.Walls.Back.Pen.Color = Color.Blue）。

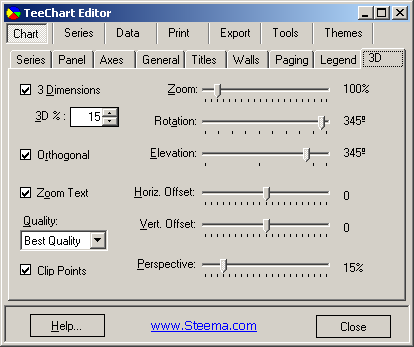
查看ChartPen关于绘图笔的另外一些特性。您可以在设计时修改这些属性。  
  
**模式**   
通过模式按钮您将看到ChartBrush关于Walls类的定义。ChartBrush与其它TeeChart项目所用到的格式刷是一样的。  
  
**渐变**   
通过渐变按钮您将看到渐变编辑器，您可以为已经选定的立面选定渐变色。

**3D**

3D在TeeChart Pro.中是非常灵活的。你现在可以选择图表显示为：

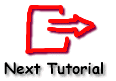
 2D Flat Chart, viewed from front

 3D Orthogonal 3D represented by vectored Lines to indicate depth

 3D Native Windows 3D using 3D Canvas with 180 degree Chart rotation   
  
  
  
**3 Dimensions**   
在设计时，选择此属性（切换）可将图表从3D变成2D，而在运行时您将看到这些设置的效果。  
  
**3D %**   
控制3D效果的纵深。在设计时，修改此属性的值您将看到改变3D百分比所产生的效果。  
  
使用CheckBox和Horizontal ScrollBar这些属性的一个应用实例：

[C#.Net]   
private void checkBox1\_CheckedChanged(object sender, System.EventArgs e)   
        {   
            if(checkBox1.Checked)   
            {   
                tChart1.Aspect.View3D = true;   
                hScrollBar1.Enabled = true;   
            }   
            else   
            {   
                tChart1.Aspect.View3D = false;   
                hScrollBar1.Enabled = false;   
            }   
        }   
   
private void hScrollBar1\_Scroll(object sender, System.Windows.Forms.ScrollEventArgs e)   
        {   
            tChart1.Aspect.Chart3DPercent = e.NewValue;   
        }   
   
[VB.Net]   
Private Sub CheckBox1\_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged   
        If CheckBox1.Checked Then   
            TChart1.Aspect.View3D = True   
            HScrollBar1.Enabled = True   
        Else   
            TChart1.Aspect.View3D = False   
            HScrollBar1.Enabled = False   
        End If   
End Sub

Private Sub HScrollBar1\_Scroll(ByVal sender As Object, ByVal e As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll   
        TChart1.Aspect.Chart3DPercent = e.NewValue   
End Sub

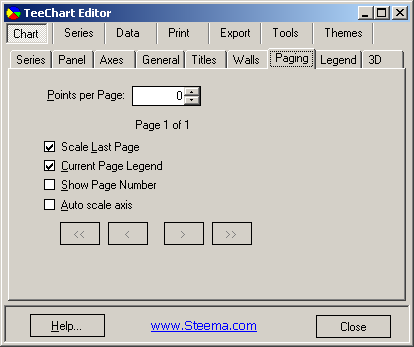
无论Orthogonal选项是启用还是禁用，Windows Native 3D 模式都允许您进行旋转，提升和偏移图表操作。通过缩放选项，您可以显示或移除整个图表。缩放图表的数据序列在运行时仍然是可用。  
Perspective属性允许您设置透视的距离，就像进入一个房间进行寻找。 Perpsective提供了一个用于增强图表的视觉演示功能。对于3D操作的参考资料请查看 [教程15－3D图表.](#Bookmark16) (See 1.1.15)  
  
[](#Bookmark2)(See 1.1.1)     [](#Bookmark4)(See 1.1.3)

### 1.1.3 教程3 － 图表页

**教程3 － 图表页**

如果您的图表数据源包含了比图表屏幕显示更多的数据，你不妨划分为可以翻阅页的图表来显示。要做到这一点可以通过TeeChart编辑器或编程。  
[**使用TeeChart编辑器进行分页**](#Bookmark4_EditorPaging)   
[**使用代码进行分页**](#Bookmark4_CodePaging)

**使用图表编辑器进行分页**

页面大小可以通过TeeChart编辑器来设定。这仍然是需要添加分页按钮或ChartPageNavigator组件到您的项目中，或者您也可以让TeeChart编辑器在运行时允许用户使用它来修改页面。   
  
在设计时，请在TeeChart编辑器中选择分页选项。  
  
  
  
每页显示的数目**:** 在框中输入（或滚动）你希望在图表页上看到数据序列的数量。如果您通过编码来设定这个数字，导航就不会突出显示按钮，直到您运行该项目，输入数字并在运行时显示图表编辑器。如果您连接到ADO.NET数据源您应该在设计时就立即看到分页的效果。  
  
运行时：  
使用ShowEditor可以在运行时访问图表编辑器的分页属性。

[C#.Net]   
tChart1.ShowEditor();   
   
[VB.Net]   
TChart1.ShowEditor()

**使用代码进行分页**

使用分页属性和方法可以通过Page类。  
添加分页到您的图表所需的步骤：

 定义每页显示的数字   
使用MaxPointsPerPage定义每个页面上显示的数量：

[C#.Net]   
tChart1.Page.MaxPointsPerPage = 10;   
[VB.Net]   
TChart1.Page.MaxPointsPerPage = 10

 添加导航按钮到图表的窗体   
例子：   
第一页

[C#.Net]   
tChart1.Page.Current = 0;   
[VB.Net]   
TChart1.Page.Current = 0

下一页

[C#.Net]   
tChart1.Page.Next();   
[VB.Net]   
TChart1.Page.Next()

前一页

[C#.Net]   
tChart1.Page.Previous();   
[VB.Net]   
TChart1.Page.Previous()

最后一页

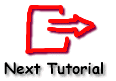
[C#.Net]   
tChart1.Page.Current = tChart1.Page.Count;   
[VB.Net]   
With TChart1   
  .Page.Current = .Page.Count   
End With

 决定最后一页是什么   
最后一页是不大可能拥有完全正确的数目来配合其他图表页面的数字。您可以选择最后一页，它将是最适合剩余内容的一页，调整相应的轴刻度，或者您可以像前面的页面一样处理这些页面，即使没有更多的页面分给它，数目也是完全正确的且不会给最后一页留下空白，。

[C#.Net]   
tChart1.Page.ScaleLastPage = false;   (default = true)   
[VB.Net]   
TChart1.Page.ScaleLastPage = False  (default = True)

 查看用户页码   
PageNumber图表工具可用于实现这一目标：

[C#.Net]   
private void Form1\_Load(object sender, System.EventArgs e)   
        {   
            Bar bar1 = new Bar();   
            PageNumber pageNumber1 = new PageNumber();   
            tChart1.Series.Add(bar1);   
            tChart1.Tools.Add(pageNumber1);   
            bar1.FillSampleValues(100);   
            tChart1.Page.MaxPointsPerPage = 10;   
        }   
   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            tChart1.Page.Previous();   
        }   
   
private void button2\_Click(object sender, System.EventArgs e)   
        {   
            tChart1.Page.Next();   
        }   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar()   
        Dim PageNumber1 As New Steema.TeeChart.Tools.PageNumber()   
        TChart1.Series.Add(Bar1)   
        TChart1.Tools.Add(PageNumber1)   
        Bar1.FillSampleValues(100)   
        TChart1.Page.MaxPointsPerPage = 10   
End Sub   
   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        TChart1.Page.Previous()   
End Sub   
   
Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click   
        TChart1.Page.Next()   
End Sub

这就是本教程的全部内容！下一个教程将介绍轴和图例的高级操作。  
  
  
[](#Bookmark3)(See 1.1.2)     [](#Bookmark5)(See 1.1.4)

### 1.1.4 教程4 － 坐标轴控件

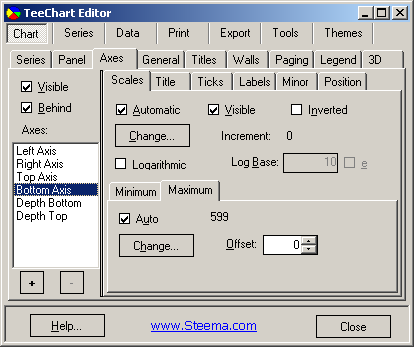
**教程4 － 坐标轴控件**

TeeChart Pro会自动为您定义所有坐标轴的标签，它们具有高度的灵活性可适应您任何特定的要求。TeeChart Pro提供了真正的多种坐标轴。在设计或运行时这些内容是可见的，另外它还提供了很多功能并在定义坐标轴时具有高度的灵活性。请参见本教程的各个章节以获得更多信息。

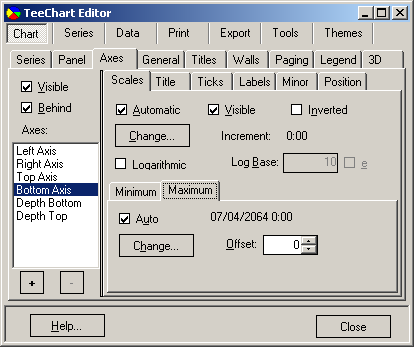
**内容**

[**坐标轴控件 － 关键区域**](#Bookmark5_AxesControl)  
     [刻度](#Bookmark5_Scales)   
     [增量](#Bookmark5_Increment)   
     [标题](#Bookmark5_Titles)   
     [标签](#Bookmark5_Labels)   
     [刻度标记](#Bookmark5_Ticks)   
     [坐标轴位置](#Bookmark5_PositionPercent)   
  
[**附加坐标轴**](#Bookmark5_AdditionalAxes)   
     [复制坐标轴](#Bookmark5_CustomDraw)   
     [多种坐标轴](#Bookmark5_MultiAxis)   
  
[**坐标轴事件**](#Bookmark5_AxisEvents)   
     [OnClickAxis](#Bookmark5_OnClickAxis)   
     [OnGetAxisLabel](#Bookmark5_OnGetAxisLabel)   
     [OnGetNextAxisLabel](#Bookmark5_OnGetNextAxisLabel)

**坐标轴控件 － 关键区域**

**刻度**   
当你添加数据序列到您的图表时，坐标轴的刻度会自动进行设置。您可在设计时或运行时通过坐标轴的属性来修改它们。  
  


非日期型数据

当添加一个新的数据序列时，在TeeChart编辑器Axis页的Scales部分将显示自动选择和其他灰色的选择。 所有值均为数字。  
  


日期型数据

在一个数据序列上设置一个日期型数据序列为True（该轴）的时候 -> 全部的页面,在TeeChart编辑器轴页的刻度部分将显示自动选择和其他灰色输出。所有值均为日期型。  
  
自动选择最佳的坐标轴刻度范围将会非常适合您的数据。如果您关闭自动选择，那么刻度的部分将变成不可选择项，这时您可以改变坐标轴的值。最重要的是，请您记住在当前页左边的列表框中选择您所想要进行设置的坐标轴。  
  
在设计时使用TeeChart编辑器添加一个线型数据序列到您的图表，然后用下面的代码添加一个命令按钮：

[C#.Net]   
Random rnd = new Random();   
for(int i = 0; i <= 40; ++i)   
line1.Add(Convert.ToDouble(i),rnd.Next(100),Color.Red);   
   
[VB.Net]   
Dim i As Integer   
For i = 0 To 40   
   Line1.Add(Convert.ToDouble(i), Rnd() \* 100, Color.Red)   
Next i

运行按钮中的代码将用40个随机值产生一个线型数据序列。 在设计时转到TeeChart编辑器。在Axis页的底轴刻度部分，将Automatic选项设为“off”，现在您就可以配置坐标轴刻度的最大和最小值。再次运行该代码将显示您为坐标轴设定的值。用鼠标的右键您可以滚动查看到剩余的值。

Setting axis scales by code

使用下面的代码您可以在运行时修改最大和最小值：

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.Automatic = false;   
bottomAxis.Maximum = 36;   
bottomAxis.Minimum = 5;   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Automatic = False   
   .Maximum = 36   
   .Minimum = 5   
End With

你可以分别设置坐标轴刻度最大和最小值的自动化属性。例如：

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.AutomaticMaximum = true;   
bottomAxis.AutomaticMinimum = false;   
bottomAxis.Minimum = 5;   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .AutomaticMaximum = True   
   .AutomaticMinimum = False   
   .Minimum = 5   
End With

**增量**   
您可以定制坐标轴的间隔。从Axis页刻度部分选择增量组合框并输入您想要的增量值。您也可以在运行时通过以下代码修改这个值：

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.Increment = 20;   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Increment = 20   
End With

Datetime data

如果您是日期型数据（您可以通过总体页的数据序列标签设置您的数据序列为日期型），转到Chart标签->Axis页，刻度部分将显示日期范围。从增量组合框所示增量的范围选择并添加一些示例数据：

[C#.Net]   
Random rnd = new Random();   
DateTime today = DateTime.Today;   
TimeSpan oneDay = TimeSpan.FromDays(1);   
line1.XValues.DateTime = true;   
for(int i = 1; i <= 25; ++i)   
     line1.Add(today,rnd.Next(100),Color.Red);   
     today += oneDay;   
   
   
[VB.Net]   
Dim i As Integer   
Dim Today As DateTime = DateTime.Today   
Dim OneDay As TimeSpan = TimeSpan.FromDays(1)   
Line1.XValues.DateTime = True   
For i = 1 To 25   
     Line1.Add(Today, Rnd() \* 100, Color.Red)   
     Today = Today.Add(OneDay)   
Next

在运行时修改增量值：

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.Increment = Steema.TeeChart.Utils.GetDateTimeStep(Steema.TeeChart.DateTimeSteps.TwoDays);   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Increment = Steema.TeeChart.Utils.GetDateTimeStep(Steema.TeeChart.DateTimeSteps.TwoDays)   
End With

了解更多有关日期坐标轴标签的信息，请查看AxisLabels.ExactDateTime。

Note

当频繁的修改坐标轴标签时，请记住TeeChart会根据AxisLabels.Separation属性的设置来避免标签的重叠。这意味着如果该标签被修改的频率高于标签所能接受的，那么TeeChart将为它分配最合适的那一个。 修改标签的角度和标签的separation两个属性可以帮助您找到最适合您的标签。请查看标签页以及AxisLabels.Angle属性。  
  
**标题**   
标题放置在Axis 页的Titles部分，您可以修改轴的标题文本以及文本的字体和阴影属性。标题文本的角度和大小都可以被指定。请在运行时查看AxisTitle类。  
  
**标签**   
从标签属性的摘要查看AxisLabels类。

Note

当频繁的修改坐标轴标签，请记住TeeChart会根据AxisLabels.Separation属性的设置来避免标签重叠的部分。这意味着如果该标签被修改的频率高于标签所能接受的，那么TeeChart将为它分配最合适的那一个。 修改标签的角度和标签的separation两个属性可以帮组您找到最适合您的标签。请查看AxisLabels.Angle属性。

Label formats

您可以将所有的标准数字和日期格式应用于坐标轴标签。在Axis页上，标签部分含有“Values format”域， 如果您的数据是日期型的，那么您可以将日期域的名字修改为“Date time format”。在运行时使用下面的代码：

[C#.Net]   
tChart1.Axes.Bottom.Labels.ValueFormat = "#,##0.00;(#,##0.00)";   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Labels.ValueFormat = "#,##0.00;(#,##0.00)"   
End With

日期型数据

[C#.Net]   
tChart1.Axes.Bottom.Labels.DateTimeFormat = "dddd/MMMM/yyyy";   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Labels.DateTimeFormat = "dddd/MMMM/yyyy"   
End With

MultiLine labels

坐标轴标签在显示时可以用多行文本来代替单行文本。使用LineSeparator character ()进行换行：  
  
例如

[C#.Net]   
bar1.Add(1234, "New" + Steema.TeeChart.Texts.LineSeparator + "Cars", Color.Red);   
bar1.Add(2000, "Old" + Steema.TeeChart.Texts.LineSeparator + "Bicycles", Color.Red);   
tChart1.Panel.MarginBottom = 10;   
   
[VB.Net]   
Bar1.Add(1234, "New" + Steema.TeeChart.Texts.LineSeparator + "Cars", Color.Red)   
Bar1.Add(2000, "Old" + Steema.TeeChart.Texts.LineSeparator + "Bicycles", Color.Red)   
TChart1.Panel.MarginBottom = 10

日期型标签示例：  
下面的代码将显示一个拥有两行文本的底轴标签，第一行显示月份和日期，第二行显示年份：  
  
Feb-28     Mar-1  ..   
2003        2003 ..

[C#.Net]   
bar1.Add(DateTime.Parse("28/2/2003"), 100, Color.Red);   
bar1.Add(DateTime.Parse("1/3/2003"), 200, Color.Red);   
bar1.Add(DateTime.Parse("2/3/2003"), 150, Color.Red);   
bar1.XValues.DateTime = true;   
tChart1.Axes.Bottom.Labels.DateTimeFormat = "MM/dd hh:mm";   
tChart1.Axes.Bottom.Labels.MultiLine = true;   
tChart1.Panel.MarginBottom = 10;       
   
[VB.Net]   
Bar1.Add(DateValue("28/2/2003"), 100, Color.Red)   
Bar1.Add(DateValue("1/3/2003"), 200, Color.Red)   
Bar1.Add(DateValue("2/3/2003"), 150, Color.Red)   
Bar1.XValues.DateTime = True   
TChart1.Axes.Bottom.Labels.DateTimeFormat = "MM/dd hh:mm"   
TChart1.Axes.Bottom.Labels.MultiLine = True   
TChart1.Panel.MarginBottom = 10

设置AxisLabels.MultiLine属性为True的时候，标签文本会在遇到空格时自动将文本换行，即将标签分成了两行：  
'mm/dd'为第一行  
'hh:mm'为第二行    
  
在运行时，您同样可以在OnGetAxisLabel事件中通过编程实现对标签的分行：

[C#.Net]   
private void tChart1\_GetAxisLabel(object sender, Steema.TeeChart.TChart.GetAxisLabelEventArgs e)   
           
            string myLabelText = e.LabelText;   
            tChart1.Axes.Bottom.Labels.SplitInLines(ref myLabelText, " ");   
            e.LabelText = myLabelText;   
           
   
[VB.Net]   
Private Sub TChart1\_GetAxisLabel(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetAxisLabelEventArgs) Handles TChart1.GetAxisLabel   
        Dim myLabelText As String   
        myLabelText = e.LabelText   
        TChart1.Axes.Bottom.Labels.SplitInLines(myLabelText, " ")   
        e.LabelText = myLabelText   
End Sub

在上面的例子中，“TeeSplitInLines”过程将标签文本中所有的空格都转换成行的分隔符（返回）。  
  
坐标轴的AxisLabels.Angle属性同样也可被用于多行坐标轴标签。

Customising Axis labels

可通过坐标轴事件使用标签控件的更多功能。这些事件允许您激活/停用/更改任何坐标轴标签。下面的示例将在该点索引值前面输入文本短语来修改每个标签：

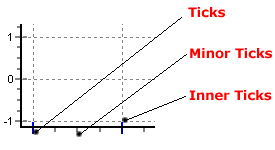
[C#.Net]   
private void button1\_Click(object sender, System.EventArgs e)   
           
            bar1.FillSampleValues(20);   
            tChart1.Axes.Bottom.Labels.Style = AxisLabelStyle.Mark;   
           
   
private void tChart1\_GetAxisLabel(object sender, Steema.TeeChart.TChart.GetAxisLabelEventArgs e)   
           
            if(((Steema.TeeChart.Axis)sender).Equals(tChart1.Axes.Bottom))   
            e.LabelText = "Period " + Convert.ToString(e.ValueIndex);   
           
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Bar1.FillSampleValues(20)   
        TChart1.Axes.Bottom.Labels.Style = Steema.TeeChart.AxisLabelStyle.Mark   
End Sub   
   
Private Sub TChart1\_GetAxisLabel(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetAxisLabelEventArgs) Handles TChart1.GetAxisLabel   
        If CType(sender, Steema.TeeChart.Axis) Is TChart1.Axes.Bottom Then   
            e.LabelText = "Period " & e.ValueIndex   
        End If   
End Sub

查看[坐标轴事件](#Bookmark5_AxisEvents) 的部分可以获取更多关于使用坐标轴事件来定制坐标轴标签的信息。

Logarithmic Labels

通过下面的方式设定一个标准的对数标签：

[C#.Net]   
private void button1\_Click(object sender, System.EventArgs e)   
           
            Random rnd = new Random();   
            Steema.TeeChart.Axis leftAxis = tChart1.Axes.Left;   
            tChart1.Aspect.View3D = false;   
            bar1.Marks.Visible = false;   
            for(int i = 0; i <= 100; ++i)   
            bar1.Add(rnd.Next(100) \* i);   
            leftAxis.LogarithmicBase = 10;   
            leftAxis.Logarithmic = true;   
            leftAxis.SetMinMax(0, 10000);   
            leftAxis.Labels.ValueFormat = "#e+0"; //exponential format    
           
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Dim i As Integer   
        TChart1.Aspect.View3D = False   
        Bar1.Marks.Visible = False   
        For i = 0 To 10000 Step 100   
            Bar1.Add(Rnd() \* i)   
        Next   
        With TChart1.Axes.Left   
            .LogarithmicBase = 10   
            .Logarithmic = True   
            .SetMinMax(0, 10000)   
            .Labels.ValueFormat = "#e+0" ' exponential format    
        End With   
End Sub

标签会根据对数的基数（默认10）而设置，因此在此例中显示的是1，10，100，1000，10000的标签。  
  
**刻度标记和分刻度**   
  
  
这里包含了3种刻度标记和2种网格。您可以修改刻度标记的长度，宽度，颜色以及网格类型。您可以通过“Ticks”标签修改刻度标记，它们相关联的网格和内部刻度标记；你还可以通过“Minor”标签修改分刻度标记和相关联的网格。新的TeeChart Pro 5版本可以改变刻度标记和网格样式，其宽度大于1（默认）。

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.Ticks.Length = 7;   
bottomAxis.Ticks.Color = Color.Green;   
bottomAxis.MinorTickCount = 10;       
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .Ticks.Length = 7   
   .Ticks.Color = Color.Green   
   .MinorTickCount = 10   
End With

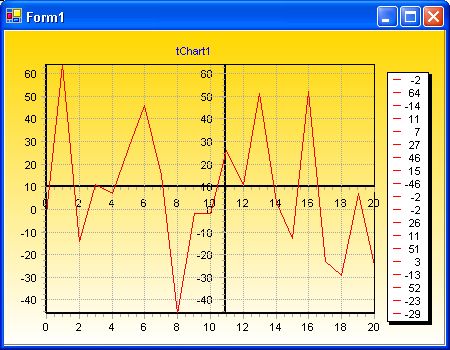
**坐标轴位置**   
坐标轴有一个属性可以对每个坐标轴的位置进行修改。在这个例子中，坐标轴移动了图表总宽度的50％，因此它在图表的中心位置进行显示：

[C#.Net]   
Steema.TeeChart.Axis bottomAxis = tChart1.Axes.Bottom;   
bottomAxis.PositionUnits = PositionUnits.Percent;   
bottomAxis. RelativePosition = 50   
   
[VB.Net]   
With TChart1.Axes.Bottom   
   .PositionUnits = PositionUnits.Percent   
   .RelativePosition = 50   
End With

**附加坐标轴**

**复制坐标轴**   
TeeChart提供了5种与数据序列相关联的坐标轴：左边，顶部，底部，右边和纵深。当您向图表添加一个新的数据序列时，您需要定义该数据序列和哪种类型的坐标轴相关联（在总体页上选择数据序列标签）。通过坐标轴的Customdraw方法，您可以复用图表上的任何地方前4个坐标轴中的任何一个（或全部）。请注意，这个方法只是复制了已经存在的坐标轴，并没有添加一个新的自定义数据序列。查看下个章节（自定义多坐标轴）获取更多信息。  
例如：

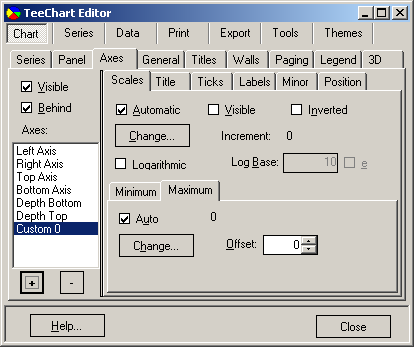
[C#.Net]   
private void Form1\_Load(object sender, System.EventArgs e)   
           
            Random Rnd = new Random();   
            tChart1.Aspect.View3D = false;   
            tChart1.Panel.Gradient.Visible = true;   
            for(int t = 0; t <= 20; ++t)   
            line1.Add(t, ((Rnd.Next(100)) + 1) - ((Rnd.Next(70)) + 1), Color.Red);   
   
           
   
private void line1\_BeforeDrawValues(object sender, Steema.TeeChart.Drawing.Graphics3D g)   
           
            int posAxis = 0;   
            if(tChart1.Axes.Left.Maximum > 0)   
               
                tChart1.Axes.Left.Draw(g.ChartXCenter - 10,g.ChartXCenter - 20,g.ChartXCenter,true);   
                posAxis = tChart1.Axes.Left.CalcYPosValue(10);   
                tChart1.Axes.Bottom.Draw(posAxis + 10, posAxis + 40, posAxis, true);   
               
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim t As Integer   
        TChart1.Aspect.View3D = False   
        TChart1.Panel.Gradient.Visible = True   
        For t = 0 To 20   
            Line1.Add(t, ((Rnd() \* 100) + 1) - ((Rnd() \* 70) + 1), Color.Red)   
        Next   
    End Sub   
   
Private Sub Line1\_BeforeDrawValues(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles Line1.BeforeDrawValues   
        Dim posAxis As Integer   
        If TChart1.Axes.Left.Maximum > 0 Then   
            TChart1.Axes.Left.Draw(g.ChartXCenter - 10, g.ChartXCenter - 20, g.ChartXCenter, True)   
            posAxis = TChart1.Axes.Left.CalcYPosValue(10)   
            TChart1.Axes.Bottom.Draw(posAxis + 10, posAxis + 40, posAxis, True)   
        End If   
End Sub

上面的实例代码所产生的效果如下图所示：  
  


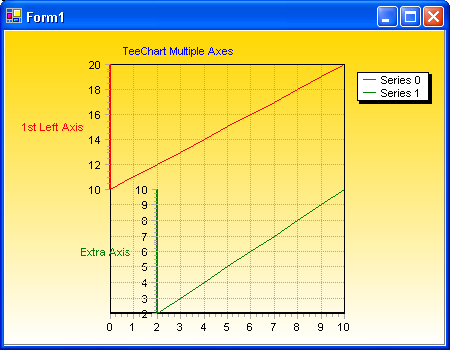
Custom axes

在这个例子中，TeeChart将在您的图表中心绘制新坐标轴，一个水平坐标轴，一个垂直坐标轴。当您滚动图表时 (使用鼠标右键拖拽)，新的垂直坐标轴将始终保持在图表中心，新的水平坐标轴将上升和垂直滚动。新的坐标轴是默认坐标轴的精确副本。  
  
**自定义多坐标轴**   
连同PositionPercent和stretching 属性，它不可能有浮动在图表上任意点的无限坐标轴。坐标轴的滚动，缩放和命中检测都使用用户自己创建的坐标轴。您既可以在设计时通过TeeChart编辑器来创建附加的坐标轴，也可以在运行时添加一些代码来创建附加的坐标轴：

Via the Chart Editor

  
  
在设计时TeeChart为您提供了创建自定义坐标轴的功能，而且它们能够以TeeChart的目标文件格式进行保存。 要做到这一点，首先打开图表编辑器并在Axis页选择“+”按钮来添加一个自定义坐标轴。在确保你已经拥有新自定义坐标轴后，选择Position标签，此页上的水平复选框允许你定义新的自定义坐标轴，它们可以是水平坐标轴或不选择该复选框让其默认为垂直坐标轴。本页面的其余部分和Axis页的其他标签可以像前面介绍的那样用来改变自定义坐标轴的刻度，增量，标题，标签，刻度标记，分刻度和位置。为了使新自定义坐标轴能够和数据序列相关联，您需要进入Series标签的General页，并通过其中的“Horizontal Axis”和“Vertical Axis”下拉列表框定义自定义坐标轴的具体位置，但是这需要根据先前定义新自定义坐标轴的水平或者垂直方向来决定。

Via Code  
[C#.Net]   
private void Form1\_Load(object sender, System.EventArgs e)   
           
            Line line1 = new Line();   
            Line line2 = new Line();    
   
            tChart1.Aspect.View3D = false;   
            tChart1.Panel.Gradient.Visible = true;   
            tChart1.Header.Text = "TeeChart Multiple Axes";   
            tChart1.Series.Add(line1);   
            tChart1.Series.Add(line2);   
   
            for(int t = 0; t <= 10; ++t)   
               
                line1.Add(Convert.ToDouble(t), Convert.ToDouble(10 + t), Color.Red);   
                if(t > 1)   
                line2.Add(Convert.ToDouble(t), Convert.ToDouble(t), Color.Green);   
               
   
            Axis leftAxis = tChart1.Axes.Left;   
   
            leftAxis.StartPosition = 0;   
            leftAxis.EndPosition = 50;   
            leftAxis.AxisPen.Color = Color.Red;   
            leftAxis.Title.Font.Color = Color.Red;   
            leftAxis.Title.Font.Bold = true;   
            leftAxis.Title.Text = "1st Left Axis";   
   
//            You are able to then position the new Axis in overall relation to the Chart    
//            by using the StartPosition and EndPosition  properties.   
//   
//            StartPosition=50   
//            EndPosition=100   
//   
//            These figures are expressed as percentages of the Chart Rectangle with 0 (zero)   
//            (in the case of a vertical Axis) being Top. These properties can be applied to    
//            the Standard Axes to create completely partitioned 'SubCharts' within the Chart.   
   
            Axis axis1 = new Axis(false, false, tChart1.Chart);   
   
            tChart1.Axes.Custom.Add(axis1);   
   
            line2.CustomVertAxis = axis1;   
   
            axis1.StartPosition = 50;   
            axis1.EndPosition = 100;   
            axis1.AxisPen.Color = Color.Green;   
            axis1.Title.Font.Color = Color.Green;   
            axis1.Title.Font.Bold = true;   
            axis1.Title.Text = "Extra Axis";   
            axis1.PositionUnits= PositionUnits.Percent;   
                                                axis1.RelativePosition = 20;   
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Line1 As New Steema.TeeChart.Styles.Line()   
        Dim Line2 As New Steema.TeeChart.Styles.Line()   
        Dim t As Integer   
   
        TChart1.Aspect.View3D = False   
        TChart1.Panel.Gradient.Visible = True   
        TChart1.Header.Text = "TeeChart Multiple Axes"   
        TChart1.Series.Add(Line1)   
        TChart1.Series.Add(Line2)   
   
        For t = 0 To 10   
            Line1.Add(t, 10 + t, Color.Red)   
            If (t > 1) Then   
                Line2.Add(t, t, Color.Green)   
            End If   
        Next   
   
        With TChart1.Axes.Left   
            .StartPosition = 0   
            .EndPosition = 50   
            .AxisPen.Color = Color.Red   
            .Title.Font.Color = Color.Red   
            .Title.Font.Bold = True   
            .Title.Text = "1st Left Axis"   
        End With   
   
        'You are able to then position the new Axis in overall relation to the Chart    
        'by using the StartPosition and EndPosition  properties.   
   
        '      StartPosition = 50   
        '      EndPosition = 100   
   
        'These figures are expressed as percentages of the Chart Rectangle with 0 (zero)   
        '(in the case of a vertical Axis) being Top. These properties can be applied to    
        'the Standard Axes to create completely partitioned 'SubCharts' within the Chart.   
   
        Dim Axis1 As New Steema.TeeChart.Axis(False, False, TChart1.Chart)   
   
        TChart1.Axes.Custom.Add(Axis1)   
   
        Line2.CustomVertAxis = Axis1   
   
        Axis1.StartPosition = 50   
        Axis1.EndPosition = 100   
        Axis1.AxisPen.Color = Color.Green   
        Axis1.Title.Font.Color = Color.Green   
        Axis1.Title.Font.Bold = True   
        Axis1.Title.Text = "Extra Axis"   
        Axis1.PositionUnits.=  PositionUnits.Percent;   
        Axis1.RelativePosition = 20   
End Sub

以上代码所产生的效果如下图所示：  
  


Multiple axes

所有操作是没有任何限制的！我们强烈建议您一定要谨慎使用自定义坐标轴，因为它很容易使用新的坐标轴来填充屏幕从而失去了您最希望跟踪管理的那个坐标轴!

**坐标轴事件**

坐标轴事件在运行时可以灵活地修改坐标轴标签并且可以通过点击坐标轴来增强当前用户的交互性。

**OnClickAxis**

查看OnClickAxis事件。   
  
例如：

[C#.Net]   
private void tChart1\_ClickAxis(object sender, System.Windows.Forms.MouseEventArgs e)   
           
            if(((Steema.TeeChart.Axis)sender).Equals(tChart1.Axes.Bottom))   
               
                MessageBox.Show("Clicked Bottom Axis at: " + line1.XScreenToValue(e.X));   
               
           
   
[VB.Net]   
Private Sub TChart1\_ClickAxis(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs) Handles TChart1.ClickAxis   
        If CType(sender, Steema.TeeChart.Axis) Is TChart1.Axes.Bottom Then   
            MsgBox("Clicked Bottom Axis at: " & Line1.XScreenToValue(e.X))   
        End If   
End Sub

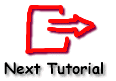
**OnGetAxisLabel**

用来修改坐标轴的标签。查看OnGetAxisLabel事件。  
  
例子：

[C#.Net]   
private void button1\_Click(object sender, System.EventArgs e)   
           
            bar1.FillSampleValues(20);   
            tChart1.Axes.Bottom.Labels.Style = AxisLabelStyle.Mark;   
           
   
private void tChart1\_GetAxisLabel(object sender, Steema.TeeChart.TChart.GetAxisLabelEventArgs e)   
           
            if(((Steema.TeeChart.Axis)sender).Equals(tChart1.Axes.Bottom))   
            e.LabelText = "Period " + Convert.ToString(e.ValueIndex);   
           
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Bar1.FillSampleValues(20)   
        TChart1.Axes.Bottom.Labels.Style = Steema.TeeChart.AxisLabelStyle.Mark   
End Sub   
   
Private Sub TChart1\_GetAxisLabel(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetAxisLabelEventArgs) Handles TChart1.GetAxisLabel   
        If CType(sender, Steema.TeeChart.Axis) Is TChart1.Axes.Bottom Then   
            e.LabelText = "Period " & e.ValueIndex   
        End If   
End Sub

**OnGetNextAxisLabel**   
它可用来判断显示哪个坐标轴标签。查看OnGetNextAxisLabel事件。 你应该使用e.Stop的Bool属性来包含/排除坐标轴标签。   
  
例如：

[C#.Net]   
private void Form1\_Load(object sender, System.EventArgs e)   
           
            line1.FillSampleValues(20);   
           
   
private void tChart1\_GetNextAxisLabel(object sender, Steema.TeeChart.TChart.GetNextAxisLabelEventArgs e)   
           
            if(((Steema.TeeChart.Axis)sender).Equals(tChart1.Axes.Bottom))   
               
                e.Stop = false;   
                switch(e.LabelIndex)   
                   
                    case 0: e.LabelValue = 5; break;   
                    case 1: e.LabelValue = 13; break;   
                    case 2: e.LabelValue = 19; break;   
                    default: e.Stop = true; break;   
                   
               
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Line1.FillSampleValues(20)   
End Sub   
   
Private Sub TChart1\_GetNextAxisLabel(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetNextAxisLabelEventArgs) Handles TChart1.GetNextAxisLabel   
        If CType(sender, Steema.TeeChart.Axis) Is TChart1.Axes.Bottom Then   
            e.Stop = False   
            Select Case e.LabelIndex   
                Case 0 : e.LabelValue = 5   
                Case 1 : e.LabelValue = 13   
                Case 2 : e.LabelValue = 19   
                Case Else : e.Stop = True   
            End Select   
        End If   
End Sub

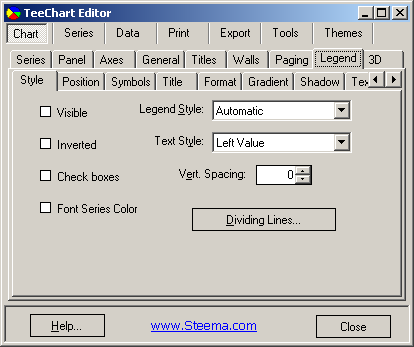
[](#Bookmark4)(See 1.1.3)    [](#Bookmark6)(See 1.1.5)

### 1.1.5 教程5 － 图例设计

**教程5 － 图例设计   
目录**

[**图例控件**](#Bookmark6_LegendControl)   
     [样式标签](#Bookmark6_Style)   
     [定位](#Bookmark6_Position)[标签](#Bookmark6_Style)   
     [符号](#Bookmark6_Symbols)[标签](#Bookmark6_Style)   
     [格式标签](#Bookmark6_Format)   
     [文本标签](#Bookmark6_Text)   
     [渐变标签](#Bookmark6_Gradient)   
     [阴影标签](#Bookmark6_Shadow)   
     [边框纵深梯度标签](#Bookmark6_Bevel)   
  
[**自定义图例**](#Bookmark6_Customising)   
     [OnGetLegendRect 事件](#Bookmark6_OnGetLegendRect)   
     [OnGetLegendPos 事件](#Bookmark6_OnGetLegendPos)   
     [OnGetLegendText 事件](#Bookmark6_OnGetLegendText)

**图例控件**

通过TeeChart编辑器、图表标签、图例页面可以设置图例的各种参数。  
  
  
  
图例参数。有关更多信息，请参考图例类的帮助文档。  
  
**样式标签**

图例样式

当图表中只有唯一的一个数据序列时，图例默认的“Automatic”样式将会把数据序列的值放置到图例中。当图表中所包含的数据序列不止一个时，“Automatic”样式将会把数据序列的名称放置到图例中。在编辑器中可使用下拉列表框获取除默认样式以外的其它样式。如果您修改了图例样式来显示数值而您图表中的数据序列又不止一个，那么TeeChart Pro 将显示的是第一个数据序列的值。同样您可以通过自定义操作来修改显示的方式。请查看[自定义图例 。](#Bookmark6_Customising)   
  
把每个数据序列的最后一个值放入图例框中：

[C#]   
tChart1.Legend.LegendStyle = LegendStyles.LastValues;   
   
[VB.Net]   
TChart1.Legend.LegendStyle = Steema.TeeChart.LegendStyles.LastValues

文本样式

参考图例文本样式列表中的文本样式属性。

垂直间距

允许您修改两个图例项的间距。

分隔线

“Dividing Lines” 按钮可以进入画笔编辑器窗口，在这里您可以设置图例项之间的分隔线为可见以及它的颜色，样式和宽度。

可见性

“Visible”复选框可以启用/禁用图例项之间的分隔线。

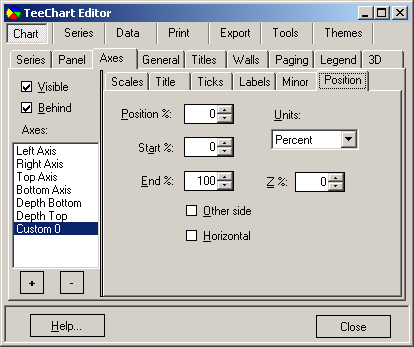
逆序

这个功能使图例项的顺序变成倒序。

复选框

将图例的样式修改为“Series Names”，这样能使每个数据序列旁边显示一个复选框，当数据序列的复选框选中时则该数据序列是可见的，否则数据序列将被隐藏。

数据序列的字体颜色

将图例文本的字体颜色修改成数据序列的颜色。  
  
**定位标签**  


定位

定位标签中带有4个可选择的属性：顶部，底部，左和右。默认位置是右。图例的默认定位指的是图表的外部。 请查看[自定义图例](#Bookmark6_Customising)获取更多关于图例的信息。

调整图表大小

如果没有启用调整图表大小属性，图例会绘制在整个图表框架区内。虽然这可能是一些令人满意的图例定位要求，更好地控制图例相关图表框架的定位可以通过使用图例的Margin属性来实现。

边距

修改边距属性可以移动图例在图表框架中的位置，反之亦然。但是，您应该确保边距属性的值不会让图例在图表中不能正常的显示（增加图表长方形的面积大小）。然而，这些属性是不能用来重新定位图表中图例的位置，最好使用[自定义图例](#Bookmark6_Customising) 中介绍的方法。

定义偏移百分比

0%意味着图例的水平方向与图表的右边框相接，图例的垂直方向与图表的顶部相接。图例也是参照这些位置。

自定义

此复选框将禁用调整图表大小的属性，并让您可以在像素级别将图例定位在图表的任意位置。  
  
**符号标签**

宽度

设置宽度属性来定义彩色矩形（符号）的宽度。

宽度单位

使用WidthUnits属性来确定宽度的单位。

定位

使用定位属性来设置带彩色矩形的位置。

连续性

使用连续属性使得不同图例的的矩形颜色相互连接起来。当其设置为True时，不同单元的彩色矩形可固定在一起（无垂直间距），当其设置为False时，彩色矩形会作为单独的矩形。   
  
**格式标签**

颜色

颜色属性是用来设置图例的背景颜色。

框架

设置图例的框架属性：可见性，样式，颜色和宽度[画笔属性]。

图案

设置图例背景图案属性：填充，纹理，渐变，图片[纹理刷属性]。

圆角框架

将图例框架的角变成圆滑的。

透明

使得图例的背景变为透明的，只显示图例的文本和符号。

透明度

当透明度设置为True的时候，您可以使用透明度百分比来设置图例的透明程度。  
  
**文本标签**   
此页可以修改图例文字的字体外观，字体填充[纹理刷属性]和阴影等属性。  
  
**渐变标签**   
使用此页可以在图例的背景上绘制一种渐变效果并且可以修改此渐变的方向和颜色。  
  
**阴影标签**   
设置图例阴影的颜色，大小，图案和透明。  
  
**边框纵深梯度标签**   
设置图例内外边框纵深梯度的颜色，大小和样式。

**自定义图例**

图例事件提供了完整地定义图例的外观和内容的操作。  
  
**OnGetLegendRect 事件**   
图例的外矩形框允许修改整体的大小和图例框的位置。与OnGetLegendPos结合使用可以重新定位图表的图例和内容。  
例子：

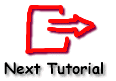
[C#]   
private void tChart1\_GetLegendRect(object sender, Steema.TeeChart.TChart.GetLegendRectEventArgs e)   
        {   
            e.Rectangle=new Rectangle(e.Rectangle.Left-100,e.Rectangle.Top,e.Rectangle.Width,e.Rectangle.Height);    
        }   
   
[VB.Net]   
Private Sub TChart1\_GetLegendRect(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetLegendRectEventArgs) Handles TChart1.GetLegendRect   
        e.Rectangle = New Rectangle(e.Rectangle.Left - 100, e.Rectangle.Top, e.Rectangle.Width, e.Rectangle.Height)   
End Sub

**OnGetLegendPos 事件**   
修改图例的内容。下面例子中所使用的代码是将图例的内容移动到一个新的图例矩形框中。

[C#]   
private void tChart1\_GetLegendPos(object sender, Steema.TeeChart.TChart.GetLegendPosEventArgs e)   
        {   
            e.X = e.X-100;   
            e.XColor = e.XColor-100;   
        }   
   
[VB.Net]   
Private Sub TChart1\_GetLegendPos(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetLegendPosEventArgs) Handles TChart1.GetLegendPos   
        e.X = e.X - 100   
        e.XColor = e.XColor - 100   
End Sub

**OnGetLegendText 事件**   
修改图例的文本内容。

[C#]   
private void tChart1\_GetLegendText(object sender, Steema.TeeChart.TChart.GetLegendTextEventArgs e)   
        {   
            if(e.Index == 3)   
            {e.Text = e.Text + " Index no.:" + e.Index.ToString();}   
        }   
   
[VB.Net]   
Private Sub TChart1\_GetLegendText(ByVal sender As Object, ByVal e As Steema.TeeChart.TChart.GetLegendTextEventArgs) Handles TChart1.GetLegendText   
        If e.Index = 3 Then   
            e.Text = e.Text + " Index no.:" + e.Index.ToString   
        End If   
End Sub

当您把图例放置到图表的矩形区域时，请记住，图例会在数据序列和坐标轴之前进行绘制，并且在任何交叉点图例都会出现在它们的下面。  
  
[](#Bookmark5)(See 1.1.4)     [](#Bookmark7)(See 1.1.6)

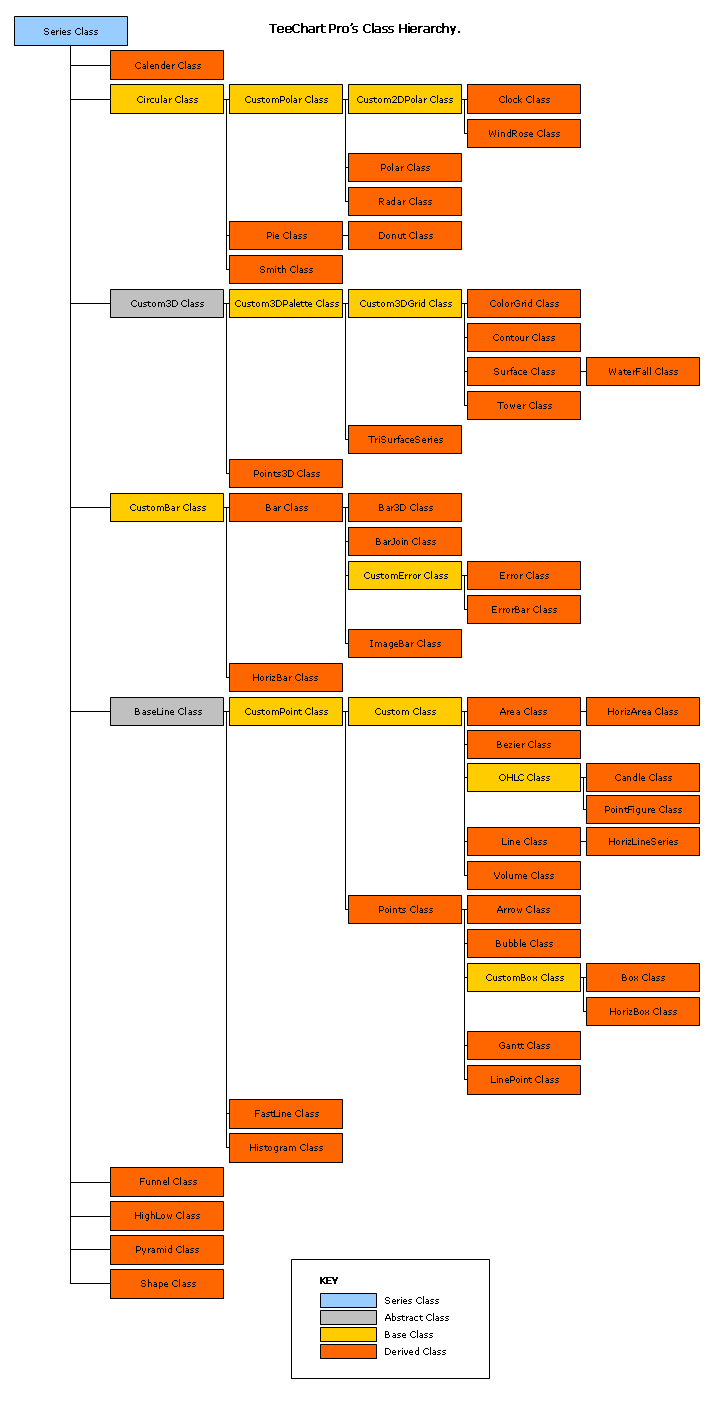
### 1.1.6 教程6 － 使用数据序列

**教程6 － 使用数据序列  
目录**

[**数据序列类型**](#Bookmark7_SeriesTypes)   
     [Series类结构](#Bookmark7_SeriesHierarchy)  
     [选择一种数据序列类型](#Bookmark7_ChooseSeries)  
     [向数据序列添加数据](#Bookmark7_AddToSeries)  
     [从数据序列中删除数据点](#Bookmark7_DeleteFromSeries)  
     [向数据序列中添加一个空点](#Bookmark7_NullPoints)  
  
[**在图表上混合数据序列类型**](#Bookmark7_MixingSeries)  
     [添加新的数据序列](#Bookmark7_AddSeries)  
     [为数据序列选择坐标轴](#Bookmark7_ChooseAxis)  
     [连接数据序列](#Bookmark7_ConnectSeries)   
     [修改数据序列顺序](#Bookmark7_ChangeSeriesOrder)  
  
[**数据序列的Value列表**](#Bookmark7_Valuelist)   
     [访问数据序列的值](#Bookmark7_AccessValues)   
     [使用Values举例](#Bookmark7_EgValues)   
  
[**数据序列事件**](#Bookmark7_SeriesEvents)   
     [OnClickSeries](#Bookmark7_ClickSeries)   
     [OnGetSeriesPointerStyle](#Bookmark7_GetPointer)   
     [OnGetSeriesMark](#Bookmark7_GetMarkText)

**数据序列类型**

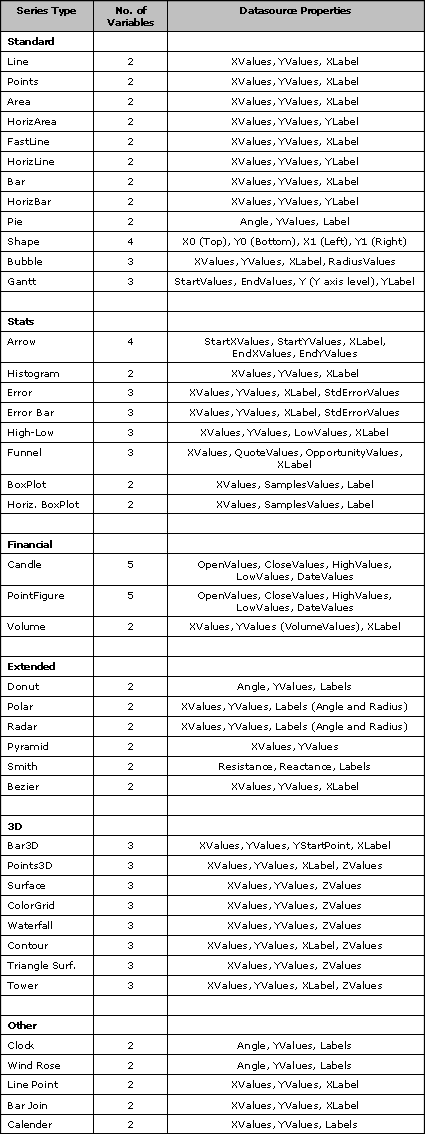
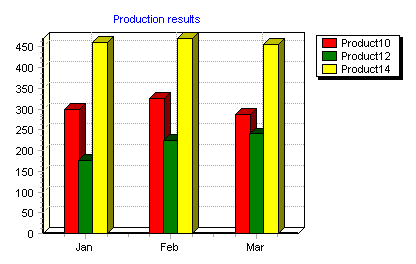
TChart数据序列类是所有数据序列类型的共有原型。当使用TeeChart在线帮助来获取任何数据序列类型的帮助时，请转到数据序列类在继承类型列表的链接，然后点击数据序列的成员，在那里包含了所有继承的属性和方法清单。

**Series类结构**以TeeChart类型库结构作为背景，这是数据序列类和接口的说明。下面的图解展示了TeeChart数据序列类间的关系。所有类都来自共同的“Series”类，因此它们也共享“Series”的属性和方法。几个抽象类派生自数据序列（Custom3DSeries，CustomBarSeries和CircledSeries），它们都是用灰色突出显示的，强调了它们的接口是不能直接进行编程的，其特点是只能被他们的后代数据序列类型所继承。所有派生的数据序列（橙色）都当作您图表的一部分，从而很容易取得TeeChart集合。以这种方式生成的TeeChart数据序列，允许通过一个共同的索引结构以可编程的方式访问继承的属性和方法（见本节中后面的示例代码）。  
  
  
在设计时使用TChart编辑器可轻松地添加数据序列，在运行时您同样可以向同一个TChart创建和添加新的不同类型的数据序列。

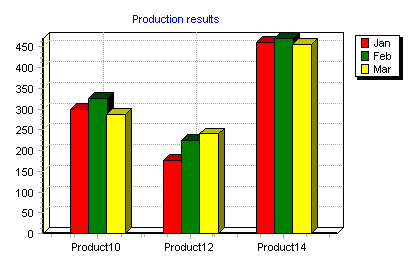
[C#]   
//Add a series at runtime   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            Steema.TeeChart.Styles.Area tmpAreaSeries = new Steema.TeeChart.Styles.Area(tChart1.Chart);    
            tmpAreaSeries.FillSampleValues(4);   
            //Or   
            //Steema.TeeChart.Styles.Area tmpAreaSeries = new Steema.TeeChart.Styles.Area();   
            //tChart1.Series.Add(tmpAreaSeries);   
            //tmpAreaSeries.FillSampleValues(4);   
        }   
   
[VB.Net]   
'Add a series at runtime   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
         Dim tmpAreaSeries As New Steema.TeeChart.Styles.Area(TChart1.Chart)   
        tmpAreaSeries.FillSampleValues(4)   
        'Or   
        'Dim tmpAreaSeries As New Steema.TeeChart.Styles.Area()   
        'TChart1.Series.Add(tmpAreaSeries)   
        'tmpAreaSeries.FillSampleValues(4)   
End Sub

在设计时，所有AreaSeries属性和方法可用于创建新的数据序列，它们可与任何数据序列相关联。  
  
在设计时，一个在同一图表中混合了不同序列类的例子，它是使用TeeChart编辑器将面积（数据序列（0）），柱形（数据序列（1））和线型（数据序列（2））数据序列添加到一个图表。所有数据序列都访问一个共同的索引结构，即图表数据序列的列表。若要与数据序列一起工作，可能应像下面这样进行操作：

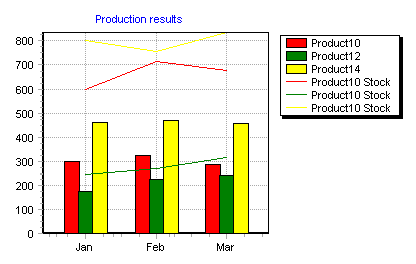
[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        {   
            //You could add the Series at runtime    
            Steema.TeeChart.Styles.Area area1 = new Steema.TeeChart.Styles.Area(tChart1.Chart);   
            Steema.TeeChart.Styles.Bar bar1 = new Steema.TeeChart.Styles.Bar(tChart1.Chart);   
            Steema.TeeChart.Styles.Line line1 = new Steema.TeeChart.Styles.Line(tChart1.Chart);   
   
            //Use Series common properties    
            tChart1.Series[0].FillSampleValues(10);   
            tChart1.Series[1].FillSampleValues(10);   
            tChart1.Series[2].FillSampleValues(10);   
            tChart1.Series[1].Marks.Visible = false;   
            tChart1.Series[2].Marks.Visible = false;   
   
            //Modify Bar specific properties    
            bar1.BarStyle = Steema.TeeChart.Styles.BarStyles.Pyramid; //Change Bar type    
            bar1.Pen.Color = Color.Yellow; //Bar bounding lines colour    
   
            //Modify Line specific properties    
            line1.Stairs = true; //Set line to Stairs    
            line1.LinePen.Color = Color.Blue; //LineSeries bounding lines colour    
   
            //Modify Area specific properties    
            area1.AreaBrush.Style = System.Drawing.Drawing2D.HatchStyle.Cross; //Area fill pattern    
        }   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        'You could add the Series at runtime    
        Dim Area1 As New Steema.TeeChart.Styles.Area(TChart1.Chart)   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Dim Line1 As New Steema.TeeChart.Styles.Line(TChart1.Chart)   
   
        'Use Series common properties    
        TChart1.Series(0).FillSampleValues(10)   
        TChart1.Series(1).FillSampleValues(10)   
        TChart1.Series(2).FillSampleValues(10)   
        TChart1.Series(1).Marks.Visible = False   
        TChart1.Series(2).Marks.Visible = False   
   
        'Modify Bar specific properties    
        Bar1.BarStyle = Steema.TeeChart.Styles.BarStyles.Pyramid 'Change Bar type    
        Bar1.Pen.Color = Color.Yellow 'Bar bounding lines colour    
   
        'Modify Line specific properties    
        Line1.Stairs = True 'Set line to Stairs    
        Line1.LinePen.Color = Color.Blue 'LineSeries bounding lines colour    
   
        'Modify Area specific properties    
        Area1.AreaBrush.Style = System.Drawing.Drawing2D.HatchStyle.Cross 'Area fill pattern    
End Sub

**选择一个数据序列类型**为一个图表选择一个数据序列类型是依赖于您自己对图表的要求。但是在某些情况下，由于变量的原因，对图表的选择可能取决于数据序列类型是否支持输入变量的数值。下表显示了每个数据序列类型所允许的变量数值。  
  
  
  
标签可以用来扩展一个含有两个变量的数据序列类型的值。看下面的例子，它在同一个图表中使用了柱形数据序列类型的3大实例。  
**例子：**   
使用柱形数据序列类型  
  
Product code     Month     Quantity produced   
10                    Jan         300   
10                    Feb         325   
10                    Mar         287   
12                    Jan         175   
12                    Feb         223   
12                    Mar         241   
14                    Jan         461   
14                    Feb         470   
14                    Mar         455   
  
通过最简单的形式，数据产生按月分组的资料，如下图所示：  
  
代码：

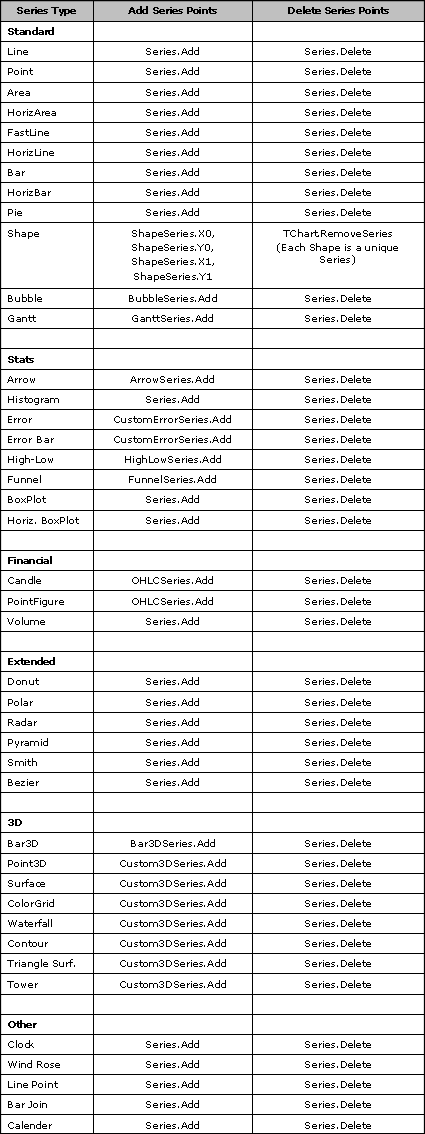
[C#]   
foreach(Steema.TeeChart.Styles.Series tSeries in tChart1.Series)   
{tSeries.Marks.Visible = false;}   
tChart1.Header.Text = "Production results";   
   
bar1.Add(300,"Jan");   
bar1.Add(325,"Feb");   
bar1.Add(287,"Mar");   
bar1.Title = "Product10";   
   
bar2.Add(175,"Jan");   
bar2.Add(223,"Feb");   
bar2.Add(241,"Mar");   
bar2.Title = "Product12";   
   
bar3.Add(461,"Jan");   
bar3.Add(470,"Feb");   
bar3.Add(455,"Mar");   
bar3.Title = "Product14";   
   
[VB.Net]   
Dim TSeries As Steema.TeeChart.Styles.Series   
For Each TSeries In TChart1.Series   
    TSeries.Marks.Visible = False   
Next   
TChart1.Header.Text = "Production results"   
   
Bar1.Add(300, "Jan")   
Bar1.Add(325, "Feb")   
Bar1.Add(287, "Mar")   
Bar1.Title = "Product10"   
   
Bar2.Add(175, "Jan")   
Bar2.Add(223, "Feb")   
Bar2.Add(241, "Mar")   
Bar2.Title = "Product12"   
   
Bar3.Add(461, "Jan")   
Bar3.Add(470, "Feb")   
Bar3.Add(455, "Mar")   
Bar3.Title = "Product14"

或者 (根据product分组）：   
  
代码：:

[C#]   
foreach(Steema.TeeChart.Styles.Series tSeries in tChart1.Series)   
{tSeries.Marks.Visible = false;}   
tChart1.Header.Text = "Production results";   
   
bar1.Add(300,"Product10");   
bar1.Add(175,"Product12");   
bar1.Add(461,"Product14");   
bar1.Title = "Jan";   
   
bar2.Add(325,"Product10");   
bar2.Add(223,"Product12");   
bar2.Add(470,"Product14");   
bar2.Title = "Feb";   
   
bar3.Add(287,"Product10");   
bar3.Add(241,"Product12");   
bar3.Add(455,"Product14");   
bar3.Title = "Mar";   
   
[VB.Net]   
Dim TSeries As Steema.TeeChart.Styles.Series   
For Each TSeries In TChart1.Series   
    TSeries.Marks.Visible = False   
Next   
TChart1.Header.Text = "Production results"   
   
Bar1.Add(300, "Product10")   
Bar1.Add(175, "Product12")   
Bar1.Add(461, "Product14")   
Bar1.Title = "Jan"   
   
Bar2.Add(325, "Product10")   
Bar2.Add(223, "Product12")   
Bar2.Add(470, "Product14")   
Bar2.Title = "Feb"   
   
Bar3.Add(287, "Product10")   
Bar3.Add(241, "Product12")   
Bar3.Add(455, "Product14")   
Bar3.Title = "Mar"

我们增加了新的值到上述表中（Stock）。  
  
Product code     Month     Quantity produced     Stock level   
10                    Jan         300                          600   
10                    Feb         325                          715   
10                    Mar         287                          676   
12                    Jan         175                          245   
12                    Feb         223                          270   
12                    Mar         241                          315   
14                    Jan         461                          800   
14                    Feb         470                          755   
14                    Mar         455                          835   
  
表中Stock的值比每月生产的大体上高一点，以便于它们在下图中显示（这次使用2D）。图表使用线型数据序列来区分Stocks。  
  
代码：:   
添加以下内容到先前的第一个代码示例中：

[C#]   
line1.Add(600,"Jan");   
line1.Add(715,"Feb");   
line1.Add(676,"Mar");   
line1.Title = "Product10 Stock";    
line1.Color = bar1.Color;   
   
line2.Add(245,"Jan");   
line2.Add(270,"Feb");   
line2.Add(315,"Mar");   
line2.Title = "Product10 Stock";     
line2.Color = bar2.Color;   
   
line3.Add(800,"Jan");   
line3.Add(755,"Feb");   
line3.Add(835,"Mar");   
line3.Title = "Product10 Stock";     
line3.Color = bar3.Color;   
   
[VB.Net]   
Line1.Add(600, "Jan")   
Line1.Add(715, "Feb")   
Line1.Add(676, "Mar")   
Line1.Title = "Product10 Stock"   
Line1.Color = Bar1.Color   
   
Line2.Add(245, "Jan")   
Line2.Add(270, "Feb")   
Line2.Add(315, "Mar")   
Line2.Title = "Product10 Stock"   
Line2.Color = Bar2.Color   
   
Line3.Add(800, "Jan")   
Line3.Add(755, "Feb")   
Line3.Add(835, "Mar")   
Line3.Title = "Product10 Stock"   
Line3.Color = Bar3.Color

**向数据序列添加数据**大多数据序列类型（除了[教程08](#Bookmark9)(See 1.1.8) ADO.NET数据库访问和[教程07](#Bookmark8)(See 1.1.7)使用函数）使用24类Add的重载方法来添加数据。这里会有一些例外，参考下面的表格：   
  
  
  
请注意，除图形数据序列外，所有数据序列特定的Add方法就像普通Add方法一样，都会被自动添加和访问（如：candleSeries1.Add(new DateTime(2002,11,27),100,400,200,300)；）。  
    
**颜色**   
在添加一个点时，您可以手动为点添加颜色。  
例子：

[C#]   
bar1.Add(50,"Tomatoes",Color.Tomato);   
[VB.Net]   
Bar1.Add(50, "Tomatoes", Color.Tomato)

或者，您可以允许TeeChart分配一个颜色。假如Series.ColorEach被设置为Ture的时候，TeeChart将从多达19种独特且尚未使用的颜色中，为每个新数据序列或每个新数据序列点挑选一种颜色。  
例子：

[C#]   
Random rnd = new Random();   
bar1.ColorEach = true;   
for(int i = 0; i < 19; ++i)   
{   
     int higher = i + 65;   
     char letter = (char) higher;   
     bar1.Add(rnd.Next(100),letter.ToString());   
}   
[VB.Net]   
Dim i As Integer   
Bar1.ColorEach = True   
For i = 0 To 19   
    Bar1.Add(Rnd() \* 100, Chr(i + 65))   
Next

一个透明的颜色可能会被添加到一个点，以保留ValueList值的空间，虽然它没有在图表上显示出来。   
例子：

[C#]   
bar1.Add(45, "My Transparent Bar", Color.Transparent);   
[VB.Net]   
Bar1.Add(45, "My Transparent Bar", Color.Transparent)

**从数据序列中删除数据点**  
使用Series.Delete可以从一个数据序列中删除一个点。Series.Delete拥有两个重载的方法：

public Void Delete(System.Int32)

从数据序列中删除某个特定（nth）的点。

public Void Delete(System.Int32, System.Int32)

从某个（nth）点开始删除数据序列中的多个（n）点。

例子：

[C#]   
bar1.Delete(7,2);  (deletes two points starting from the 8th Series point (index starts at zero))   
[VB.Net]   
Bar1.Delete(7, 2)  (deletes two points starting from the 8th Series point (index starts at zero))

Series.Clear可以清除数据序列中所有的点。  
  
**向数据序列中添加一个空点**   
Series.Add拥有三个重载的方法：它们允许您向数据序列添加一个空点：  
添加一个新的空（不可见的）点。  
public Int32 Add()   
  
使用特定的文本添加一个新的空点。  
public Int32 Add(System.String)   
  
指向一个特定X值使用文本添加一个新的空点。  
public Int32 Add(System.Double, System.String)

上述第二个重载方法将向数据序列增加一个空点，使您可以定义这个点的标签，但在数据序列那个点的位置会留下一条缝。对于线型数据序列，缝之前的最后一个点将不会加在缝后面的第一个点。  
例子：

[C#]   
line1.Add("Null Point");   
[VB.Net]   
Line1.Add("Null Point")

请在TeeChart帮助文件中查看另外两个重载方法的示例。

**在图表中混合数据序列类型**

TeeChart Pro提供一个类似数据序列背景的空图表背景。这意味着没有图表的类型是预先定义好的。你可以定义您所需的图表类型，例如您想要显示混合的数据序列类型。由于一些数据序列类型的专业性，它们是不能和其它的数据序列混合在一起的，当你在添加一个新数据序列时，TeeChart会在图表集中帮助您去除那些不适合的数据序列。在这里，您向一个图表中添加数据序列的数目是没有实际限制的。  
  
**添加新的数据序列**   
您可以使用TeeChart编辑器（参阅[教程01](#Bookmark2)(See 1.1.1))或者代码来添加数据序列。  
例子：

[C#]   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            Bar bar1 = new Bar(tChart1.Chart);   
            bar1.FillSampleValues(10);   
        }   
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Bar1.FillSampleValues(10)   
End Sub

数据序列被添加到SeriesList，您可以通过Index来访问它们（TChart1.Series（Index）），第一个数据序列的Index是0。TeeChart Pro会为数据序列添加一个默认的名称（Series 0，Series 1等）。您可以通过Series.Title属性来修改这个名称。   
  
**为数据序列选择坐标轴**  
数据序列添加到图表后会自动使用左坐标轴和底坐标轴来作为参考坐标轴。您可以在图表编辑器中选择Series General页的相关主题来修改参考坐标轴，这里有四种可用的坐标轴：顶部，左，底部和右。下面所示的是通过代码来修改参考坐标轴：

[C#]   
bar1.VertAxis = Steema.TeeChart.Styles.VerticalAxis.Right;   
bar1.HorizAxis = Steema.TeeChart.Styles.HorizontalAxis.Top;   
[VB.Net]   
Bar1.VertAxis = Steema.TeeChart.Styles.VerticalAxis.Right   
Bar1.HorizAxis = Steema.TeeChart.Styles.HorizontalAxis.Top

每个坐标轴可能会相关联1个以上的数据序列。TeeChart会决定最佳的刻度来适应坐标轴所匹配的数据序列，同样的，您自己也可以改变坐标轴的刻度（参阅[坐标轴的教程](#Bookmark5_Scales)(See 1.1.4)）。附加坐标轴可能会增加，他们将复制与前4个坐标轴相关联的坐标轴尺度（参阅[附加坐标轴](#Bookmark5_AdditionalAxes)(See 1.1.4)）。  
  
**连接数据序列**  
你可以使用其它数据序列的数据源来作为一个数据序列。您可以使用图表编辑器来设置第二个数据序列的数据源。转到Series标签，Datasource页。选择“Function”作为数据源的类型，出现两个列表框：可用数据序列和已选择数据序列。选择您希望使用的数据序列作为当前数据序列的数据源，然后在Combobox上方的Functions:中选择Average作为函数的类型，最后点击Apply按钮。请注意，通过这种方式，任何数据序列都可以被定义为其它数据序列的函数，函数类型则是函数列表框中所提供的所有可用的函数类型。下面的代码可以做同样的事情：

[C#]   
Steema.TeeChart.Functions.Average average1 = new Steema.TeeChart.Functions.Average();   
line1.Function = average1;   
line1.DataSource = bar1;   
bar1.FillSampleValues(10);   
line1.CheckDataSource();   
   
[VB.Net]   
Dim Average1 As New Steema.TeeChart.Functions.Average()   
Line1.Function = Average1   
Line1.DataSource = Bar1   
Bar1.FillSampleValues(10)   
Line1.CheckDataSource()

如何使用TeeChart的函数，请参阅[教程07 － 使用函数](#Bookmark8) (See 1.1.7)获取更多相关信息。  
  
**修改数据序列顺序**   
使用图表编辑器可轻松地修改数据序列的顺序。转到编辑器的首页，并突出显示您想要移动的数据序列。使用右边的箭头来向上或向下移动数据序列的顺序。数据序列的顺序将决定它相对于其它数据序列在图表中显示的相对位置。如果数据序列的Active设置为False时，那么数据序列会被图表隐藏起来，但是它所包含的数据内容还是完整的。  
通过对Series.Exchange编码来修改数据序列的顺序。

[C#]   
tChart1.Series.Exchange(0, 1);  //Change Series(0) with Series(1) in the index order   
[VB.Net]   
TChart1.Series.Exchange(0, 1) 'Change Series(0) with Series(1) in the index order

\*注意：交换数据序列的顺序后，数据序列的Index也会被修改。因此，上面的代码行完全修改了Series '0'和Series '1'，如果代码返回0则变成了1，1变成了0。

**数据序列的Value列表**

TeeChart数据序列将他们的值存储在Valuelist，并通过ValueList类来访问和修改这些值。  
  
**访问数据序列的Values**   
您可以访问列表中的任何值。  
例子：

[C#]   
MessageBox.Show(bar1.YValues[3].ToString()); //Displays value of 4th point (index starts at 0) of a BarSeries   
[VB.Net]   
MsgBox(Bar1.YValues(3)) 'Displays value of 4th point (index starts at 0) of a BarSeries

下面这种方式访问的值可能会被用来在数据序列数据上设置跟踪标记：

[C#]   
for(int i = 0; i < bar1.Count; ++i)   
            {   
                if(bar1.YValues[i] > 500)   
                {   
                    MessageBox.Show("Value: (" + bar1.XValues[i] + ", " + bar1.YValues[i] + ") exceeds limit");   
                }   
            }   
[VB.Net]   
Dim i As Integer   
For i = 0 To Bar1.Count   
    If Bar1.YValues(i) > 500 Then   
        MsgBox("Value: (" & Bar1.XValues(i) & ", " & Bar1.YValues(i) & ") exceeds limit")   
    End If   
Next

同样的值可以通过在一些数据序列方法和图表事件才使用点的ValueIndex来获取。  
例子：

[C#]   
private void tChart1\_ClickSeries(object sender, Steema.TeeChart.Styles.Series s, int valueIndex, System.Windows.Forms.MouseEventArgs e)   
        {   
            if(s.Equals(bar1))   
            {   
                MessageBox.Show("ValueIndex is: " + valueIndex.ToString());   
                MessageBox.Show("Point's YValue is " + bar1.YValues[valueIndex].ToString());   
            }   
        }   
   
[VB.Net]       
Private Sub TChart1\_ClickSeries(ByVal sender As Object, ByVal s As Steema.TeeChart.Styles.Series, ByVal valueIndex As Integer, ByVal e As System.Windows.Forms.MouseEventArgs) Handles TChart1.ClickSeries   
        If s Is Bar1 Then   
            MsgBox("ValueIndex is: " & valueIndex)   
            MsgBox("Point's YValue is " & Bar1.YValues(valueIndex))   
        End If   
End Sub

**使用Values举例**  
下面这些代码是根据用户鼠标的点击来修改柱形数据序列的值。  
例子：   
  
使用TChart.ClickSeries事件来判断用户点击的地方。

[C#]   
private void tChart1\_ClickSeries(object sender, Steema.TeeChart.Styles.Series s, int valueIndex, System.Windows.Forms.MouseEventArgs e)   
        {   
            UpDatePoint(valueIndex,tChart1.Axes.Left.CalcPosPoint((e.Y)));   
        }   
   
[VB.Net]   
Private Sub TChart1\_ClickSeries(ByVal sender As Object, ByVal s As Steema.TeeChart.Styles.Series, ByVal valueIndex As Integer, ByVal e As System.Windows.Forms.MouseEventArgs) Handles TChart1.ClickSeries   
        UpDatePoint(valueIndex, TChart1.Axes.Left.CalcPosPoint(e.Y))   
End Sub

调用UpdatePoint方法来修改柱形的值：

[C#]   
private void UpDatePoint(int Bar, double Y)   
        {   
            if(Bar < tChart1.Series[0].Count)   
            {   
                tChart1.Series[0].YValues[Bar] = Y;   
                tChart1.Series[0].Repaint();     
            }                                                                
        }   
   
[VB.Net]   
Private Sub UpDatePoint(ByVal Bar As Integer, ByVal Y As Double)   
        If Bar < TChart1.Series(0).Count Then   
            TChart1.Series(0).YValues(Bar) = Y   
            TChart1.Series(0).Repaint()   
        End If   
End Sub

**数据序列事件**

上一节介绍了一些关于数据序列事件的使用。本节说明一些其他用途。  
  
**OnClickSeries**   
您可以使用OnClickSeries事件来获取几乎任何一个数据序列的信息（请参阅，访问数据序列值）。  
  
这些例子适用于带有日期型数据的数据序列，例如将这些测试值用在下面的事件例子中：

[C#]   
private void button1\_Click(object sender, System.EventArgs e)   
        {   
            Random rnd = new Random();   
            line1.XValues.DateTime = true;   
            line1.Pointer.Visible = true;   
            line1.Add(DateTime.Parse("25/12/2002 10:30:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("25/12/2002 22:30:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("26/12/2002 09:20:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("26/12/2002 23:30:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("27/12/2002 11:10:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("27/12/2002 20:15:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("28/12/2002 08:15:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("28/12/2002 21:45:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("29/12/2002 12:45:00"),rnd.Next(100),"", Color.Red);   
            line1.Add(DateTime.Parse("29/12/2002 22:05:00"),rnd.Next(100),"", Color.Red);   
   
            line1.HorizAxis = Steema.TeeChart.Styles.HorizontalAxis.Top;   
        }   
   
private void tChart1\_ClickSeries(object sender, Steema.TeeChart.Styles.Series s, int valueIndex, System.Windows.Forms.MouseEventArgs e)   
        {   
            //The below will show the Value of the nearest Point, not the exact Axis value at the clicked X and Y.    
            MessageBox.Show("Date is: " + DateTime.FromOADate(line1.XValues[valueIndex])   
            + " Value is: " + line1.YValues[valueIndex]);    
        }   
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Dim rnd As New Random()   
        Line1.XValues.DateTime = True   
        Line1.Pointer.Visible = True   
        Line1.Add(DateTime.Parse("25/12/2002 10:30:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("25/12/2002 22:30:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("26/12/2002 09:20:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("26/12/2002 23:30:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("27/12/2002 11:10:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("27/12/2002 20:15:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("28/12/2002 08:15:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("28/12/2002 21:45:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("29/12/2002 12:45:00"), rnd.Next(100), "", Color.Red)   
        Line1.Add(DateTime.Parse("29/12/2002 22:05:00"), rnd.Next(100), "", Color.Red)   
   
        Line1.HorizAxis = Steema.TeeChart.Styles.HorizontalAxis.Top   
End Sub   
   
Private Sub TChart1\_ClickSeries(ByVal sender As Object, ByVal s As Steema.TeeChart.Styles.Series, ByVal valueIndex As Integer, ByVal e As System.Windows.Forms.MouseEventArgs) Handles TChart1.ClickSeries   
        'The below will show the Value of the nearest Point, not the exact Axis value at the clicked X and Y.    
        MsgBox("Date is: " & DateTime.FromOADate(Line1.XValues(valueIndex)) \_   
              & " Value is: " & Line1.YValues(valueIndex))   
End Sub

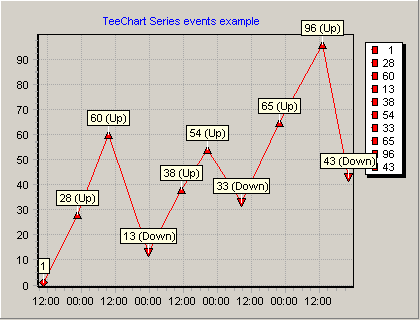
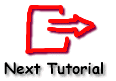
**OnGetSeriesPointerStyle**   
对于适用TChart指针的数据序列，您可以通过OnGetSeriesPointer事件来访问和修改指针：

如果该点高于最后一个，那么就绘制一个Uptriangle，如果较低就绘制一个DownTriangle等：

[C#]   
private void line1\_GetPointerStyle(Steema.TeeChart.Styles.CustomPoint series, Steema.TeeChart.Styles.GetPointerStyleEventArgs e)   
        {   
            if(e.ValueIndex > 0)   
            {   
                if(line1.YValues[e.ValueIndex] > line1.YValues[e.ValueIndex - 1])   
                {   
                    e.Style = Steema.TeeChart.Styles.PointerStyles.Triangle;   
                }   
                else if(line1.YValues[e.ValueIndex] < line1.YValues[e.ValueIndex - 1])   
                {   
                    e.Style = Steema.TeeChart.Styles.PointerStyles.DownTriangle;   
                }   
                else   
                {   
                    e.Style = Steema.TeeChart.Styles.PointerStyles.Diamond;   
                }   
            }   
            else   
            {   
                e.Style = Steema.TeeChart.Styles.PointerStyles.Diamond;   
            }   
        }   
   
[VB.Net]   
Private Sub Line1\_GetPointerStyle(ByVal series As Steema.TeeChart.Styles.CustomPoint, ByVal e As Steema.TeeChart.Styles.GetPointerStyleEventArgs) Handles Line1.GetPointerStyle   
        If e.ValueIndex > 0 Then   
            If (Line1.YValues(e.ValueIndex) > Line1.YValues(e.ValueIndex - 1)) Then   
                e.Style = Steema.TeeChart.PointerStyles.Triangle   
            ElseIf (Line1.YValues(e.ValueIndex) < Line1.YValues(e.ValueIndex - 1)) Then   
                e.Style = Steema.TeeChart.Styles.PointerStyles.DownTriangle   
            Else   
                e.Style = Steema.TeeChart.Styles.PointerStyles.Diamond   
            End If   
        Else   
            e.Style = Steema.TeeChart.Styles.PointerStyles.Diamond   
        End If   
End Sub

**OnGetSeriesMark**   
在运行时，使用OnGetSeriesMark事件修改标记内容。 下面的代码演示了根据最后的值来改变了MarkText的过程；  
TeeChart支持通过DragMarks工具来拖拽重叠的标记。

[C#]   
private void line1\_GetSeriesMark(Steema.TeeChart.Styles.Series series, Steema.TeeChart.Styles.GetSeriesMarkEventArgs e)   
        {   
            if(e.ValueIndex > 0)   
            {   
                if(line1.YValues[e.ValueIndex] > line1.YValues[e.ValueIndex - 1])   
                {   
                    e.MarkText = e.MarkText + " (Up)";   
                }   
                else if(line1.YValues[e.ValueIndex] < line1.YValues[e.ValueIndex - 1])   
                {   
                    e.MarkText = e.MarkText + " (Down)";   
                }   
                else   
                {   
                    e.MarkText = e.MarkText + " (No Change)";   
                }   
            }   
        }   
   
[VB.Net]   
Private Sub Line1\_GetSeriesMark(ByVal series As Steema.TeeChart.Styles.Series, ByVal e As Steema.TeeChart.Styles.GetSeriesMarkEventArgs) Handles Line1.GetSeriesMark   
        If (e.ValueIndex > 0) Then   
            If (Line1.YValues(e.ValueIndex) > Line1.YValues(e.ValueIndex - 1)) Then   
                e.MarkText = e.MarkText + " (Up)"   
            ElseIf (Line1.YValues(e.ValueIndex) < Line1.YValues(e.ValueIndex - 1)) Then   
                e.MarkText = e.MarkText + " (Down)"   
            End If   
        Else   
            e.MarkText = e.MarkText + " (No Change)"   
        End If   
End Sub

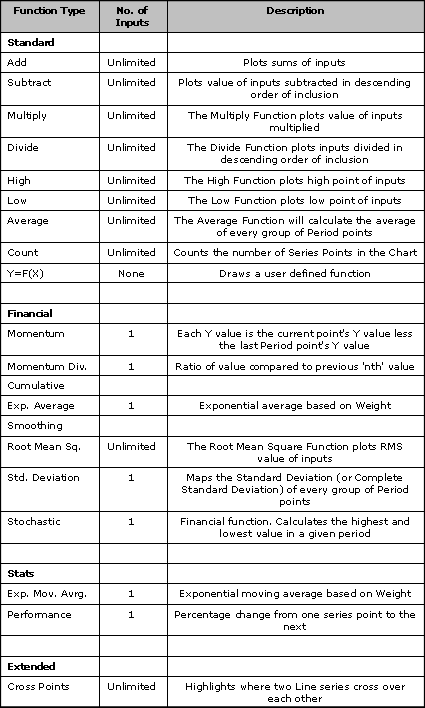
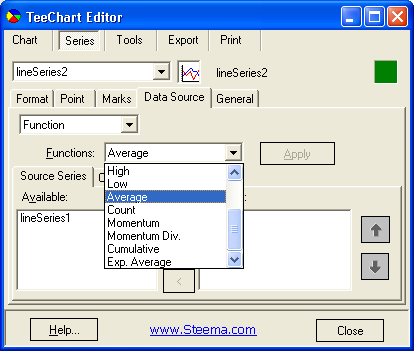
由最后两个事件所产生的图表效果如下：  
  
  
  
  
[](#Bookmark6)(See 1.1.5)        [](#Bookmark8)(See 1.1.7)

### 1.1.7 教程7 － 使用函数

**教程7 － 使用函数  
目录**

[**函数类型**](#Bookmark8_FunctionTypes)   
     [函数特性](#Bookmark8_Features)   
     [添加一个函数](#Bookmark8_AddFunction)   
     [定义一个数据源](#Bookmark8_Datasource)   
     [函数周期](#Bookmark8_FunctionPeriod)  
     [周期样式](#Bookmark8_PeriodStyle)  
     [继承传统函数](#Bookmark8_CustomFunction)

**函数类型**

**函数特性**TeeChart Pro的函数就是一个数据序列（可以是任何的数据序列类型），对此代数函数也适用，另外数据源也是另一个图表数据序列。  
  
在Steema.TeeChart.Functions命名空间中所有的函数都来源于Function类并且继承函数的 [Period属性](#Bookmark8_FunctionPeriod)。 TeeChart Pro提供了下列预定义函数的清单：  
  
  
  
有几个函数类型只支持一种输入数据序列。不过可以控制函数的联系，举一个例子来说明，使用几个数据序列的平均值来创建一个Average Function Series，然后通过像Trend Function输入一样来使用Average Function，从而标识出平均趋势。  
  
**添加一个函数**   
在TeeChart编辑器的首页上，就像添加一个新数据序列到图表一样，选择Add按钮。在TeeChart Gallery中选择Functions标签来选择您想要的Function。每个Function都是以线型数据序列显示的，稍后您可以通过首页上的Change按钮来修改Function所关联的数据序列类型。函数的定义很容易被Datasource页上的函数数据序列所修改。在这里，也很简单，您可以修改已经添加到图表的数据序列的函数的定义（函数实际上是数据源的定义，而不是数据序列类型的定义）。   
  
下面的图片展示了编辑函数时Datasource 页的内容。Line Series（“line2”标题）已经被定义了。Datasource页底部的左列表框展示了在图表中其它可见的输入（这里指“line1”）。  
  
  
  
假设我们以一个全新的空图表开始，代码中的步骤建立了一个简单的与图表相关的Series-Function。

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
   
   
            //Add a data Series   
            Line line1 = new Line(tChart1.Chart);   
   
            //Populate it with data (here random)   
            line1.FillSampleValues(10);   
   
            //Add a series to be used for an Average Function   
            Line line2 = new Line(tChart1.Chart);   
   
            //Define the Function Type for the new Series   
            Steema.TeeChart.Functions.Average average1 = new Steema.TeeChart.Functions.Average();   
            line2.Function = average1;   
   
            //Define the Datasource for the new Function Series   
            line2.DataSource = line1;   
   
            //\*Note - When populating your input Series manually you will need to    
            //use the Checkdatasource method    
            //- See the section entitled 'Defining a Datasource'   
            //Change the Period of the Function so that it groups averages   
            //every 2 Points   
            line2.Function.Period = 2;   
            line2.CheckDataSource();   
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        'Add a data Series   
        Dim Line1 As New Steema.TeeChart.Styles.Line(TChart1.Chart)   
           
        'Populate it with data (here random)   
        Line1.FillSampleValues(10)   
   
        'Add a series to be used for an Average Function   
        Dim Line2 As New Steema.TeeChart.LineSeries(TChart1.Chart)   
   
        'Define the Function Type for the new Series   
        Dim Average1 As New Steema.TeeChart.Functions.Average()   
        Line2.Function = Average1   
   
        'Define the Datasource for the new Function Series   
        Line2.DataSource = Line1   
   
        '\*Note - When populating your input Series manually you will need to    
        'use the Checkdatasource method    
        '- See the section entitled 'Defining a Datasource'   
        'Change the Period of the Function so that it groups averages   
        'every 2 Points   
        Line2.Function.Period = 2   
        Line2.CheckDataSource()   
End Sub

我们可以添加其他函数来告诉我们关于前一个函数的一些东西。

[C#]   
private void button1\_Click(object sender, System.EventArgs e)   
           
            //Let's change to 2D for visibility   
            tChart1.Aspect.View3D = false;   
            //Add another Series to be used for a 2nd Function    
            Line line3 = new Line(tChart1.Chart);   
            //Define the Function Type for the new Series    
            Steema.TeeChart.Functions.High high1 = new Steema.TeeChart.Functions.High();   
            line3.Function = high1;   
            //Define the Datasource for the new Function Series    
            //Use the existing Function (Series2) as input    
            line3.DataSource = tChart1.Series[1];   
            //Leave the Period at default 0 (No Period set) to draw    
            //A line at Highest of all points of the Average Function    
           
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        'Let's change to 2D for visibility   
        TChart1.Aspect.View3D = False   
        'Add another Series to be used for a 2nd Function    
        Dim Line3 As New Steema.TeeChart.LineSeries(TChart1.Chart)   
        'Define the Function Type for the new Series    
        Dim High1 As New Steema.TeeChart.Functions.High()   
        Line3.Function = High1   
        'Define the Datasource for the new Function Series    
        'Use the existing Function (Series2) as input    
        Line3.DataSource = TChart1.Series(1)   
        'Leave the Period at default 0 (No Period set) to draw    
        'A line at Highest of all points of the Average Function    
End Sub

**定义一个数据源**   
在上一节的例子以代码的方式强调通过数据源来填充函数。数据序列使用数据源来定义函数的输入，或者使用ADO.NET数据源（查看关于访问数据库的教程）。  
  
使用TeeChart编辑器添加一个函数后，Function Series' Datasource页会显示包含在函数定义中的所有可用数据序列的清单。在这里您可以修改您想要运用到函数的数据序列，还可以从左边“Available”列表框中选择数据序列并添加到右边“Selected”列表框中。

对Series.Datasource属性编程来定义数据源。  
  
**例子：**   
假设在设计时，我们通过TeeChart编辑器向图表添加了两个数据序列。我们向这两个数据序列添加Average Function：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
           
            tChart1.Aspect.View3D = false;   
            bar1.FillSampleValues(10);   
            bar2.FillSampleValues(10);   
           
   
        private void button1\_Click(object sender, System.EventArgs e)   
           
            Steema.TeeChart.Styles.Line line1 = new Steema.TeeChart.Styles.Line(tChart1.Chart);   
            Steema.TeeChart.Functions.Average average1 = new Steema.TeeChart.Functions.Average();    
            line1.DataSource = new object[]  this.bar2,this.bar1;   
            line1.Function = average1;   
            line1.Marks.Visible = true;   
           
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        TChart1.Aspect.View3D = False   
        Bar1.FillSampleValues(10)   
        Bar2.FillSampleValues(10)   
End Sub   
   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        Dim DataSource As New ArrayList()   
   
        DataSource.Add(Bar1)   
        DataSource.Add(Bar2)   
   
        Dim Line1 As New Steema.TeeChart.Styles.Line(TChart1.Chart)   
        Dim Average1 As New Steema.TeeChart.Functions.Average()   
   
        Line1.Function = Average1   
        Line1.DataSource = DataSource   
End Sub

我们向2个数据序列添加点：

[C#]   
private void button2\_Click(object sender, System.EventArgs e)   
           
            Random rnd = new Random();   
            for(int i = 0; i < 10; ++i)   
               
                bar1.Add(rnd.Next(500));   
                bar2.Add(rnd.Next(500));   
               
           
   
[VB.Net]   
Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click   
        Dim rnd As New Random()   
        Dim i As Integer   
        For i = 0 To 10   
            Bar1.Add(rnd.Next(500))   
            Bar2.Add(rnd.Next(500))   
        Next   
End Sub

请注意，函数是不显示的。您需要在Button2\_Click()事件中添加Series.CheckDatasource方法为函数读取值。

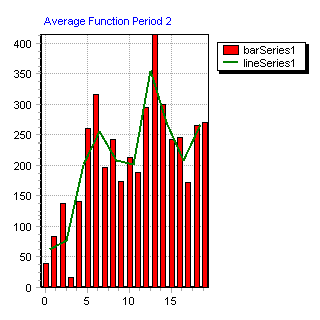
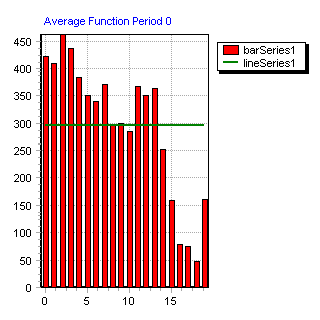
[C#]   
tChart1.Series[2].CheckDataSource();   
[VB.Net]   
TChart1.Series(2).CheckDataSource()

在运行时，通过重新定义Series.DataSource属性，函数定义可能会被简单地更改为向数据序列分配一个新的函数：

[C#]   
private void button3\_Click(object sender, System.EventArgs e)   
           
            Steema.TeeChart.Functions.Cumulative cumulative1 = new Steema.TeeChart.Functions.Cumulative();   
            tChart1.Series[2].Function = cumulative1;   
           
   
[VB.Net]   
Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click   
        Dim Cumulative1 As New Steema.TeeChart.Functions.Cumulative()   
        TChart1.Series(2).Function = Cumulative1   
End Sub

**函数周期**   
周期是函数工作中最重要的属性，因为周期定义了一个函数所应用点的范围。  
例子：   
  
我们有6个带值的数据点（如柱形数据序列的柱形）：  
  
     3, 8, 6, 2, 9 and 12    
  
我们定义一个周期为0（默认）的函数数据序列，平均值是：  
  
     6.667   
  
随着周期设置为2，我们得到3个函数输出的平均值：  
  
     5.5, 4 and 10.5   
  
这些值将集中在他们的周期范围内，即第一个值介于Bar1和Bar2的输入数据序列，第二个值介于Bar3和Ba4之间等。   
  
您可以在Datasource页通过选择有关数据序列和函数来定义周期并点击“Options”标签，或者您也可以在运行时通过FunctionType来修改周期。  
  
例如：line1是一个函数数据序列：

[C#]   
line1.Function.Period = 2;   
[VB.Net]   
Line1.Function.Period = 2

下面的2个图表显示了一个应用周期的影响。  
  
  
  
**周期样式**   
周期可以定义为一个范围。在使用DateTime数据序列和想要把“Period”功能表示为一个TimeStep时是非常有用的。“PeriodStyle”属性控制“Period”如何表示。   
  
例如在一个日期型源数据序列中，你可以使用一个标准“Average”函数来划分“monthly average of sales”功能，并设置函数周期为“one month”：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
            //Add in a BarSeries and Average Function at design-time.   
            Random rnd = new Random();   
            tChart1.Aspect.View3D = false;     
              
            TimeSpan month = new TimeSpan(30,0,0,0);   
            DateTime today = DateTime.Today;   
   
            bar1.Marks.Visible = false;   
            bar1.XValues.DateTime = true;   
            tChart1.Axes.Bottom.Labels.Angle = 90;   
   
            for(int i = 0; i < 60; ++i)   
                today = today.AddDays(5);   
                bar1.Add(today, rnd.Next(100),"",Color.Red);   
               
   
            average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First;   
            average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range;   
            average1.Period = month.TotalDays;   
            line1.DataSource = bar1;   
            line1.CheckDataSource();   
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        'Add in a BarSeries and Average Function at design-time.   
        TChart1.Aspect.View3D = False   
   
        Dim Month As New TimeSpan(30, 0, 0, 0)   
        Dim Today As DateTime = DateTime.Today   
        Dim i As Integer   
   
        Bar1.Marks.Visible = False   
        Bar1.XValues.DateTime = True   
        TChart1.Axes.Bottom.Labels.Angle = 90   
   
        For i = 0 To 60   
            Today = Today.AddDays(5)   
            Bar1.Add(Today, Rnd() \* 100, "", Color.Red)   
        Next   
   
        Average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First   
        Average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range   
        Average1.Period = Month.TotalDays   
        Line1.DataSource = Bar1   
        Line1.CheckDataSource()   
End Sub

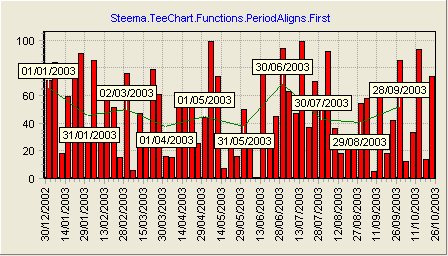
这样的结果是每个点显示的是BarSeries中每个月“average”的数据。  
  
在日期型周期上计算的时候，源数据序列的点必须存储为日期，这是强制性的。   
  
该范围也可以用在非日期型数据序列：

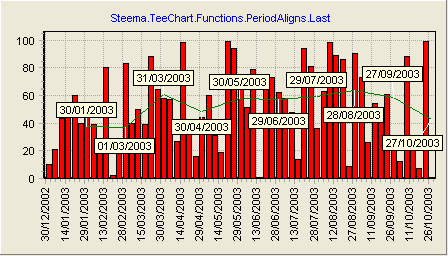
[C#]   
for(int i = 0; i < 60; ++i)   
     bar1.Add(Convert.ToDouble(i), rnd.Next(100),"",Color.Red);   
   
average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First;   
average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range;   
average1.Period = 6;   
   
[VB.Net]   
For i = 0 To 60   
            Bar1.Add(i, Rnd() \* 100, "", Color.Red)   
Next   
Average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First   
Average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range   
Average1.Period = 6

这将以“6”为分隔点，为每个组别的点计算平均值。  
（X的点X>=6,X<6时用来计算第一个平均值，X的点X>=6,X<12时用来计算第二个平均值...）   
注意，这不同于每6点计算一次平均值。

使用周期的Alignment属性对齐数据序列范围内的功能点。下面的操作会在满一个月周期的时候绘制功能点：

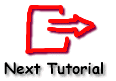
[C#]   
average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First;   
average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range;   
average1.Period = month.TotalDays;   
   
[VB.Net]   
Average1.PeriodAlign = Steema.TeeChart.Functions.PeriodAligns.First   
Average1.PeriodStyle = Steema.TeeChart.Functions.PeriodStyles.Range   
Average1.Period = Month.TotalDays

Period = Month.TotalDays和PeriodAligns.First   
如下图所示，“average”在每月结束的时候绘制。  
  
  
  
Period = Month.TotalDays和PeriodAligns.Last   
这种情况下，“average”在每月开始的时候绘制。

  
  
**继承传统函数**  
创建一个新的函数组件就是简单地从函数类继承一个新的组件（同时也可以从已经存在的函数类继承）。在函数中，有两个非常重要的虚拟方法可以被重写用于创建一个新的函数类型。  
  
1) Function.Calculate: public virtual double Calculate(Series Source,int First,int Last)   
  
2) Function.CalculateMany:  public virtual double CalculateMany(ArrayList SourceSeries, int ValueIndex)   
  
Calculate方法是用来计算函数的结果，前提是只有一个数据序列来作为数据源。Calculate方法是用来计算函数的结果，前提是有多个数据序列来充当数据源。  
  
例子：创建新的SquareSum函数。   
  
我们决定SquareSum函数需要返回“sum of squares”。  
  
这个函数可以拥有一个或多个数据源，因此我们重写Calculate和CalculateMany两个方法。

fnnnt [C#]   
public class SquareSum : Steema.TeeChart.Functions.Function   
        public SquareSum(): base()   
        public SquareSum(Steema.TeeChart.Chart c): base(c)   
  
        public override double Calculate(Series SourceSeries,int FirstIndex,int LastIndex)     
            ValueList v=ValueList(SourceSeries);   
            if ( FirstIndex==-1 ) return v.Total;   
            else   
                double result=0;   
                for (int t=FirstIndex; t<=LastIndex; t++)     
                    result+=Math.Sqrt(v[t]);   
                return result;   
               
           
  
        public override double CalculateMany(ArrayList SourceSeriesList,int ValueIndex)   
            ValueList v;   
            double result=0;   
  
            for (int t=0; t<SourceSeriesList.Count; t++)   
                v=ValueList((Series)SourceSeriesList[t]);   
                if ( v.Count>ValueIndex )     
                    result+=Math.Sqrt(v[ValueIndex]);   
               
            return result;   
           
  
  
[VB.Net]   
Public Class SquareSum   
        Inherits Steema.TeeChart.Functions.Function   
  
        Public Sub New()   
            MyBase.New()   
        End Sub   
  
        Public Sub New(ByVal c As Steema.TeeChart.Chart)   
            MyBase.New(c)   
        End Sub   
  
        Public Overrides Function Calculate(ByVal Source As Steema.TeeChart.Series, ByVal First As Integer, ByVal Last As Integer) As Double   
            Dim v As Steema.TeeChart.ValueList   
            Dim t As Integer   
            v = ValueList(Source)   
  
            If First = -1 Then   
                Return v.Total   
            Else   
                Dim Result As Double = 0   
                For t = First To t < Last   
                    Result += Math.Sqrt(v(t))   
                Next   
                Return Result   
            End If   
        End Function   
  
        Public Overrides Function CalculateMany(ByVal SourceSeries As System.Collections.ArrayList, ByVal ValueIndex As Integer) As Double   
            Dim v As Steema.TeeChart.ValueList   
            Dim Result As Double = 0   
            Dim t As Integer   
  
            For t = 0 To t < SourceSeries.Count   
                v = ValueList(CType(SourceSeries(t), Steema.TeeChart.Series))   
                If v.Count > ValueIndex Then   
                    Result += Math.Sqrt(v(ValueIndex))   
                End If   
            Next   
  
            Return Result   
        End Function   
End Class

FirstIndex和EndIndex变量是用来循环整个SourceSeries的点，因而我们就可以计算squares的和。

“ValueList”方法用来提取Steema.TeeChart.ValueList让类同数据序列类型一起工作，比如在“XValues”持有点值而不是“YValues”的HorizBarSeries。  
  
当只有唯一的一个数据序列充当数据源时“Calculate”方法才能被使用。当有多个数据序列充当数据源时“CalculateMany”方法才被调用。  
  
在源数据序列中的每个点都可以调用“CalculateMany”方法一次，从0开始直到所有数据源点总和的最小值。  
  
理解Calculate和CalculateMany的不同是非常重要的。当只有唯一的数据源时才调用“Calculate”方法并且只能调用一次。当存在多个数据序列充当数据源时就可以多次（每个点只有一次）调用“CalculateMany” 方法。  
  
[](#Bookmark7)(See 1.1.6)     [](#Bookmark9)(See 1.1.8)

### 1.1.8 教程8 － ADO.NET数据库访问

**教程8 － ADO.NET数据库访问   
目录**

[**说明**](#Bookmark9_ADOIntro)  
[**设计时连接到ADO.NET数据源**](#Bookmark9_ADODesign)   
     [DataAdapter](#Bookmark9_DataAdapterD)   
     [单条记录](#Bookmark9_SingleD)   
  
[**运行时连接到ADO.NET数据源**](#Bookmark9_ADORuntime)   
     [DataAdapter](#Bookmark9_DataAdapterR)   
     [单条记录](#Bookmark9_SingleR)   
  
[**ASP.NET使用ADO.NET**](#Bookmark9_ADOASP)

**说明**

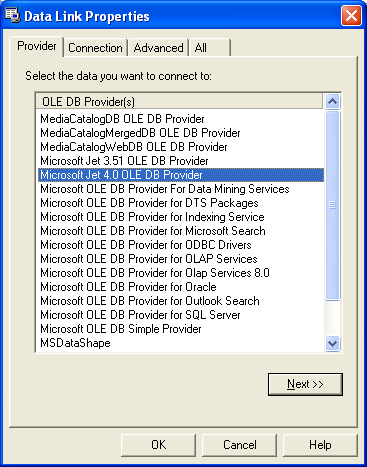
在设计时，您可以使用TeeChart编辑器把TeeChart控件连接到ADO.NET数据库，您也可以在运行时通过一些代码来实现这个功能。  
  
任何数据序列都能连接到ADO.NET表单或通过TeeChart编辑器来进行查询。每个数据序列都可以连接到相同或不同的表单，或者通过相同或不同的ADO.NET数据库进行查询。  
  
这种设计可以让您非常灵活的把图表连接到数据库，因为您不局限于只有一个数据库，或只有一个表单或查询。  
  
在设计时，数据也可以被检索出来，所以您在开发过程中也能看到实际数据。

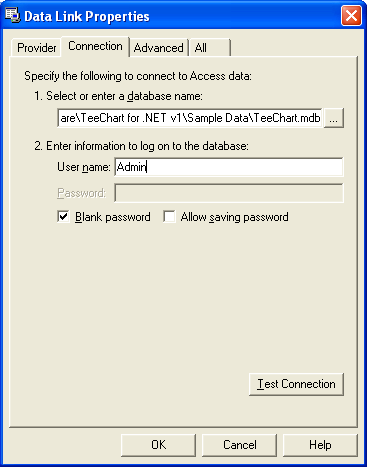
**设计时连接到ADO.NET**

**DataAdapter**  
在设计时，数据序列连接到ADO.NET数据源的关键步骤如下：

 1. 从VS.NET工具栏的Data标签，拖拽一个OleDbDataAdapter到一个拥有TeeChart for .Net项目的WinForm上。

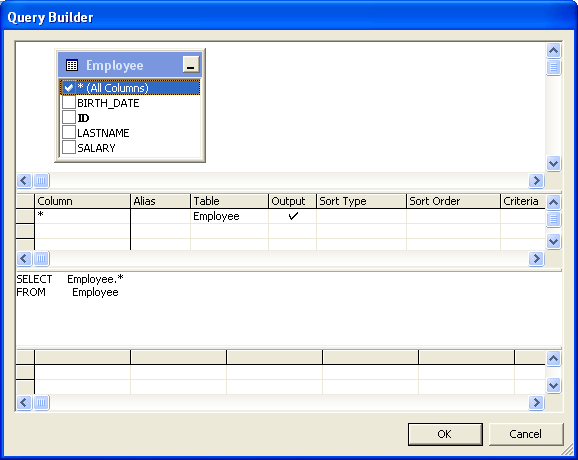
 2. 在Data Adapter Configuration Wizard上点击“Next”并在下一页点击“New Connection...”按钮。

 3. 在新窗口上点击“Provider”标签并选择“Microsoft Jet OLE DB Provider”，如下图所示：   
  


 4. 在当前的窗口上点击“Connection”标签并从TeeChart for .NET根目录下的示例数据文件中选择TeeChart.mdb文件：   
  


 5. 点击“Test Connection”按钮来测试这个连接并在当前视窗中点击“OK”按钮。

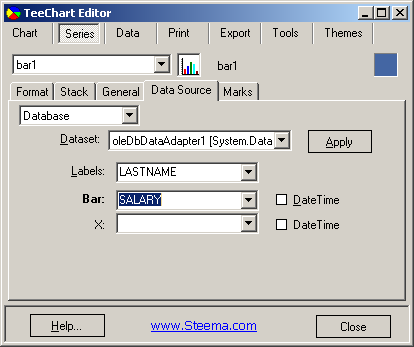
 6. 在Data Adapter Configuration Wizard视窗上点击“Next”按钮，勾选出现在视窗上的“Use SQL Statements”并再次点击“Next”按钮。

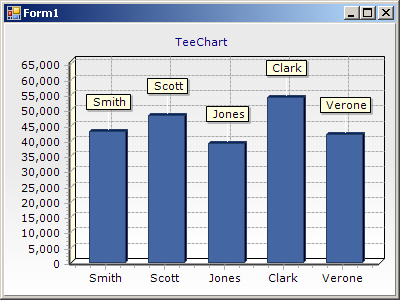
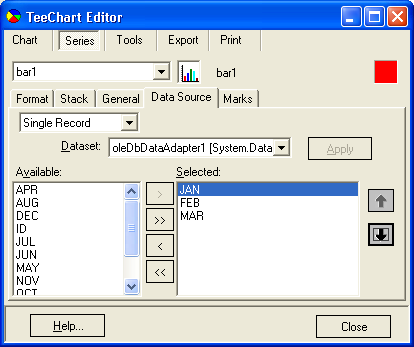
 7. 点击“Query Builder”按钮并添加“Employee”表单。关闭“Add Table”视窗并选中“\*(All Columns)”复选框：   
  


 8. 在当前窗口上点击“OK”按钮，并点击“Next”，然后点击“Finish”按钮。 ??  
??

 9. 现在打开TeeChart编辑器并添加一个BarSeries，然后转到Series标签的DataSource页。

 10. 从组合框中选择“Database”，并从DataSet 组合框中选择oleDbDataAdapter1 [System.Data.OleDb.OleDbDataAdapter]。 ??  
??

 11. 设置标签为“LASTNAME”并将Y坐标轴设置为“SALARY”，如下图所示：   
    


 12. 现在点击“Apply”按钮并运行窗体，您将看到如下图所示的结果：   
  
  
  
**单条记录**Single Record Charting允许显示关联在TeeChart上单一数据库的全部或部分记录。它可能对数据库视图或表是非常有用的，例如，每年的销售数字是作为一个单独月的列来存储的，如'JAN'，'FEB'，'MAR'等，并通过记录显示连续多年的每月销售额。  
  
Single Record Charting支持Datasets, Tables和DataViews。使用图表编辑器选择单条记录来作为一个图表数据序列的数据源类型。  
  
  
  
完美连接到一个单条记录数据中的大部分步骤，都是需要通过图表编辑器来实现的。想要找到数据源，并相应地更新到您的图表需要通过Windows.Form的CurrencyManager编写一些代码。请参阅关于[编码的步骤](#Bookmark9_SingleR) 章节中的使用示范。

**运行时连接到ADO.NET**

**[.NET 2003版本]**   
**DataAdapter**上述在设计时中的步骤可以通过一些编码来实现。这使您能更加自由和完全的控制数据库操作。  
  
每个数据序列都拥有“DataSource”属性，属性决定了数据库值（表或查询）的来源，并且“YValues.DataMember”和“LabelMember”属性指定我们想要绘制的字段。  
  
将一个新的TeeChart for .NET项目拖拽一个新的项目和一个新的窗体上。

[C#]   
using System.Data;   
using System.Data.OleDb;   
using System.Security;   
using System.Security.Permissions;   
   
private void Form1\_Load(object sender, System.EventArgs e) {   
            DataSet masterDataSet = new DataSet();   
            Bar bar1 = new Bar(tChart1.Chart);   
            try   
            {   
                OpenConnWithJet40(ref masterDataSet);   
                DataTable employeeTable = masterDataSet.Tables["EMPLOYEE"];   
                bar1.YValues.DataMember = employeeTable.Columns["SALARY"].ToString();   
                bar1.LabelMember = employeeTable.Columns["LASTNAME"].ToString();   
                bar1.DataSource = employeeTable;   
            }   
            catch (SecurityException)   
            {   
                MessageBox.Show("The calling method has not been granted sufficient permission to access the Microsoft Jet 4.0 OLE DB Provider.");   
            }   
}   
          
   
 [OleDbPermission(SecurityAction.Demand, Provider = "Microsoft.Jet.OLEDB.4.0")]   
    
private void OpenConnWithJet40(ref DataSet masterDataSet) {   
               
            OleDbConnection testConn = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" + @"Data Source=C:\Program Files\Steema Software\TeeChart for .NET v3\Sample data\TeeChart.mdb;");             
            testConn.Open();   
            string strCom = "SELECT \* FROM Employee";   
            OleDbDataAdapter myCommand = new OleDbDataAdapter(strCom,testConn);   
            myCommand.Fill(masterDataSet, "Employee");   
            testConn.Close();   
   
        }   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
            Dim MasterDataSet As New DataSet()   
            Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
            Try   
                OpenConnWithJet40(MasterDataSet)   
                Dim employeeTable As DataTable = MasterDataSet.Tables("Employee")   
                Bar1.YValues.DataMember = employeeTable.Columns("SALARY").ToString()   
                Bar1.LabelMember = employeeTable.Columns("LASTNAME").ToString()   
                Bar1.DataSource = employeeTable   
            Catch ex As Exception   
                MessageBox.Show("Microsoft Jet 4.0 OLE DB Provider problem: " & ex.Message)   
            End Try   
End Sub   
   
Private Sub OpenConnWithJet40(ByRef MasterDataSet As DataSet)   
            Dim TestConn As New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=C:\Program Files\Steema Software\TeeChart for .NET v3\Sample data\TeeChart.mdb;")   
            TestConn.Open()   
   
            Dim StrCom As String = "SELECT \* FROM Employee"   
            Dim MyCommand As New OleDbDataAdapter(StrCom, TestConn)   
            MyCommand.Fill(MasterDataSet, "Employee")   
            TestConn.Close()   
        End Sub   
End Class

在.NET 2005版本中，稍微有一些针对2005版的变化。

**[.NET 2005版本]   
DataAdapter**

[C#]   
using System.Data;   
using System.Data.OleDb;   
using System.Security;   
using System.Security.Permissions;   
private void Form1\_Load(object sender, EventArgs e)   
 {   
            //string ConnString = SqlDataSource1.ConnectionString;         
            string sQuery = "select \* from Employee";   
             
            //SqlConnection myConnection = new SqlConnection(ConnString);               
            string myConnString = "Driver={Microsoft Access Driver (\*.mdb)};Dbq=C:\\Program Files\\Steema Software\\TeeChart for .NET v3\\Sample data\\TeeChart.mdb;Uid=Admin;Pwd=;";   
            System.Data.Odbc.OdbcConnection myConnection = new System.Data.Odbc.OdbcConnection(myConnString);   
             
            //SqlCommand myCommand = new SqlCommand(sQuery, myConnection);   
            System.Data.Odbc.OdbcCommand myCommand = new System.Data.Odbc.OdbcCommand(sQuery, myConnection);   
            myConnection.Open();   
             
            //SqlDataReader myReader = myCommand.ExecuteReader();   
            System.Data.Odbc.OdbcDataReader myReader = myCommand.ExecuteReader();   
   
            int XColumn = myReader.GetOrdinal("ID");   
            int YColumn = myReader.GetOrdinal("SALARY");   
            int LabelColumn = myReader.GetOrdinal("LASTNAME");   
   
            Steema.TeeChart.Styles.Bar bar1 = new Steema.TeeChart.Styles.Bar(tChart1.Chart);   
   
            while (myReader.Read())   
            {   
                bar1.Add(Convert.ToInt32(myReader[XColumn]), Convert.ToDouble(myReader[YColumn]), Convert.ToString(myReader[LabelColumn]));   
            }   
   
            myReader.Close();   
            myConnection.Close();   
}

[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

        Dim MasterDataSet As New DataSet()   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Try   
            OpenConnWithJet40(MasterDataSet)   
            Dim employeeTable As DataTable = MasterDataSet.Tables("EMPLOYEE")   
            Bar1.YValues.DataMember = employeeTable.Columns("SALARY").ToString()   
            Bar1.LabelMember = employeeTable.Columns("LASTNAME").ToString()   
            Bar1.DataSource = employeeTable   
        Catch ex As Exception   
            MessageBox.Show("Microsoft Jet 4.0 OLE DB Provider problem: " & ex.Message)   
        End Try   
 End Sub   
   
 Private Sub OpenConnWithJet40(ByRef MasterDataSet As DataSet)   
          
        Dim TestConn As New OleDb.OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;" & "Data Source=C:\Program Files\Steema Software\TeeChart for .NET v3\Sample data\TeeChart.mdb;")   
        TestConn.Open()   
        Dim StrCom As String = "SELECT \* FROM Employee"   
        Dim MyCommand As New OleDb.OleDbDataAdapter(StrCom, TestConn)   
        MyCommand.Fill(MasterDataSet, "Employee")   
        TestConn.Close()   
    End Sub   
   
End Class

如果数据库的值发生改变，你要“refresh”连接并重新检索值，你应该这样做：

TChart1.Series(0).CheckDataSource

“CheckDataSource”方法会迫使数据序列重新检索所有的记录。  
所有的数据序列同样拥有“XValues.ValueSource”属性，这个属性只用在XY图表中（当你在每个点都拥有X坐标）。  
  
**单条记录**  
**连接到表**  
使用图表编辑器将在窗体的初始化中创建下面的代码。类似的代码可用于在运行时创建并连接新的单条记录数据源。  
  
（假设一个DataSet调用sourceTableSet的表单：sourceTable）

this.singleRecordSource1 = new Steema.TeeChart.Data.SingleRecordSource();   
this.sourceTableSet.DataSetName = "NewDataSet";   
this.sourceTableSet.Tables.AddRange(new System.Data.DataTable[] {this.sourceTable});   
this.singleRecordSource1.DataSource = this.sourceTableSet;   
this.singleRecordSource1.ValueMembers = new string[] {   
                                                             "JAN",   
                                                             "FEB",   
                                                             "MAR"};

this.bar1 = new Steema.TeeChart.Styles.Bar();   
this.bar1.DataSource = this.singleRecordSource1;

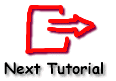
**表单检索**  
TeeChart的SingleRecord使用CurrencyManager组件来检索数据库的表单并更新到图表。在表单中的记录变更后应该使用CheckDatasource来更新图表。  
  
**例子**

private CurrencyManager myCurrencyManager;   
   
public void ConnectChartToTable()   
{   
     myCurrencyManager = (CurrencyManager)this.BindingContext[sourceTable];   
     singleRecordSource1.RecordCurrency=myCurrencyManager;   
     tChart1[0].CheckDataSource();   
}

表单（和图表）可以通过CurrencyManager来进行检索。

    private void MovePrevious()   
    {   
      if (myCurrencyManager.Position>0)   
      {   
        myCurrencyManager.Position=myCurrencyManager.Position+1;   
        tChart1[0].CheckDataSource();   
      }   
    }   
   
    private void MoveNext()   
    {   
      if (myCurrencyManager.Position<myCurrencyManager.Count)   
      {   
        myCurrencyManager.Position=myCurrencyManager.Position+1;   
        tChart1[0].CheckDataSource();   
      }   
    }

**ASP.NET使用ADO.NET**

您可以像WinForm中TeeChart组件连接数据源的方式一样，在ASP.NET WebForm中使用TeeChart WebChart，通过ASP.NET来连接到数据源。 请参阅 [互联网应用程序](#Bookmark10)(See 1.1.9) 教程获取更多关于ASP.NET的信息。  
  
[](#Bookmark8)(See 1.1.7)     [](#Bookmark10)(See 1.1.9)

### 1.1.9 教程9 － ASP.NET应用程序

**教程9 － TeeChart、WebChart、ASP.NET**

[**WebForms**](#Bookmark10_WebForms)     [向窗体添加一个图表](#Bookmark10_addChart)   
     [使用图表编辑器和属性选项进行设计](#Bookmark10_EditChart)  
     [图像格式（PictureFormat属性）](#Bookmark10_ImgFormat)   
     [TempChart 属性 － File，Session，Cache，Httphandler](#Bookmark10_TmpChart)   
     [使用交互式图表事件（AutoPostBack属性）](#Bookmark10_AutoPost)   
     [事件举例](#Bookmark10_EventSamples)   
     [安全问题](#Bookmark10_Security)  
  
[**ASP应用程序中的脚本编程**](#Bookmark10_FreeScripts)     [使用脚本添加图表](#Bookmark10_ChartToScript)  
     [处理流程](#Bookmark10_Procflow)   
     [对TeeChart编码](#Bookmark10_Coding)   
     [表单检索的关键代码元素](#Bookmark10_GettingTheChart)   
     [添加事件](#Bookmark10_AddEvent)   
  
[**WebChart工具**](#Bookmark10_WebChartTools)   
     [HotspotTool](#Bookmark10_hotspottool)   
     [ScrollTool](#Bookmark10_scrolltool)   
     [ZoomTool](#Bookmark10_zoomtool)   
     [Window onload 事件](#Bookmark10_winOnload)

**WebForms**

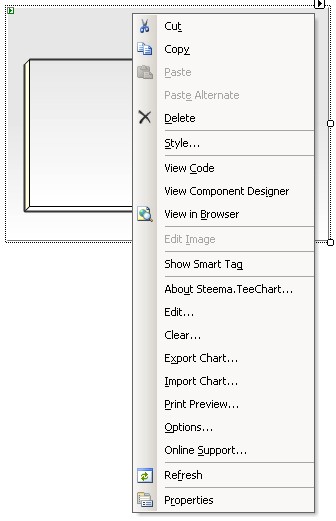
TeeChart是通过**WebChart** TeeChart控件与WebForms集成的。WebChart可以在标准TeeChart.NET安装中的工具箱中找到。  
  
C:\Users\SONY\AppData\Local\Temp\407984\Tutorial source\Tutorial source\images\WebChartIcon.jpg

TooBox中WebChart的图标

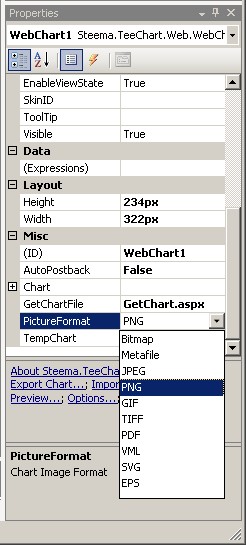
**向窗体添加一个图表**

 在ToolBox中选择WebChart组件并拖拽到WebForm上，然后调整它的大小

 右击图表选择“Edit...”选项从而打开图表编辑器

 使用图表编辑器您可以添加和修改数据序列、坐标轴以及图表的其它特性。  
  
**使用图表编辑器和属性选项来进行设计**  
  
图表编辑器提供了多种选项来配置图表在运行时的很多问题。设置会被保存到WebForm。请参阅关于图表编辑器使用的教程1和教程2，WebChart编辑器的使用与基于图表的Windows窗体的使用是完全相同的。  
  
  
  
**图像格式（PictureFormat）**  
WebChart就像一张图片显示在WebForm上，默认的格式是PNG，但您可以在Properties Windows中通过PictureFormat属性来进行修改。由于网页的一致性，我们推荐使用PNG，JPEG或者GIF格式，尽管Internet Explorer也支持Bitmap格式。

使用GIF图像格式

相对于全部彩色复制，GIF使用了256色的缩小，所以我们建议您使用其它图像格式的一种。  
**\*注意** 您使用GIF图像格式可能会遇到授权的问题。  
参阅 [Unisys and LZW](http://www.unisys.com/about__unisys/lzw/) 获取更多关于GIF LZW编码的图像使用信息。  
  
  
  
  
**TempChart属性 － File，Session，Cache，Httphandler**   
  
TempChart属性可以用来选择临时图表创建后及在网页上显示前的保存方式。这里有4个可用的设置：

 File

 Session

 Cache

 Httphandler   
性能与所有的缓存技术是类似的，但磁盘性能可能会积极地或消极地影响使用'File'选项来产生TempChart，所以‘File’是最快的选择。  
  
**File**   
如果你想要把临时文件写到磁盘上，那么就可以使用File设置。默认情况下，TeeChart安装时就创建了关于临时文件位置的注册表键值和虚拟文件夹的名称。注册表键值如下所示：

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Steema Software\TeeChart.NET]   
"VirtualShare"="/TeeChartForNET"   
"ShareFolder"="C:\\Program Files\\Steema Software\\TeeChart for .NET v3\\TeeChartForNET"

如果接受默认的安装位置，以上设置是在标准的英语语言机器上所创建的。TeeChartNET虚拟文件夹是被TeeChart安装文件建立的。您可以更改这些注册表设置的前提是你创建必要的IIS虚拟共享位置，以反映新的注册表值。  
  
WebChart把文件保存到ShareFolder注册表键值下一个名为\_Chart\_tmp的文件夹中。  
临时文件不会被TeeChart删除，虽然一个应用程序可能会像Windows预设任务一样自动的为你这样做，请检查当前版本TeeChar所含内容。  
  
**Session**   
Session是一个在磁盘上不留下任何文件的临时文件存储的设置，并删除自己已缓存的内存文件（也可参阅见Cache和HttpHandler选项）。在某些情况下，你可能会发现它略慢于临时磁盘文件。图表能够成功运行Session变量，有两个条件必须为真：

 Session必须被应用程序激活。‘Enabled’是默认的方式，您可以在项目中的web.config文件中进行确认：

    <sessionState    
            mode="InProc"

InProc是默认的设置并且意味着Session已经被应用程序激活。

 您需要包含一个aspx script模块来从Memory中检索图表。aspx script的名称可以称为‘GetChart’。一个包含TeeChart发布的WebForm ASP.NET的例子，要从头创建一个新的GetChart模块需要在您的项目中添加一个新的WebForm（或者是任何aspx文件）命名为'GetChart'并将下面的代码添加到Page\_Load事件中：  
  
GetChart.aspx.cs

        private void Page\_Load(object sender, System.EventArgs e)   
        {   
            string chartName=Request.QueryString["Chart"];   
   
            if (Session[chartName]!=null)   
            {   
                System.IO.MemoryStream chartStream = new System.IO.MemoryStream();   
                chartStream=((System.IO.MemoryStream)Session[chartName]);   
                Response.ContentType = "image/" + "png";   
                Response.OutputStream.Write(chartStream.ToArray(),0,(int)chartStream.Length);   
                chartStream.Close();   
                Session.Remove(chartName);   
            }   
        }

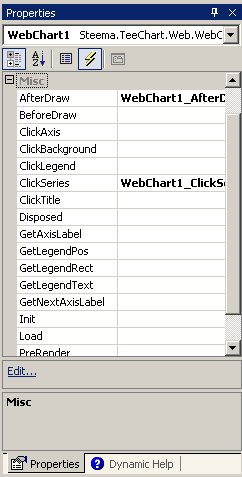
该脚本恢复了事先已经被WebChart保存到Session缓存的图表，该脚本不仅向浏览器返回了图表而且从缓存中移除了图表。  
  
**Cache**   
Cache选项与安装到Session Chart缓存选项十分相似。'Cache'是一个实用的选项，因为它使用的缓存对象属于ASP.NET WebForm页面。鉴于'Session'会受到某些Web Servers的限制，因此使用Cache很容易对大多数ASP.NET应用程序进行配置（如在管理员选择出于安全原因禁用会话支持）。  
  
Cache选项要求使用GetChart.aspx文件。

 您需要包含一个aspx script模块来从Memory中检索图表。aspx script的名称可以称为‘GetChart’。一个包含TeeChart发布WebForm ASP.NET的例子的例子。要从头创建一个新的GetChart模块需要在您的项目中添加一个新的WebForm（或者是任何aspx文件）命名为'GetChart'并将下面的代码添加到Page\_Load事件中：  
  
  
GetChart.aspx.cs

        private void Page\_Load(object sender, System.EventArgs e)   
        {   
            string chartName=Request.QueryString["Chart"];   
   
            if (Page.Cache[chartName]!=null)   
            {   
                MemoryStream chartStream = new MemoryStream();   
                chartStream=((MemoryStream)Page.Cache[chartName]);   
                Response.ContentType = "image/" + "png";   
                Response.OutputStream.Write(chartStream.ToArray(),0,(int)chartStream.Length);   
                chartStream.Close();   
            }   
        }

该脚本恢复了事先已经被WebChart保存到Page Cache的图表，该脚本不仅向浏览器返回了图表而且从缓存中移除了图表。  
  
**Httphandler**   
HttpHandler选项使用内部的TeeChart（WebChart）生成器恢复该页的图表图像。这个选项不需要产生任何临时文件以及附加的aspx文件来恢复图表（在Session和Cache的情况下）。该选项只需要对web.config文件做一个修改：  
  
在web.config的<system.web> </system.web>标签中的任何位置假如以下部分：

<system.web>   
  <httpHandlers>   
       <add verb="\*" path="TeeChartImgGen.ashx" type="Steema.TeeChart.Web.TeeChartImgGen, TeeChart"/>   
  </httpHandlers>   
   
/\*..content cut for brevity ..\*/   
   
</system.web>

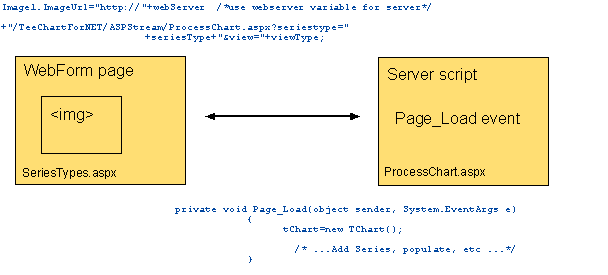
**使用交互式图表事件（当Chart没有WebChart Tools时，AutoPostBack属性为True）**   
图表中的Load事件和在运行时响应用户鼠标点击的交互事件可能用于TeeChart的WebChart。没有特别的动作需要使用绘制事件如'AfterDraw'，我们可以从属性浏览器的事件列表来选择一个图表事件。如果没有TeeChart WebChart Tool如Zoom，Scroll或者Hotspot显示在图表中，您需要设置AutoPostback属性为True来激活交互（用户点击）事件。对于交互事件，如果上述Tools是当前点击自动激活的反应，图表将响应用户的鼠标点击事件，回发在哪里发生的点击的信息到服务器，于是你的服务器端代码可以采取针对该事件的行动。你可以从属性浏览器的事件列表选择想要使用的事件，然后对该事件进行编码。  
参阅‘Event example’教程获取更多关于WebChart事件的信息。  
  
**事件举例**   
如果您想要在WebChart中使用一个Click事件如ClickSeries，请确认图表的AutoPostback设置为True。  
事件可以通过双击在属性浏览器事件列表中需要的事件来进行设置：  
  
  
  
双击列表中的一个条目创建代码事件的声明。  
  
**例子**

        private int xVal;   
        private int yVal;   
   
        private void WebChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g)   
        {   
            g.TextOut(xVal,yVal,"onAfterDraw");   
        }

上述事件将输出文字到图表上的一个由xVal和yVal变量值决定的位置。  
  
根据点击的位置来设置x和y的值。

        private void WebChart1\_ClickSeries(object sender, Steema.TeeChart.Series s, int valueIndex, System.EventArgs e)   
        {   
            xVal=s.CalcXPos(valueIndex);   
            yVal=s.CalcYPos(valueIndex);       
        }

**ASP应用程序中的脚本编程**

一个对于WebChart的替代使用是在asp（aspx）页面中以scripted ‘non-visible’控件的方式使用TeeChart。 TeeChart for .NET发布了一个形象的例子来演示此技术。你可以在TeeChart's ASP.NET演示项目中找到它，在‘Interacting with Charts\Chart as Image’演示文件下有个例子的标题是‘Series Types as Image’。  
  
通过内部的图表类，TeeChart也许可以在ASP中称为free-scripted，为了利用TeeChart事件，我们建议使用TeeChart Windows Forms组件或者WebChart控件。  
  
**使用脚本添加图表**   
假设使用TeeChart，下面是您建立TeeChart项目的初始化步骤：  
1. 创建新的ASP.NET Forms项目。这样就建立了一个WebForm页面，您将会当作客户端页面来使用（参阅例子SeriesTypes.aspx页面）。   
2. 添加新的WebForm页面。此页面将是不可见的，将用于处理图表代码的服务器端。（参阅例子ShowSeries.aspx 页面） 。   
- 该页面必须包含System.Windows.Forms.dll的引用。因为TeeChart组件是一个基于Windows.Forms的组件。  
3. Follow the code steps highlighted here and in the ASPStreams example as a guide to how to setup your project.   
  
**处理流程**   
客户端浏览器页面（SeriesTypes.aspx）包含一个图像，其IMG连接是aspx服务器脚本（ShowSeries.aspx）。客户端页面上的提交按钮调用含有ServerChart处理的参数化变量的服务器脚本。该按钮发送变量来作为设置客户端页面上图表图像的imageURL的参数的一部分，因此，返回一个基于已处理的参数的动态图表。  
  
  
  
**对TeeChart 编码**为了在服务器端对TeeChart进行编码应在脚本页面上添加一个TeeChart的引入：  
  
例如：

using Steema.TeeChart;

然后为TeeChart声明一个变量并在Page\_Load事件的开始的时候建立。  
  
例如：   
private TChart tChart;

        private void Page\_Load(object sender, System.EventArgs e)   
        {   
            tChart=new TChart();         
   
            /\* ...add Series and data, etc ... \*/   
        }

**表检索的关键码元素**   
表检索的关键码元素是：  
1. 客户端‘Get’说明。这可以由一个提交按钮来初始化或者其它手段，例如一个ComboBox并基于客户端参数从服务端‘gets’图表。一个图表的请求可能像‘traditional Get url line where the parameters are passed as one URL line or may be done via ASP.NET’的“Passing Server Control Values Between Pages”这样（请参阅微软对这个问题的帮助主题获取更多详情）。调用服务器脚本的代码将在页面后的的WebForms Code中编写。  
  
**例子**   
在此例中，变量是来自不同WebForm的页面元素并且以参数的方式添加到图像’Get‘URL路径。

        private void sendInfo()   
        {   
            seriesType=DropDownList1.Items[DropDownList1.SelectedIndex].ToString();   
            viewType=CheckBox1.Checked.ToString();   
            Image1.ImageUrl="http://"+webServer  /\*use webserver variable for server\*/   
                +"/TeeChartForNET/ASPStream/ProcessChart.aspx?seriestype="   
                +seriesType+"&view="+viewType;   
        }

2. 服务端脚本接收来自客户端的GET请求并运行Page\_Load事件。在事件中，图表可能被创建且接收到的参数可能被用来定义图表如何填充。图表会呈现为一个图像并向客户端返回一个数据流。  
  
**例子**

        private void Page\_Load(object sender, System.EventArgs e)   
        {   
            tChart=new TChart();   
            tChart.AfterDraw += new Steema.TeeChart.TChart.PaintChartEventHandler(this.tChart\_AfterDraw);         
   
            /\* ...add Series and data, etc ... \*/   
   
            //Process Get parameters received from client   
            if (Request.QueryString["view"]=="False")   
                tChart.Aspect.View3D=false;   
            else   
                tChart.Aspect.View3D=true;   
   
            MemoryStream tempStream = new MemoryStream();   
            tChart.Export.Image.PNG.Save(tempStream);   
            Response.ContentType="Image/PNG";   
            Response.OutputStream.Write(tempStream.ToArray(),0,(int)tempStream.Length);   
            tempStream.Close();   
        }

**添加事件**  
通过在图表建立后添加事件，图表的建立事件可能会被添加到Free-scripted TeeChart ASP应用程序。  
**例子**

        private void Page\_Load(object sender, System.EventArgs e)   
        {   
            tChart=new TChart();   
            tChart.AfterDraw += new Steema.TeeChart.TChart.PaintChartEventHandler(this.tChart\_AfterDraw);         
   
            /\* ...add Series and data, etc ... \*/   
   
            MemoryStream tempStream = new MemoryStream();   
            tChart.Export.Image.PNG.Save(tempStream);   
            Response.ContentType="Image/PNG";   
            Response.OutputStream.Write(tempStream.ToArray(),0,(int)tempStream.Length);   
            tempStream.Close();   
        }   
   
        private void tChart\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g)   
        {   
            string tmpStr="Copyright My Organisation "+DateTime.Now.ToString();   
            g.TextOut(g.Chart.Width-(int)g.TextWidth(tmpStr)-5,g.Chart.Height-(int)g.TextHeight("H")-3,tmpStr);   
        }

在图表建立的时候，AfterDraw事件中的代码就会被执行，在这种情况下，当图表呈现为图像时，相关信息将包含在Chart Canvas上。如果您想要在客户端页面的图表上使用基于用户鼠标点击的交互式事件，我们建议您在WebForm上使用WebChart组件。  
  
**安全问题**   
通过Installer默认安装的TeeChart for .NET创建了文件夹和足以访问TeeChart所包含的ASP.NET例子的IIS虚拟文件夹。TWebForm应用程序采取IIS支持的Session来移动临时图表的信息。例子中的项目可以在Visual Studio.NET中使用临时存储图表的媒介‘File’被修改。经测试，在Windows 2000中运行良好。在Windows 2003中默认安装了附加的安全限制，这可能限制了默认文件夹‘\_chart\_temp’（临时文件夹的名称和位置是可配置的）的使用。Win2003服务器的安全权利必须经过系统管理员修改才允许有关ASP.NET应用程序中的临时文件保存到磁盘。

**WebChart工具**

WebChart工具可以从Editor Tools调色板中添加一个WebChart。许多non-WebChart专用工具可用于WebCharts，除了那些响应鼠标移动（除WebChart工具本身所涵盖的）。所有工具的技术都在TeeChart WebChart的示例中进行了描述。  
  
**HotspotTool**   
该工具用最简单的形式激活一个鼠标悬停在数据点的标签，显示为鼠标通过了一点。HotspotTool也可用于WinForm Charts，但操作性质不同于此处所描述的WebChart。WebChart HotspotTool生成与每个数据点相关联的 区域。一个Map Action 可以从下面的选项中来选择：  
  
- Mark － 根据选择的Style选项显示一个Point Mark。  
- URL － 当一个数据点被点击时连接到URL。  
- Script － 当一个数据点被点击时运行客户端脚本。  
  
**Mark**   
使用Style属性来定义Mark Style。  
  
例子（通过代码实现）

Steema.TeeChart.Tools.SeriesHotspot hotspotTool =    
         ((Steema.TeeChart.Tools.SeriesHotspot)WebChart1.Chart.Tools[0]);

hotspotTool.MapAction = Steema.TeeChart.Styles.MapAction.Mark;   
hotspotTool.Style = Steema.TeeChart.Styles.MarksStyles.LabelPercentTotal;   
  
**URL**   
使用GetHTMLMap事件来设置的一个被数据点调用的URL。您可以通过编辑器来设置URL选项，但是您必须对事件进行编码来设置当前URL。  
  
例子

//init code....   
Steema.TeeChart.Tools.SeriesHotspot hotspotTool =    
         ((Steema.TeeChart.Tools.SeriesHotspot)WebChart1.Chart.Tools[0]);   
   
hotspotTool.GetHTMLMap += new Steema.TeeChart.Tools.SeriesHotspotEventHandler(hotspotTool\_GetHTMLMap);   
   
//event code:   
private void hotspotTool\_GetHTMLMap(Steema.TeeChart.Tools.SeriesHotspot sender, Steema.TeeChart.Tools.SeriesHotspotEventArgs e)   
{   
     if (CheckBox1.Checked)  //open new window?   
        e.PointPolygon.Attributes="target='\_blank'";   
     else   
        e.PointPolygon.Attributes="target='\_self'";        tab   
   
     if (e.Series==WebChart1.Chart.Series[0]) e.PointPolygon.HREF="http://" + TextBox1.Text; //set URL according to textbox text + TextBox1.Text; //set URL according to textbox text   
     if (e.Series==WebChart1.Chart.Series[1]) e.PointPolygon.HREF="http://" + TextBox2.Text;   
     if (e.Series==WebChart1.Chart.Series[2]) e.PointPolygon.HREF="http://" + TextBox3.Text; + TextBox3.Text;   
     if (e.Series==WebChart1.Chart.Series[3]) e.PointPolygon.HREF="http://" + TextBox4.Text;   
}

**Script**   
Script选项可以用来处理客户端内容或者您通过JavaScript想要添加到图表的value-add。当Script被当作一个选项选择时，TeeChart添加此选项来使用一个辅助脚本，现有的：‘Annotation’（参考Steema.TeeChart.Tools.HotspotHelperScripts）。您也可以不选择Helperscript并自己定义一个输出。  
  
Annotation示例：

protected void Page\_Load(object sender, System.EventArgs e)   
{   
   
  //initialization   
   
  Chart ch1 = WebChart1.Chart;   
   
  Steema.TeeChart.Themes.ColorPalettes.ApplyPalette(ch1, 9);   
  Steema.TeeChart.Tools.SeriesHotspot hotspot1 = new Steema.TeeChart.Tools.SeriesHotspot();   
  ch1.Legend.Visible = false;   
  ch1.Tools.Add(hotspot1);   
  hotspot1.MapAction = Steema.TeeChart.Styles.MapAction.Script;   
   
  hotspot1.GetHTMLMap += new Steema.TeeChart.Tools.SeriesHotspotEventHandler(hotspot1\_GetHTMLMap);   
     
  //....etc... more init code   
}   
   
private void hotspot1\_GetHTMLMap(Steema.TeeChart.Tools.SeriesHotspot sender, Steema.TeeChart.Tools.SeriesHotspotEventArgs e)   
{   
     //This example calls a Bar Series but e.Series and e.PointPolygon.ValueIndex could be sent   
     //as arguments for a drilldown query.   
   
     //The prepared HelperScriptAnnotation text accepts your text as a variable for the Annotation   
     //In the following case it calls an aspx script to generate and return a Chart as an image   
     e.PointPolygon.Attributes=String.Format(Texts.HelperScriptAnnotation,   
                            "<IMG SRC=ShowSeries.aspx?seriestype=Bar&view=False&width=100&height=80>");   
   
     //The annotation could, alternatively, present text in the mouseover hint, eg.:   
     //e.PointPolygon.Attributes=String.Format(Steema.TeeChart.Texts.HelperScriptAnnotation,"hello world.");   
}

在上面例子中使用的HelperScriptAnnotation如下：   
HelperScriptAnnotation ="onmouseover=\"ShowAnnotation('{0}');\" onmouseout=\"ShowAnnotation('');\"";   
如果你要添加您自己的输出，您需要调用自己的代码。  
  
例子

string myProcess = ="onmouseover=\"ShowAssociatedDataTable('{0}');\" onmouseout=\"ShowAssociatedDataTable('');\"";   
e.PointPolygon.Attributes=String.Format(myProcess,e.PointPolygon.ValueIndex.ToString());

于是你可以自定义ShowAssociatedDataTable方法以显示相关的数据表。  
  
**ScrollTool**   
ScrollTool向WebChart的底部添加一个滚动条。你可以设置图表可见部分的大小和起始位置。图表也可以拖动，虽然该功能可能会不尽人意。ScrollTool可于HotspotTool结合起来组合。  
  
使用编辑器添加一个ScrollTool将自动设置图表为2D并且移动较低的坐标轴来空出滚动条的空间。在设计时，滚动条是不可见的。  
  
ViewSegmentSize和StartPosition与SegmentViewUnits结合起来使用可以设置滚动的大小。  
  
例子

Steema.TeeChart.Tools.ScrollTool scrollTool =    
                        ((Steema.TeeChart.Tools.ScrollTool)WebChart1.Chart.Tools[0]);   
scrollTool.StartPosition = 30;   
scrollTool.SegmentViewUnits = Steema.TeeChart.Tools.ScrollToolViewUnit.percent;   
scrollTool.ViewSegmentSize = 20;

如果您想要使用Windows onload事件请参阅 [Window onload事件](#Bookmark10_winOnload) 章节。  
  
**ZoomTool**   
ZoomTool允许选择一个图表子区域进行缩放处理。该区域可以通过在图表上拖动鼠标来进行选择，Zoom co-ordinates被返回到服务器用来创建一个可缩放区域。如果反方向拖动那么该区域是不可以缩放的。  
  
为了能够使用ZoomTool，您需要在页面代码中添加下面的方法。  
Using Session:

private void CheckZoom(WebChart wChart)   
{   
     ArrayList zoomedState=(ArrayList)Session[wChart.ID+"Zoomed"];   
     zoomedState=((Steema.TeeChart.Tools.ZoomTool)wChart.Chart.Tools[0]).SetCurrentZoom(Request,   
        zoomedState);   
     if (zoomedState==null)   
        Session.Remove(wChart.ID+"Zoomed");   
     else   
        Session.Add(wChart.ID+"Zoomed",zoomedState);   
}

或者Page.Cache:

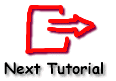
private void CheckZoom(WebChart wChart)   
{   
  ArrayList zoomedState = (ArrayList)Page.Cache[wChart.ID + "Zoomed"];   
  zoomedState = ((Steema.TeeChart.Tools.ZoomTool)wChart.Chart.Tools[0]).SetCurrentZoom(Request,   
    zoomedState);   
  if (zoomedState == null)   
    Page.Cache.Remove(wChart.ID + "Zoomed");   
  else   
    Page.Cache.Add(wChart.ID + "Zoomed", zoomedState);   
}

Page\_Load方法结束后调用此方法，传递您想要缩放的图表的名称。  
例子

CheckZoom(WebChart1);

图表上任何没有x或y位移的点击/拖动都当成一个点击发送回服务器，而不是一个缩放。如果您想要使用Windows onload事件请参阅 [Window onload event](#Bookmark10_winOnload)章节。  
  
  
**Window onload事件**  
当一个Zoom或者Scroll Tool添加到您的WebChart时，您需要使用Window.onload事件来初始化Tool的特性。如果您想要进一步根据自己的需求来使用onload事件，那么请在页面上添加一个新功能来调用windowOnload事件并在事件中使用您自己的代码。  
  
例如：在页面的<HEAD>标签中添加windowOnload事件

<script language=javascript>   
function windowOnload()    
{   
   window.status = new Date().getTime();   
}   
</script>

[](#Bookmark9)(See 1.1.8)     [](#Bookmark11)(See 1.1.10)

### 1.1.10 教程10 - ASP.NET 实例的初步学习

**Tutorial10 - ASP.NET 实例的初步学习   
目录**

[**Web Form 实例**](#Bookmark11)   
  
参考TeeChart的TeeChartForNET安装共享，里面有WebForm工作实例及C#.NET和VB.NET编写的ASP.NET实例

**Web Form 实例**

**怎样创建一个动态的WebChart**

 在你的服务器上创建一个新的WebForm应用程序，要确定该空白表单是否能正常运行。

 从Steema工具箱选项卡中，选择一个WebChart对象，将它拖放到WebForm中。

 选择新建的WebChart1对象，在属性窗口中找到TempChart属性，将其由File变为Session。这就意味着WebChart生成的所有的临时图表将被保存在一个session变量中而不是临时文件夹中（详见教程9 ）

 为了恢复session变量中的临时图表，我们将在新的窗体中添加一些简单的代码。右键单击您的ASP.NET项目，添加一个新的WebForm，并将其命名为GetChart.aspx。此时请确保Page\_Load事件如下所示：

private void Page\_Load(object sender, System.EventArgs e)   
{   
     string chartName=Request.QueryString["Chart"];                 
     if (Session[chartName]!=null)                
    {                    
        System.IO.MemoryStream chartStream = new System.IO.MemoryStream();                
        chartStream=((System.IO.MemoryStream)Session[chartName];                    
        Response.OutputStream.Write(chartStream.ToArray(),0,(int)chartStream.Length);                    
        chartStream.Close();                    
        Session.Remove(chartName);                
    }    
}

 现在我们要接着编写一些基本的聚焦功能；在我们原有WebForm的Form\_Load 事件中，我们可添加以下类似的代码：

private void Page\_Load(object sender, System.EventArgs e)   
{   
     //Let's work with the Chart object for convenience   
     Steema.TeeChart.Chart Chart1 = WebChart1.Chart;   
   
     //Add in a series and fill it   
     Chart1.Aspect.View3D = false;   
     Steema.TeeChart.Styles.Bubble bubble1 = new Steema.TeeChart.Styles.Bubble(Chart1);   
     bubble1.FillSampleValues();   
   
     //Add a SeriesToolTip to the Chart   
     Steema.TeeChart.Tools.SeriesHotspot seriesHotSpot1 = new Steema.TeeChart.Tools.SeriesHotspot(Chart1);   
     //Steema.TeeChart.Styles.MapAction.Mark is the default value   
     seriesHotSpot1.MapAction = Steema.TeeChart.Styles.MapAction.Mark;   
}

运行代码，在气泡图上移动鼠标您就可以看到一系列标记的值，另外还有YValues 。

 在图表中添加缩放功能，我们所要做的就是添加一个缩放工具和一些简单的代码来控制缩放状态：  
private void Page\_Load(object sender, System.EventArgs e)   
{   
     //Let's work with the Chart object for convenience   
     Steema.TeeChart.Chart Chart1 = WebChart1.Chart;   
  
     //Add in a series and fill it   
     Chart1.Aspect.View3D = false;   
     Steema.TeeChart.Styles.Bubble bubble1 = new Steema.TeeChart.Styles.Bubble(Chart1);   
     bubble1.FillSampleValues();   
  
     //Add a SeriesToolTip to the Chart   
     Steema.TeeChart.Tools.SeriesHotspot seriesHotSpot1 = new Steema.TeeChart.Tools.SeriesHotspot(Chart1);   
     //Steema.TeeChart.Styles.MapAction.Mark is the default value   
     seriesHotSpot1.MapAction = Steema.TeeChart.Styles.MapAction.Mark;   
  
     //Add a ZoomTool to the Chart   
     Steema.TeeChart.Tools.ZoomTool zoomTool1 = new Steema.TeeChart.Tools.ZoomTool(Chart1);   
  
     // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Code for zoom support \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
     //check whether zoom request is being sent   
     CheckZoom(WebChart1);   
}   
  
private void CheckZoom(WebChart wChart)   
{   
     ArrayList zoomedState=(ArrayList)Session[wChart.ID+"Zoomed"];   
     zoomedState=((Steema.TeeChart.Tools.ZoomTool)wChart.Chart.Tools[0]).SetCurrentZoom(Request,zoomedState);   
     if (zoomedState==null)   
        Session.Remove(wChart.ID+"Zoomed");   
     else   
        Session.Add(wChart.ID+"Zoomed",zoomedState);   
}   
We now have an interactive chart which responds to mouseover and mouseclick events. The SeriesHotSpot, in this case associated with a bubble series, will display the value of the Series Marks when the mouse is moved over it. However, via the MapAction property, we can customize the behavior of the SeriesHotSpot when the mouse is moved over it. For example, we may want a click on one of the bubbles to take us to a specified URL; this is perfectly possible by setting the MapAction property to URL, linking up the SeriesHotSpot event and specifying the URL in it:

 现在我们有了一个可响应鼠标悬停和鼠标点击事件的交互式图表。与气泡图相关的SeriesHotSpot，在鼠标悬停时可以显示所有标记的值。然而，通过设定MapAction属性，我们还可以自定义SeriesHotSpot在鼠标悬停时的行为。例如，我们希望点击气泡图上的一个点时，可以跳转到一个特定的网址；在图表中响应SeriesHotSpot事件并指向它的网址，通过设定MapAction属性是完全可以实现的，如下所示：   
  
在Page\_Load 事件中:

seriesHotSpot1.MapAction = Steema.TeeChart.Styles.MapAction.URL;   
seriesHotSpot1.GetHTMLMap += new Steema.TeeChart.Tools.SeriesHotspotEventHandler(seriesHotSpot1\_GetHTMLMap);

使用GetHTMLMap方法:

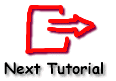
private void seriesHotSpot1\_GetHTMLMap(Steema.TeeChart.Tools.SeriesHotspot sender, Steema.TeeChart.Tools.SeriesHotspotEventArgs e)    
{   
     e.PointPolygon.Title = "Go to the Steema web";   
     e.PointPolygon.HREF = "http://www.steema.com";   
     e.PointPolygon.Attributes = "target='\_blank'";   
}

 将MapAction属性设置为Script能让您轻松地定义所需的任何行为，TeeChart为您提供了一些有用的内置scripts，并可以通过HelperScripts枚举来进行调用。例如，您想要当鼠标悬停在一个与气泡图相关联的点上时，打开一张图片，我们可以添加如下代码：   
  
在Page\_Load 事件中:

seriesHotSpot1.MapAction = Steema.TeeChart.Styles.MapAction.Script;   
seriesHotSpot1.HelperScript = Steema.TeeChart.Tools.HotspotHelperScripts.Annotation;

这里的第二行是为了确保将相关的JavaScript添加到客户浏览器中。  
  
使用GetHTMLMap方法:

private void seriesHotSpot1\_GetHTMLMap(Steema.TeeChart.Tools.SeriesHotspot sender, Steema.TeeChart.Tools.SeriesHotspotEventArgs e)    
{   
     e.PointPolygon.Attributes=String.Format(Steema.TeeChart.Texts.HelperScriptAnnotation, "<IMG SRC=Images/myimage.jpg>");   
}

To customize the behavior even further would simply mean designing your own JavaScript routines, adding them to the client browser and then calling them by adding them and their parameters to e.PointPolygon.Attributes. 更多的自定义行为事件意味着您能轻松地设计自己的JavaScript程序，并将他们添加到客户浏览器，然后通过添加它们及e.PointPolygon.Attributes中的参数来进行调用  
  
  
[](#Bookmark10)(See 1.1.9)     [](#Bookmark12)(See 1.1.11)

### 1.1.11 教程11- 缩放和滚动

**教程 11 - 缩放和滚动   
 要在数据密集的图表中选中特定的数据，缩放和滚动是非常有用的功能。**

**目录**

[**如何使用鼠标进行图表的缩放与滚动**](#Bookmark12_Manual)   
     [缩放](#Bookmark12_ManZoom)   
     [滚动](#Bookmark12_ManScroll)   
  
[**如何通过代码来进行图表的缩放与滚动**](#Bookmark12_Code)   
     [缩放](#Bookmark12_CodeZoom)   
     [动态缩放](#Bookmark12_CodeAnimZoom)   
     [缩放事件](#Bookmark12_ZoomEvents)   
     [滚动](#Bookmark12_CodeScroll)

**如何使用鼠标进行图表的缩放与滚动**

**缩放**   
如果要放大一个图表，可在你想放大的区域左上角一直按着鼠标左键，在缩放区域的右下角拖动矩形框。松开鼠标时，图表中您所选定的区域将被刷新。   
如果要撤销缩放，可在图表区域的任何区域一直按着鼠标左键向左上角拖动。松开鼠标后图表就会恢复到原始的图表。  
  
**滚动**要滚动一张图表，可以一直按着鼠标右键，向您想要滚动的方向拖动鼠标。当您松开鼠标后图表就会停留在一个新的位置。  
如果要取消滚动，可在图表的任何区域一直按着鼠标左键向左上角拖动。松开鼠标后图表就会恢复到原始的图表。

**如何通过代码来进行图表的缩放与滚动**

**缩放**   
缩放功能是默认可用的。通过Zoom.Allow属性可禁用缩放功能。参考Zoom类与Zoom相关的所有属性和方法。运用ZoomRect方法来定义要缩放的一个矩形区域。

例如   
[C#]   
tChart1.Zoom.ZoomRect(new Rectangle(100,100,120,120));   
[VB.Net]   
TChart1.Zoom.ZoomRect(New Rectangle(100, 100, 120, 120))

ZoomRect中的co-ordinates是在屏幕像素中定义的，在这里0,0是图表左上角的点。   
下面的代码可使图表在x轴上的第2和第5点的区域内放大，设定整个图表y轴上最大和最小的缩放比例：

[C#]   
int x = points1.CalcXPos(2);   
int y = tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MaxYValue);   
int height = tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MinYValue) - tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MaxYValue);   
int width = points1.CalcXPos(5) - x;   
Rectangle r = new Rectangle(x,y,width,height);   
tChart1.Zoom.ZoomRect(r);   
   
[VB.Net]   
Dim X As Integer = Points1.CalcXPos(2)   
Dim Y As Integer = TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MaxYValue)   
Dim Height As Integer = TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MinYValue) - TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MaxYValue)   
Dim Width As Integer = Points1.CalcXPos(5) - X   
Dim R As New Rectangle(X, Y, Width, Height)   
TChart1.Zoom.ZoomRect(R)

使用 'Undo' 撤销缩放

TChart1.Zoom.Undo

**动态缩放**   
动态缩放提供了更进一步的缩放功能。取代了由“缩小”到“放大”的一步式操作，您可以将Animated属性设为可用，并为缩放操作定义交错的步骤。一旦Animated属性可用，您就可以通过鼠标或代码手动的进行图表缩放。例如：

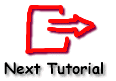
[C#]   
int x = points1.CalcXPos(2);   
int y = tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MaxYValue);   
int height = tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MinYValue) - tChart1.Axes.Left.CalcYPosValue(tChart1.Axes.Left.MaxYValue);   
int width = points1.CalcXPos(5) - x;   
Rectangle r = new Rectangle(x,y,width,height);   
tChart1.Zoom.Animated = true;   
tChart1.Zoom.AnimatedSteps = 100;   
tChart1.Zoom.ZoomRect(r);   
   
[VB.Net]   
Dim X As Integer = Points1.CalcXPos(2)   
Dim Y As Integer = TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MaxYValue)   
Dim Height As Integer = TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MinYValue) - TChart1.Axes.Left.CalcYPosValue(TChart1.Axes.Left.MaxYValue)   
Dim Width As Integer = Points1.CalcXPos(5) - X   
Dim R As New Rectangle(X, Y, Width, Height)   
TChart1.Zoom.Animated = True   
TChart1.Zoom.AnimatedSteps = 100   
TChart1.Zoom.ZoomRect(R)

**缩放事件**  
手动或通过代码进行放大操作，将会触发TChart.Zoomed事件。缩小操作将会触发TChart.UndoneZoom事件。  
  
**滚动**   
在所有的方向上进行滚动操作时默认可用的。可使用 Scroll.Allow属性来禁用滚动操作或限制某一个方向上的滚动操作。使用代码来实现滚动最简单的方法是用Axis Scroll方法：

[C#]   
tChart1.Axes.Bottom.Scroll(3, false);   
[VB.Net]   
TChart1.Axes.Bottom.Scroll(3, False)

它是不赋值的。'False'表示无论TeeChart是否允许在Series值之间的滚动限制。  
控制滚动的另外一种方法是通过代码定义Axis滚动的最大和最小值：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        int range = Convert.ToInt32(bar1.XValues.Maximum - bar1.XValues.Minimum / 2);   
        bar1.FillSampleValues(20);   
        tChart1.Panning.Allow = ScrollModes.None;   
        hScrollBar1.Value = range;   
        hScrollBar1.Minimum = range - 50;   
        hScrollBar1.Maximum = range + 50;   
}   
       
private void hScrollBar1\_Scroll(object sender, System.Windows.Forms.ScrollEventArgs e) {   
        tChart1.Axes.Bottom.Automatic = false;   
        tChart1.Axes.Bottom.Minimum = e.NewValue;   
        tChart1.Axes.Bottom.Maximum = e.NewValue + bar1.Count;   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
            Dim Range As Integer = Bar1.XValues.Maximum - Bar1.XValues.Minimum / 2   
            Bar1.FillSampleValues(20)   
            TChart1.Panning.Allow = Steema.TeeChart.ScrollModes.None   
            HScrollBar1.Value = Range   
            HScrollBar1.Minimum = Range - 50   
            HScrollBar1.Maximum = Range + 50   
End Sub   
   
Private Sub HScrollBar1\_Scroll(ByVal sender As Object, ByVal e As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll   
            TChart1.Axes.Bottom.Automatic = False   
            TChart1.Axes.Bottom.Minimum = e.NewValue   
            TChart1.Axes.Bottom.Maximum = e.NewValue + Bar1.Count   
End Sub

[](#Bookmark11)(See 1.1.10)     [](#Bookmark13)(See 1.1.12)

### 1.1.12 教程12 - 导出和导入图表

**教程 12 - 导出和导入图表**

本教程概述了导出不同格式的图表以及导入TeeChart自带的 .ten 格式的图表模板

**目录**

[**导出图表**](#Bookmark13_Export)   
     [支持的格式](#Bookmark13_Formats)   
     [导出实例](#Bookmark13_EgExport)   
  
[**导出图片**](#Bookmark13)  
     [BMP](#Bookmark13_BMP)   
     [JPEG](#Bookmark13_JPEG)   
     [WMF & EMF](#Bookmark13_WMF&EMF)   
     [GIF](#Bookmark13_GIF)   
     [PNG](#Bookmark13_PNG)   
     [TIFF](#Bookmark13_TIFF)   
     [Flex/Flash/SWF](#Bookmark13_SWF)   
  
[**导出数据**](#Bookmark13_Exportdat)   
     [Text, XML, HTML, XLS](#Bookmark13_Exportdat)   
  
[**TeeChart的'Ten'模板和数据导出导入的格式**](#Bookmark13)   
     [Ten文件](#Bookmark13_TenExport)   
  
[**导入**](#Bookmark13)     [将XML导入TeeChart](#Bookmark13_XMLImport)   
     [导入Ten格式的文件](#Bookmark13_TenFormat)   
     [导入Ten文件的实例](#Bookmark13_egImport)   
     [导入图表会引起效果的改变](#Bookmark13_EffectImport)   
     [其他数据源](#Bookmark13_OtherInput)

**导出图表**

**支持的格式**   
TeeChart所支持的导出格式。所有格式都可复制到任何一个文件夹、剪贴板或数据流中。

图片格式

 BMP: BitmapFormat类

 JPEG: JPEGFormat 类

 WMF & EMF: MetafileFormat 类

 GIF: GIFFormat 类

 PNG: PNGFormat 类

 TIFF: TIFFFormat 类

数据格式

 Text: TextFormat 类

 XML: XMLFormat 类

 HTML: HTMLFormat 类

 Excel: ExcelFormat 类

其他格式

Ten格式是一种灵活的格式，它可以存储图表属性信息和图表数据。这种格式的文件不仅小（依数据而定），而且是使用网络来实时更新图表客户端的理想选择。

 TEN (TeeChart): TemplateExport 类   
  
**导出实例**   
导出一个文件的功能是非常简单实用的。在大多数情况下，你只需要定义一个目标文件名。

[C#]   
private void button1\_Click(object sender, System.EventArgs e)   
            saveFileDialog1.Filter = "Images (\*.bmp)|\*.bmp|Images(\*.jpg)|\*.jpg|Images(\*.wmf)|\*.wmf|Images(\*.emf)|\*.emf|Images (\*.gif)|\*.gif|Images (\*.png)|\*.png|Images (\*.tiff)|\*.tiff";   
            saveFileDialog1.ShowDialog();   
            if(saveFileDialog1.FileName != "")   
                switch(saveFileDialog1.FilterIndex)   
                    case 1:   
                        tChart1.Export.Image.Bitmap.Save(saveFileDialog1.FileName);   
                        break;   
                    case 2:   
                        tChart1.Export.Image.JPEG.Save(saveFileDialog1.FileName);   
                        break;   
                    case 3:   
                        tChart1.Export.Image.Metafile.Enhanced = false;   
                        tChart1.Export.Image.Metafile.Save(saveFileDialog1.FileName);   
                        break;   
                    case 4:   
                        tChart1.Export.Image.Metafile.Enhanced = true;   
                        tChart1.Export.Image.Metafile.Save(saveFileDialog1.FileName);   
                        break;   
                    case 5:   
                        tChart1.Export.Image.GIF.Save(saveFileDialog1.FileName);   
                        break;   
                    case 6:   
                        tChart1.Export.Image.PNG.Save(saveFileDialog1.FileName);   
                        break;   
                    case 7:   
                        tChart1.Export.Image.TIFF.Save(saveFileDialog1.FileName);   
                        break;   
                   
               
           
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        SaveFileDialog1.Filter = "Images (\*.bmp)|\*.bmp|Images(\*.jpg)|\*.jpg|Images(\*.wmf)|\*.wmf|Images(\*.emf)|\*.emf|Images (\*.gif)|\*.gif|Images (\*.png)|\*.png|Images (\*.tiff)|\*.tiff"   
        SaveFileDialog1.ShowDialog()   
        If SaveFileDialog1.FileName <> "" Then   
            Select Case SaveFileDialog1.FilterIndex   
                Case 1   
                    TChart1.Export.Image.Bitmap.Save(SaveFileDialog1.FileName)   
                Case 2   
                    TChart1.Export.Image.JPEG.Save(SaveFileDialog1.FileName)   
                Case 3   
                    TChart1.Export.Image.Metafile.Enhanced = False   
                    TChart1.Export.Image.Metafile.Save(SaveFileDialog1.FileName)   
                Case 4   
                    TChart1.Export.Image.Metafile.Enhanced = True   
                    TChart1.Export.Image.Metafile.Save(SaveFileDialog1.FileName)   
                Case 5   
                    TChart1.Export.Image.GIF.Save(SaveFileDialog1.FileName)   
                Case 6   
                    TChart1.Export.Image.PNG.Save(SaveFileDialog1.FileName)   
                Case 7   
                    TChart1.Export.Image.TIFF.Save(SaveFileDialog1.FileName)   
            End Select   
        End If   
End Sub

**导出图片**

**BMP**   
导出BMP文件并添加colours 和 size两个参数  
例如

[C#]   
case 1:   
     Steema.TeeChart.Export.BitmapFormat bitMap = tChart1.Export.Image.Bitmap;   
     bitMap.Height = 500;   
     bitMap.Width = 700;   
     bitMap.Monochrome = true;   
     bitMap.Save(saveFileDialog1.FileName);   
     break;   
   
   
[VB.Net]   
Case 1   
     Dim BitMap As Steema.TeeChart.Export.BitmapFormat = TChart1.Export.Image.Bitmap   
     BitMap.Height = 500   
     BitMap.Width = 700   
     BitMap.Monochrome = True   
     BitMap.Save(SaveFileDialog1.FileName)

**JPEG**   
现在导出JPEG文件，所有的SaveToFile和SaveToStream都有speed和quality两个参数。

例如

[C#]   
case 2:   
Steema.TeeChart.Export.JPEGFormat jPEG = tChart1.Export.Image.JPEG;   
     jPEG.GrayScale = true;   
     jPEG.Height = 500;   
     jPEG.Width = 700;   
     jPEG.Quality = 100;   
     jPEG.Save(saveFileDialog1.FileName);   
     break;   
   
[VB.Net]   
Case 2   
Dim JPEGf As Steema.TeeChart.Export.JPEGFormat = TChart1.Export.Image.JPEG   
     JPEGf.Height = 500   
     JPEGf.Width = 700   
     JPEGf.GrayScale = True   
     JPEGf.Quality = 100   
     JPEGf.Save(SaveFileDialog1.FileName)

Quality的值（1-100）会使文件变大，从而导致网络传输速度变慢—虽然图片质量会更好！所以您需要找到一个适合应用程序的最佳平衡点。  
  
**WMF & EMF**   
将“Enhanced”设为Ture就可轻松地创建增强型元文件。  
例如

[C#]   
case 3:   
     Steema.TeeChart.Export.MetafileFormat metaFile = tChart1.Export.Image.Metafile;   
     metaFile.Height = 500;   
     metaFile.Width = 700;   
     metaFile.Enhanced = false;   
     metaFile.Save(saveFileDialog1.FileName);   
     break;   
   
[VB.Net]   
Case 3   
     Dim MetaFile As Steema.TeeChart.Export.MetafileFormat = TChart1.Export.Image.Metafile   
     MetaFile.Height = 500   
     MetaFile.Width = 700   
     MetaFile.Enhanced = False   
     MetaFile.Save(SaveFileDialog1.FileName)

**GIF**   
由于Micosoft 的.Net Framework 类库的运用，TeeChart .Net可以支持导出GIF格式的位图，特别是包含在 [System.Drawing.Imaging Namespace](ms-help://MS.VSCC/MS.MSDNVS/cpref/html/frlrfsystemdrawingimaging.htm)中的那些类。 时至今日对于开发者来说，他们依然不清楚在应用程序中使用这些类，是否需要从Unisys获得GIF专利的授权许可。如果还有其他疑问，我们建议您访问 [Unisys Website](http://www.unisys.com/).   
例如

[C#]   
case 5:   
     Steema.TeeChart.Export.GIFFormat gIF = tChart1.Export.Image.GIF;   
     gIF.Height = 500;   
     gIF.Width = 700;   
     gIF.ColorReduction = Steema.TeeChart.Export.GIFFormat.GIFColorReduction.GrayScale;   
     gIF.Save(saveFileDialog1.FileName);   
     break;   
   
[VB.Net]   
Case 5   
     Dim GIF As Steema.TeeChart.Export.GIFFormat = TChart1.Export.Image.GIF   
     GIF.Height = 500   
     GIF.Width = 700   
     GIF.ColorReduction = Steema.TeeChart.Export.GIFFormat.GIFColorReduction.GrayScale   
     GIF.Save(SaveFileDialog1.FileName)

**PNG**   
PNG格式延续了GIF格式的很多优点，但是也提供了一些GIF所没有的功能。PNG在GIF的基础上改进了它的性能，使它能够更好的显示图像；它就像从通过网络传输的图片一样，显示得越来越好。  
例如

[C#]   
case 6:   
     Steema.TeeChart.Export.PNGFormat pNG = tChart1.Export.Image.PNG;   
     pNG.Compression = 100;   
     pNG.GrayScale = true;   
     pNG.Height = 500;   
     pNG.Width = 700;   
     pNG.Save(saveFileDialog1.FileName);   
     break;   
   
[VB.Net]   
Case 6   
     Dim PNGf As Steema.TeeChart.Export.PNGFormat = TChart1.Export.Image.PNG   
     PNGf.Compression = 100   
     PNGf.GrayScale = True   
     PNGf.Height = 500   
     PNGf.Width = 700   
     PNGf.Save(SaveFileDialog1.FileName)

**TIFF**   
TIFF是一种灵活并可进行扩展的格式，它支持多种平台和图像处理程序。  
例如

[C#]   
case 7:   
     Steema.TeeChart.Export.TIFFFormat tIFF = tChart1.Export.Image.TIFF;   
     tIFF.Compression = Steema.TeeChart.Export.TIFFFormat.TIFFCompression.LZW;   
     tIFF.Height = 500;   
     tIFF.Width = 700;   
     tIFF.Save(saveFileDialog1.FileName);   
     break;   
   
[VB.Net]   
Case 7   
     Dim TIFF As Steema.TeeChart.Export.TIFFFormat = TChart1.Export.Image.TIFF   
     TIFF.Compression = Steema.TeeChart.Export.TIFFFormat.TIFFCompression.LZW   
     TIFF.Height = 500   
     TIFF.Width = 700   
     TIFF.Save(SaveFileDialog1.FileName)

**Adobe Flex/Flash/SWF**现在TeeChart还可以导出Adobe Flex脚本文件（\*.mxml），然后可以通过Flex免费编译器(mxmlc.exe) 进行编译后创建一个\*swf(Flash)文件。关于如何在运行时实现上述功能，以及如何添加动画到已导出图表的综合实例，可以在'Welcome !\New in Canvas\Flash "CompactCharts"'下的功能演示中找到。

**导出数据**

四种数据导出格式：

 Text: TextFormat 类

 XML: XMLFormat 类

 HTML: HTMLFormat 类

 Excel: ExcelFormat 类

以上的控件与图表Series有关系，他们可以导出数据到任何文件，数据流或是剪贴板。下面的实例是从图表Series中将数据导出到HTML表格中：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
            Random rnd = new Random();   
            char c = new char();   
   
            for(int i = 0; i < 10; ++i)   
                c = Convert.ToChar(65 + i);   
                bar1.Add(Convert.ToDouble(i),rnd.Next(100),c.ToString(),Color.Red);   
               
            Steema.TeeChart.Export.DataExport dataExport = new Steema.TeeChart.Export.DataExport(tChart1.Chart);   
            dataExport.HTML.IncludeHeader = true;   
            dataExport.HTML.IncludeIndex = true;   
            dataExport.HTML.IncludeLabels = true;   
            dataExport.HTML.Series = bar1;   
            dataExport.HTML.Save(@"C:.htm");   
           
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
            Dim C As New Char()   
            Dim i As New Integer()   
   
            For i = 0 To 10   
                C = Convert.ToChar(65 + i)   
                Bar1.Add(Convert.ToDouble(i), Rnd() \* 100, C.ToString(), Color.Red)   
            Next   
            Dim DataExport As New Steema.TeeChart.Export.DataExport(TChart1.Chart)   
            DataExport.HTML.IncludeHeader = True   
            DataExport.HTML.IncludeIndex = True   
            DataExport.HTML.IncludeLabels = True   
            DataExport.HTML.Series = Bar1   
            DataExport.HTML.Save("C:.htm")   
End Sub

以上输出的随机数据集如下所示：  
<table border="1">   
<tr><td>Index</td><td>Text</td><td>bar1</td></tr>   
<tr><td>0</td><td>A</td><td>69</td></tr>   
<tr><td>1</td><td>B</td><td>93</td></tr>   
<tr><td>2</td><td>C</td><td>18</td></tr>   
<tr><td>3</td><td>D</td><td>88</td></tr>   
<tr><td>4</td><td>E</td><td>42</td></tr>   
<tr><td>5</td><td>F</td><td>33</td></tr>   
<tr><td>6</td><td>G</td><td>58</td></tr>   
<tr><td>7</td><td>H</td><td>43</td></tr>   
<tr><td>8</td><td>I</td><td>63</td></tr>   
<tr><td>9</td><td>J</td><td>49</td></tr>   
</table>

**TeeChart的'Ten'模板和数据导出/入格式**

**Ten 文件**   
Ten文件是TeeChart自带的用来保存图表和数据的模板格式。  
修改的图表属性将保存在模板中，当将模板导入到一个新的图表时它们就会重现。  
优点:

 Ten 文件非常小，大多数情况下比纯图画格式更有优势（更快）。??

 该模板的目标图表是‘活的’，它可以缩放及滚动，同时也可以修改它的属性。

 根据您的喜好，数据可以随机的包含在ten模版中。   
  
例如

[C#]   
private void button1\_Click(object sender, System.EventArgs e)   
            saveFileDialog1.Filter = "TenFile (.ten)|\*.ten";   
            saveFileDialog1.ShowDialog();   
            if(saveFileDialog1.FileName != "")   
                tChart1.Export.Template.IncludeData = true;   
                tChart1.Export.Template.Save(saveFileDialog1.FileName);   
               
           
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        SaveFileDialog1.Filter = "TenFile (.ten)|\*.ten"   
        SaveFileDialog1.ShowDialog()   
        If SaveFileDialog1.FileName <> "" Then   
            TChart1.Export.Template.IncludeData = True   
            TChart1.Export.Template.Save(SaveFileDialog1.FileName)   
        End If   
End Sub

**导入**

**将XML导入到TeeChart**  
从TextBox导入XML.

例如   
[C#]   
protected override void button1\_Click(object sender, System.EventArgs e)   
            tChart1.Series.RemoveAllSeries();   
            Steema.TeeChart.Styles.Points points1 = new Stee

ma.TeeChart.Styles.Points(tChart1.Chart);   
            points1.Title = "Points Series";   
            xmlSource1.Series = points1;   
            xmlSource1.SeriesNode = "Points Series";   
            xmlSource1.Load(textBox2.Text);   
   
   
            tChart1.Refresh();

从URL导入XML .

例如  
[C#]   
protected override void button2\_Click(object sender, System.EventArgs e)   
            tChart1.Series.RemoveAllSeries();   
            xmlSource1.Chart = tChart1.Chart;   
            xmlSource1.Load("http://www.steema.com/SampleNETData.xml");   
   
            tChart1.Refresh();

**导入Ten 格式文件**

从本地文件源或http数据源中导入一个已保存的Tee文件  
**导入实例**

例如

从文件中导入

[C#]   
tChart1.Import.Template.LoadFileDialog();   
[VB.Net]   
TChart1.Import.Template.LoadFileDialog()

URL..

[C#]   
tChart1.Import.Template.FromURL("http://www.berneda.com/demo.ten");   
[VB.Net]   
TChart1.Import.Template.FromURL("http://www.berneda.com/demo.ten")

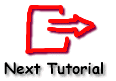
Stream..

[C#]   
System.IO.MemoryStream tenStream = new System.IO.MemoryStream();   
tChart1.Export.Template.Save(tenStream);   
tChart2.Import.Template.Load(tenStream);   
[VB.Net]   
Dim TenStream As New System.IO.MemoryStream()   
TChart1.Export.Template.Save(TenStream)   
TChart2.Import.Template.Load(TenStream)

**导入图表会引起效果的改变**  
由于.ten文件只保存标准图表定义的改变，所以目标图表的属性变化与被导入的图表无关，而是由原始图表来执行。这是一个图表页脚的实例。如果.ten文件没有设定页脚和目标图表，那么页脚将会在图表导入后才显示。这将是一个理想的结果。如果您期望在导入图表时引起变化，它能在图表导出后逆向影响所期望的结果，那么您就能创建一个新图表例子，如下所示：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        Bar bar1 = new Bar(tChart1.Chart);   
        bar1.FillSampleValues(20);   
   
private void button1\_Click(object sender, System.EventArgs e)   
        saveFileDialog1.Filter = "TenFile (.ten)|\*.ten";   
        saveFileDialog1.ShowDialog();   
        if(saveFileDialog1.FileName != "")   
            tChart1.Export.Template.IncludeData = true;   
            tChart1.Export.Template.Save(saveFileDialog1.FileName);   
           
   
private void button2\_Click(object sender, System.EventArgs e)   
        tChart1.Dispose();   
        tChart1 = new Steema.TeeChart.TChart();   
        ini\_Chart();   
   
private void ini\_Chart()   
        this.tChart1.Header.Lines = new string[] "TeeChart";   
        this.tChart1.Location = new System.Drawing.Point(24, 24);   
        this.tChart1.Name = "tChart1";   
        this.tChart1.Size = new System.Drawing.Size(400, 240);   
        this.tChart1.TabIndex = 5;   
        this.Controls.AddRange(new System.Windows.Forms.Control[] this.tChart1);   
   
private void button3\_Click(object sender, System.EventArgs e)   
        tChart1.Import.Template.LoadFileDialog();   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Bar1.FillSampleValues(20)   
End Sub   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        SaveFileDialog1.Filter = "TenFile (.ten)|\*.ten"   
        SaveFileDialog1.ShowDialog()   
        If Not SaveFileDialog1.FileName = "" Then   
            TChart1.Export.Template.IncludeData = True   
            TChart1.Export.Template.Save(SaveFileDialog1.FileName)   
        End If   
End Sub   
Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click   
        TChart1.Dispose()   
        TChart1 = New Steema.TeeChart.TChart()   
        Ini\_Chart()   
End Sub   
Private Sub Ini\_Chart()   
        Me.TChart1.Header.Lines = New String() "TeeChart"   
        Me.TChart1.Location = New System.Drawing.Point(16, 16)   
        Me.TChart1.Name = "TChart1"   
        Me.TChart1.Size = New System.Drawing.Size(400, 296)   
        Me.TChart1.TabIndex = 3   
        Me.Controls.AddRange(New System.Windows.Forms.Control() Me.TChart1)   
End Sub   
Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click   
        TChart1.Import.Template.LoadFileDialog()   
End Sub

**其他数据源**

关于如何从其他数据库信息源导入数据请参考  
[Database Access tutorial](#Bookmark9)(See 1.1.8).   
[](#Bookmark12)(See 1.1.11)     [](#Bookmark14)(See 1.1.13)

### 1.1.13 教程13 - 在图表面板上自定义绘图

**教程 13 - 在图表面板上自定义绘图**

TeeChart通过Canvas提供了许多绘图的功能。借由Canvas您可以在图表面板的任何地方添加图形、直线和文本，同时还可以设定他们的颜色，画笔及格式刷。

**目录**

[**TeeChart Canvas**](#Bookmark14_Canvas)   
     [绘图顺序](#Bookmark14_DrawOrder)   
     [确保自定义绘图项目保存到Canvas中](#Bookmark14_PermanentDraw)   
     [绘制直线](#Bookmark14_AddLines)   
     [Canvas的画笔及格式刷](#Bookmark14_PenBrush)  
     [添加 2D 图形](#Bookmark14_AddShapes)   
     [添加 3D 图形](#Bookmark14_Add3DShapes)   
     [添加文本](#Bookmark14_AddText)   
     [应用实例](#Bookmark14_egCanvas)

**TeeChart Canvas**

**绘图顺序**   
 在运用TeeChart的Canvas方法时，请记住绘图顺序是非常重要的。如果在图表上先画线然后再添加一系列的Series数据点，这样会导致线被透支。  
这里有四个基本的图表绘图事件，其顺序如下所示：

 BeforeDraw 事件

 BeforeDrawAxes 事件

 BeforeDrawSeries 事件

 AfterDraw 事件   
  
例如:

[C#]   
private bool afterDraw;   
private bool beforeDraw;   
private bool beforeDrawAxis;   
private bool beforeDrawSeries;   
   
private void Form1\_Load(object sender, System.EventArgs e) {    
     SetFlags(ref beforeDraw);   
     Bar bar1 = new Bar(tChart1.Chart);   
     bar1.FillSampleValues(20);   
     radioButton1.Checked = true;   
}   
private void SetFlags(ref bool Flag) {   
     beforeDraw = false;   
     afterDraw = false;   
     beforeDrawAxis = false;   
     beforeDrawSeries = false;   
     Flag = true;   
}   
private void DrawShape(Steema.TeeChart.Drawing.Graphics3D gg) {   
     gg.Brush.Color = Color.Yellow;   
     gg.Pen.Visible = true;   
     gg.Pen.Style = System.Drawing.Drawing2D.DashStyle.Dash;   
     gg.Brush.Visible = true;   
     gg.Ellipse(1,1,gg.Chart.Width - 1,gg.Chart.Height - 1);   
}   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
     if(afterDraw) {   
     DrawShape(g);   
    }   
}   
private void tChart1\_BeforeDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
     if(beforeDraw) {   
     DrawShape(g);   
    }   
}   
private void tChart1\_BeforeDrawAxes(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
     if(beforeDrawAxis) {   
     DrawShape(g);   
    }   
}   
private void tChart1\_BeforeDrawSeries(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
     if(beforeDrawSeries) {   
     DrawShape(g);   
    }   
}   
private void radioButton4\_Click(object sender, System.EventArgs e) {   
     SetFlags(ref afterDraw);   
     tChart1.Refresh();   
}   
private void radioButton3\_Click(object sender, System.EventArgs e) {   
     SetFlags(ref beforeDrawSeries);   
     tChart1.Refresh();   
}   
private void radioButton2\_Click(object sender, System.EventArgs e) {   
     SetFlags(ref beforeDrawAxis);   
     tChart1.Refresh();   
}   
private void radioButton1\_Click(object sender, System.EventArgs e) {   
     SetFlags(ref beforeDraw);   
     tChart1.Refresh();   
}   
   
[VB.Net]       
Private BeforeDraw As Boolean   
Private BeforeDrawAxis As Boolean   
Private BeforeDrawSeries As Boolean   
Private AfterDraw As Boolean   
   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
    SetFlags(BeforeDraw)   
    Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
    Bar1.FillSampleValues(20)   
    RadioButton1.Checked = True   
End Sub   
Private Sub SetFlags(ByRef Flag As Boolean)   
    BeforeDraw = False   
    BeforeDrawAxis = False   
    BeforeDrawSeries = False   
    AfterDraw = False   
    Flag = True   
End Sub   
Private Sub DrawShape(ByVal gg As Steema.TeeChart.Drawing.Graphics3D)   
    gg.Brush.Color = Color.Yellow   
    gg.Pen.Visible = True   
    gg.Pen.Style = Drawing.Drawing2D.DashStyle.Dash   
    gg.Brush.Visible = True   
    gg.Ellipse(1, 1, gg.Chart.Width - 1, gg.Chart.Height - 1)   
End Sub   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
    If AfterDraw = True Then   
     DrawShape(g)   
    End If   
End Sub   
Private Sub TChart1\_BeforeDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.BeforeDraw   
    If BeforeDraw = True Then   
     DrawShape(g)   
    End If   
End Sub   
Private Sub TChart1\_BeforeDrawAxes(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.BeforeDrawAxes   
    If BeforeDrawAxis = True Then   
     DrawShape(g)   
    End If   
End Sub   
Private Sub TChart1\_BeforeDrawSeries(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.BeforeDrawSeries   
    If BeforeDrawSeries = True Then   
     DrawShape(g)   
    End If   
End Sub   
Private Sub RadioButton4\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles RadioButton4.Click   
    SetFlags(AfterDraw)   
    TChart1.Refresh()   
End Sub   
Private Sub RadioButton3\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles RadioButton3.Click   
    SetFlags(BeforeDrawSeries)   
    TChart1.Refresh()   
End Sub   
Private Sub RadioButton2\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles RadioButton2.Click   
    SetFlags(BeforeDrawAxis)   
    TChart1.Refresh()   
End Sub   
Private Sub RadioButton1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles RadioButton1.Click   
    SetFlags(BeforeDraw)   
    TChart1.Refresh()   
End Sub

**确保自定义绘图项目保存到Canvas中**   
在一个图表事件中，如果您没有调用Canvas绘图代码，那么自定义绘图将不会永久的被保存在Canvas中，因此当一个应用程序被缩小或另一个Windows将其覆盖时，一些添加的数据会丢失。您的代码不需要直接存放在Chart事件中；如果您的代码在BeforeDrawSeries/AfterDraw中，那么用户绘图项目将被保存为活动的图表窗体，同时还要检查Draw方法中所设定的标记，当其属性为真时即可运行您的绘图代码，详情可参考上一个实例。  
  
**绘制直线**   
现在添加一条Canvas直线:   
例如 (画一条从左上角到右下角的对角线)

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        line1.FillSampleValues(20);   
        line1.VertAxis = VerticalAxis.Both;   
        line1.HorizAxis = HorizontalAxis.Both;   
        tChart1.Aspect.View3D = false;   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Point s = new Point(tChart1.Axes.Left.Position, tChart1.Axes.Top.Position);   
        Point e = new Point(tChart1.Axes.Right.Position, tChart1.Axes.Bottom.Position);   
        g.MoveTo(s);   
        g.LineTo(e,0);   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Line1.FillSampleValues(20)   
        Line1.VertAxis = Steema.TeeChart.VerticalAxis.Both   
        Line1.HorizAxis = Steema.TeeChart.HorizontalAxis.Both   
        TChart1.Aspect.View3D = False   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim S As New Point(TChart1.Axes.Left.Position, TChart1.Axes.Top.Position)   
        Dim E As New Point(TChart1.Axes.Right.Position, TChart1.Axes.Bottom.Position)   
        g.MoveTo(S)   
        g.LineTo(E, 0)   
End Sub

在3D图表中，由于3D的正交位移，图表区域的坐标轴位置将会发生偏移，我们可以相应的移动直线:    
例如 (在3D图表的图表区域绘制一条从左上角到右下角的对角线)

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        line1.FillSampleValues(20);   
        line1.VertAxis = VerticalAxis.Both;   
        line1.HorizAxis = HorizontalAxis.Both;   
        tChart1.Aspect.Chart3DPercent = 50;   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Steema.TeeChart.Drawing.Point3D s = new Steema.TeeChart.Drawing.Point3D();   
        s.X = tChart1.Axes.Left.Position;   
        s.Y = tChart1.Axes.Top.Position;   
        s.Z = 0;   
   
        Steema.TeeChart.Drawing.Point3D e = new Steema.TeeChart.Drawing.Point3D();   
        e.X = tChart1.Axes.Right.Position;   
        e.Y = tChart1.Axes.Bottom.Position;   
        e.Z = tChart1.Aspect.Width3D;   
       
        g.MoveTo(s);   
        g.LineTo(e);   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Line1.FillSampleValues(20)   
        Line1.VertAxis = Steema.TeeChart.VerticalAxis.Both   
        Line1.HorizAxis = Steema.TeeChart.HorizontalAxis.Both   
        TChart1.Aspect.Chart3DPercent = 50   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim S As New Steema.TeeChart.Drawing.Point3D()   
        S.X = TChart1.Axes.Left.Position   
        S.Y = TChart1.Axes.Top.Position   
        S.Z = 0   
   
        Dim E As New Steema.TeeChart.Drawing.Point3D()   
        E.X = TChart1.Axes.Right.Position   
        E.Y = TChart1.Axes.Bottom.Position   
        E.Z = TChart1.Aspect.Width3D   
   
        g.MoveTo(S)   
        g.LineTo(E)   
End Sub

**Canvas 画笔及格式刷**   
上面的直线是先定义最后一个对象，然后再用画笔和格式刷绘制的。但这样不一定是您想要的画笔。可按以下的实例改变您的画笔：   
例如 (在绘制直线前定义画笔)

[C#]   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Point p5 = new Point(line1.CalcXPos(5), line1.CalcYPos(5));   
        Point p15 = new Point(line1.CalcXPos(15), line1.CalcYPos(15));   
        g.Pen.DashCap = System.Drawing.Drawing2D.DashCap.Triangle;   
        g.Pen.EndCap = System.Drawing.Drawing2D.LineCap.DiamondAnchor;   
        g.Pen.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot;   
        g.Pen.Transparency = 70;   
        g.Pen.Width = 3;   
        g.Pen.Color = Color.BlueViolet;   
        g.MoveTo(p5);   
        g.LineTo(p15, 0);   
}   
   
[VB.Net]   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim P5 As New Point(Line1.CalcXPos(5), Line1.CalcYPos(5))   
        Dim P15 As New Point(Line1.CalcXPos(15), Line1.CalcYPos(15))   
        g.Pen.DashCap = System.Drawing.Drawing2D.DashCap.Triangle   
        g.Pen.EndCap = System.Drawing.Drawing2D.LineCap.DiamondAnchor   
        g.Pen.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot   
        g.Pen.Transparency = 70   
        g.Pen.Width = 3   
        g.Pen.Color = Color.BlueViolet   
        g.MoveTo(P5)   
        g.LineTo(P15, 0)   
End Sub

**添加2D图形**   
添加Canvas图形的方法跟Canvas直线很类似。以下实例是在图表的中心区域添加一个矩形:

2D 图表

2D图表只支持2D图形   
例如

[C#]   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Size s = new Size(100,100);   
        Point l = new Point(g.ChartXCenter - (s.Width / 2), g.ChartYCenter - (s.Height / 2));   
        Rectangle r = new Rectangle(l,s);   
        g.Pen.Color = Color.Aquamarine;   
        g.Brush.Color = Color.Blue;   
        g.Rectangle(r);   
}   
   
[VB.Net]   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim S As New Size(100, 100)   
        Dim L As New Point(g.ChartXCenter - (S.Width / 2), g.ChartYCenter - (S.Height / 2))   
        Dim R As New Rectangle(L, S)   
        g.Pen.Color = Color.Aquamarine   
        g.Brush.Color = Color.Blue   
        g.Rectangle(R)   
End Sub

3D 图形

在3D图表上您也可以将矩形移到Z平面上，详情可参考RectangleWithZ 方法。在这个实例中，矩形框最初放在图表的Z立面，在中途将其移向图表的背面（靠近背立面）.

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        point3DSeries1.LinePen.Visible = false;   
        point3DSeries1.FillSampleValues(20);   
        point3DSeries1.VertAxis = VerticalAxis.Both;   
        point3DSeries1.HorizAxis = HorizontalAxis.Both;   
        tChart1.Aspect.Chart3DPercent = 50;   
        tChart1.Axes.Depth.Visible = true;   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Size s = new Size(100,100);   
        Point l = new Point(tChart1.Axes.Left.Position, g.ChartYCenter - (s.Height / 2));   
        Rectangle r = new Rectangle(l,s);   
        g.Pen.Color = Color.Aquamarine;   
        g.Brush.Color = Color.Blue;   
        g.Rectangle(r, tChart1.Aspect.Width3D/2);   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Point3DSeries1.LinePen.Visible = False   
        Point3DSeries1.FillSampleValues(20)   
        Point3DSeries1.VertAxis = Steema.TeeChart.VerticalAxis.Both   
        Point3DSeries1.HorizAxis = Steema.TeeChart.HorizontalAxis.Both   
        TChart1.Aspect.Chart3DPercent = 50   
        TChart1.Axes.Depth.Visible = True   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim S As New Size(100, 100)   
        Dim L As New Point(TChart1.Axes.Left.Position, g.ChartYCenter - (S.Height / 2))   
        Dim R As New Rectangle(L, S)   
        g.Pen.Color = Color.Aquamarine   
        g.Brush.Color = Color.Blue   
        g.Rectangle(R, TChart1.Aspect.Width3D / 2)   
End Sub

**添加3D图形**   
您可以在3D图表中添加3D图形。这个实例是在图表矩形的中间区域绘制一个立方体:

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        point3DSeries1.LinePen.Visible = false;   
        point3DSeries1.FillSampleValues(20);   
        tChart1.Aspect.Chart3DPercent = 50;   
        tChart1.Legend.Visible = false;   
        tChart1.Axes.Depth.Visible = true;   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Size s = new Size(50,50);   
        Point p = new Point(g.ChartXCenter - (s.Width/2), g.ChartYCenter - (s.Height/2));   
        Rectangle r = new Rectangle(p,s);   
        g.Cube(r, 0, 20, true);   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Point3DSeries1.LinePen.Visible = False   
        Point3DSeries1.FillSampleValues(20)   
        TChart1.Aspect.Chart3DPercent = 50   
        TChart1.Legend.Visible = False   
        TChart1.Axes.Depth.Visible = True   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim S As New Size(50, 50)   
        Dim P As New Point(g.ChartXCenter - (S.Width / 2), g.ChartYCenter - (S.Height / 2))   
        Dim R As New Rectangle(P, S)   
        g.Cube(R, 0, 20, True)   
End Sub

**添加文本**

2D文本的位置

在最后绘制的矩形框中添加文本:   
例如

[C#]   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        string text = "My Text";   
        Size s = new Size(150, 50);   
        Point p = new Point(g.ChartXCenter - (s.Width/2), g.ChartYCenter - (s.Height/2));   
        Rectangle r = new Rectangle(p,s);   
        g.Pen.Color = Color.Blue;   
        g.Rectangle(r);   
   
        g.TextOut(Convert.ToInt32(g.ChartXCenter - (g.TextWidth(text)/2)), Convert.ToInt32(g.ChartYCenter - (g.TextHeight(text)/2)), text);   
}   
   
[VB.Net]   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim Text As String = "My Text"   
        Dim S As New Size(50, 50)   
        Dim P As New Point(g.ChartXCenter - (S.Width / 2), g.ChartYCenter - (S.Height / 2))   
        Dim R As New Rectangle(P, S)   
        g.Pen.Color = Color.Blue   
        g.Rectangle(R)   
   
        g.TextOut(Convert.ToInt32(g.ChartXCenter - (g.TextWidth(Text) / 2)), Convert.ToInt32(g.ChartYCenter - (g.TextHeight(Text) / 2)), Text)   
End Sub

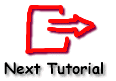
3D文本的位置

您可以通过重载z坐标的TextOut 方法，在各个3D平面内放置文本。   
例如

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        point3DSeries1.FillSampleValues(20);   
        point3DSeries1.LinePen.Visible = false;   
        tChart1.Aspect.Chart3DPercent = 50;   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        string text = "My Text";   
        g.TextOut(g.ChartXCenter, g.ChartYCenter, tChart1.Aspect.Width3D / 2, text);   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Point3DSeries1.FillSampleValues(20)   
        Point3DSeries1.LinePen.Visible = False   
        TChart1.Aspect.Chart3DPercent = 50   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim Text As String = "My Text"   
        g.TextOut(g.ChartXCenter, g.ChartYCenter, TChart1.Aspect.Width3D / 2, Text)   
End Sub

**应用实例**   
这个例子使用了Series的第三个和第十个值，在它们之间绘制一条直线，并显示第一个和最后一个点的值及他们之间的差别：  
例如

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        tChart1.Aspect.View3D = false;   
        line1.FillSampleValues(20);   
}   
   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        if(tChart1.Series.Count > 0){   
            if(tChart1.Series[0].Count > 10) {   
                Series s = tChart1.Series[0];   
                int h = Convert.ToInt32(g.TextHeight("H"));   
                Point p1 = new Point(s.CalcXPos(3), s.CalcYPos(3));   
                Point p2 = new Point(s.CalcXPos(10), s.CalcYPos(10));   
                g.Pen.Color = Color.Blue;   
                g.Pen.Width = 2;   
                g.Pen.Style = System.Drawing.Drawing2D.DashStyle.Dash;   
                g.MoveTo(p1);   
                g.LineTo(p2, 0);   
                g.TextOut(p1.X, p1.Y - h, "Point value: " + s.YValues[3].ToString());   
                g.TextOut(p2.X, p2.Y, "Point value: " + s.YValues[10].ToString());   
                g.TextOut(p2.X, p2.Y + h, "Change is: " + Convert.ToString(s.YValues[3] - s.YValues[10]));   
            }   
        }   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        TChart1.Aspect.View3D = False   
        Line1.FillSampleValues(20)   
End Sub   
   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        If TChart1.Series.Count > 0 Then   
            If TChart1.Series(0).Count > 10 Then   
                Dim S As Steema.TeeChart.Series = TChart1.Series(0)   
                Dim H As Integer = Convert.ToInt32(g.TextHeight("H"))   
                Dim P1 As New Point(S.CalcXPos(3), S.CalcYPos(3))   
                Dim P2 As New Point(S.CalcXPos(10), S.CalcYPos(10))   
                g.Pen.Color = Color.Blue   
                g.Pen.Width = 2   
                g.Pen.Style = System.Drawing.Drawing2D.DashStyle.Dash   
                g.MoveTo(P1)   
                g.LineTo(P2, 0)   
                g.TextOut(P1.X, P1.Y - H, "Point value: " & S.YValues(3))   
                g.TextOut(P2.X, P2.Y, "Point value: " & S.YValues(10))   
                g.TextOut(P2.X, P2.Y + H, "Change is: " & (S.YValues(3) - S.YValues(10)))   
            End If   
        End If   
End Sub

[](#Bookmark13)(See 1.1.12)     [](#Bookmark15)(See 1.1.14)

### 1.1.14 教程14 - 打印图表

**教程 14 - 打印图表   
目录**

[**标准打印**](#Bookmark15_StdPrinting)   
     [简单的打印指令](#Bookmark15_PrintChart)   
     [打印方向](#Bookmark15_Orientation)   
     [打印预览](#Bookmark15_Preview)   
     [黑白打印](#Bookmark15_Greyscale)   
  
[**扩展的打印方法**](#Bookmark15_ExtPrinting)   
     [打印多页图表](#Bookmark15_PrintPartial)   
     [在同一页上打印多张图表](#Bookmark15_PrintMany)   
     [图表打印与其他打印混合输出](#Bookmark15_PrintPartialHandle)

**标准打印**

TeeChart Pro提供了标准打印方法，使其按原本的样式打印Onscreen图表。  
  
**简单打印指令**   
运用Print方法打印图表。这个方法可以像屏幕上显示的一样将图表打印出来:

[C#]   
tChart1.Printer.Print();   
   
[VB.Net]   
TChart1.Printer.Print()

**打印方向**   
在默认时即使没有定义Print方法，您也可以通过设定布尔型的 landscape参数来实现横向与纵向打印。在打印完成后，Orientation的默认值将会再一次生效。Orientation默认值可通过设定Landscape属性（设为true为横向，false为纵向）来进行修改：

[C#]   
tChart1.Printer.Landscape = true;   
tChart1.Printer.Print();   
   
[VB.Net]   
TChart1.Printer.Landscape = True   
TChart1.Printer.Print()

**打印预览**   
PrintPreview窗口将会显示图表打印出来的样式。您可以在Print Preview窗口中修改打印参数。调用PrintPreview的方法如下所示:

[C#]   
tChart1.Printer.Preview();   
   
[VB.Net]   
TChart1.Printer.Preview()

**黑白打印**   
在使用黑白打印机进行打印时，您应注意将图表的颜色转换为黑白色调可易于识别。为了方便，您可以在图表Series上添加格式刷以便在打印时更易于区分Series。  
  
您也可以通过设定Grayscale属性将黑白报表打印为彩色报表。

[C#]   
tChart1.Printer.Grayscale = true;   
tChart1.Printer.Print(true);   
   
[VB.Net]   
TChart1.Printer.Grayscale = True   
TChart1.Printer.Print(True)

**扩展的打印方法**

**打印多页报表**   
使用BeginPrint() 和EndPrint()将报表传送到打印机可避免溢出。BeginPrint()和EndPrint()是用来开始和结束打印机的工作。多张图表可以在同一页/打印任务中完成，同时还包括用户自定义的输入。

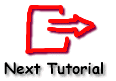
Example (Prints 2 Charts to a page):   
[C#]   
private void button1\_Click(object sender, System.EventArgs e) {   
        tChart1.Printer.BeginPrint();   
        tChart1.Printer.Print(tChart2.Chart,new Rectangle(100,10,300,200));   
        tChart1.Printer.Print(new Rectangle(100,300,300,200));   
        tChart1.Printer.EndPrint();   
}   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        TChart1.Printer.BeginPrint()   
        TChart1.Printer.Print(TChart2.Chart, New Rectangle(100, 10, 300, 200))   
        TChart1.Printer.Print(New Rectangle(100, 300, 300, 200))   
        TChart1.Printer.EndPrint()   
End Sub

**在同一页上进行多张图表的打印预览**   
 现在越来越多的图表都支持打印预览功能。可以通过设定Print方法的Rectangle来控制图表的位置。

例如 (打印预览两张图表):   
[C#]   
private void button1\_Click(object sender, System.EventArgs e) {   
        tChart1.Printer.BeginPrint();   
        tChart1.Printer.Print(tChart2.Chart,new Rectangle(100,10,300,200));   
        tChart1.Printer.Print(new Rectangle(100,300,300,200));   
        tChart1.Printer.Preview();   
}   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        TChart1.Printer.BeginPrint()   
        TChart1.Printer.Print(TChart2.Chart, New Rectangle(100, 10, 300, 200))   
        TChart1.Printer.Print(New Rectangle(100, 300, 300, 200))   
        TChart1.Printer.Preview()   
End Sub

**图表打印与其他打印混合输出**  
使用ChartPrint()事件可将TeeChart打印与非图表打印混合输出  
以下的示例获取了TeeChart Headers的文本，并在同一页中用两个TChart对象来进行打印：

[C#]   
private void button1\_Click(object sender, System.EventArgs e) {   
        tChart1.Printer.BeginPrint();   
        tChart1.Printer.Print(tChart2.Chart,new Rectangle(100,10,300,200));   
        tChart1.Printer.Print(new Rectangle(100,300,300,200));   
        tChart1.Printer.EndPrint();   
}   
   
private void tChart1\_ChartPrint(object sender, System.Drawing.Printing.PrintPageEventArgs e) {   
        e.Graphics.DrawString("Chart: "+((Steema.TeeChart.ChartPrintJob)sender).Chart.Header.Text,   
            this.Font,new SolidBrush(Color.Black),100,((Steema.TeeChart.ChartPrintJob)sender).ChartRect.Bottom+10);   
}   
   
[VB.Net]   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        TChart1.Printer.BeginPrint()   
        TChart1.Printer.Print(TChart2.Chart, New Rectangle(100, 10, 300, 200))   
        TChart1.Printer.Print(New Rectangle(100, 300, 300, 200))   
        TChart1.Printer.EndPrint()   
End Sub   
   
Private Sub TChart1\_ChartPrint(ByVal sender As Object, ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles TChart1.ChartPrint   
        e.Graphics.DrawString("Chart: " & (CType(sender, Steema.TeeChart.ChartPrintJob)).Chart.Header.Text, \_   
        Me.Font, New SolidBrush(Color.Black), 100, (CType(sender, Steema.TeeChart.ChartPrintJob)).ChartRect.Bottom + 10)   
End Sub

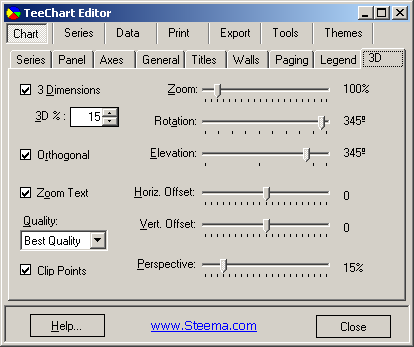
[](#Bookmark14)(See 1.1.13)     [](#Bookmark16)(See 1.1.15)

### 1.1.15 教程15 - 3D图表

**教程 15 -3D 图表   
目录**

[**图表的显示模式**](#Bookmark16_ChartDisplay)   
     [(3D)图表编辑页面](#Bookmark16_3DChartEditor)   
     [2D 图表](#Bookmark16_2DChart)   
     [3D 正交图表](#Bookmark16_OrthogChart)   
     [3D "标准" Windows 图表](#Bookmark16_3DNativeChart)

**图表的显示模式**

**(3D)图表编辑页面**   
您在设计时可在Chart Editor中设定2D/3D主要的显示选项.   
  
  
  
**2D 图表**   
在设计时，可在Chart Editor中选择2D图表。在Editor页面中通过取消3D复选框，可将图表设为2D模式。可用以下的方法在程序运行的任何时候将图表改为2D图表：

[C#]   
tChart1.Aspect.View3D = false;   
   
[VB.Net]   
TChart1.Aspect.View3D = False

在使用2D图表时没有出现特殊的问题。 所有对象都处于图表面板或图表矩形区的相对位置，不需要考虑到3D偏移位移的问题。  
  
**3D正交图表**   
3D正交模式的默认方法是获取3D'效果'。通过设定 TChart1.Aspect.Chart3D属性（运行时也可在Chart Editor页面中设定），TeeChart可绘制出一定的正交倾斜的效果。3D正交图表不能旋转，因为它的底部轴面始终是水平的。  
在自定义绘制Canvas时，当设定3D正交图表时，要考虑Width3D和Height3D的3D偏移位移。如果您希望在图表的背立面绘制一个项目，那么就需要添加偏移位移。参考以下的示例:   
  
这个示例是穿过图表前立面绘制一条对角线:

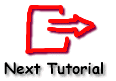
[C#]   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        Point p1 = new Point(tChart1.Axes.Left.Position, tChart1.Axes.Top.Position);   
        Point p2 = new Point(tChart1.Axes.Right.Position, tChart1.Axes.Bottom.Position);   
        g.MoveTo(p1);   
        g.LineTo(p2, 0);   
}   
   
[VB.Net]   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim P1 As New Point(TChart1.Axes.Left.Position, TChart1.Axes.Top.Position)   
        Dim P2 As New Point(TChart1.Axes.Right.Position, TChart1.Axes.Bottom.Position)   
        g.MoveTo(P1)   
        g.LineTo(P2, 0)   
End Sub

这个示例是穿过图表背立面绘制一条对角线:

[C#]   
private void tChart1\_AfterDraw(object sender, Steema.TeeChart.Drawing.Graphics3D g) {   
        int width3D = tChart1.Aspect.Width3D;   
        Point p1 = new Point(tChart1.Axes.Left.Position, tChart1.Axes.Top.Position);   
        Point p2 = new Point(tChart1.Axes.Right.Position, tChart1.Axes.Bottom.Position);   
        g.MoveTo(p1, width3D);   
        g.LineTo(p2, width3D);   
}   
   
[VB.Net]   
Private Sub TChart1\_AfterDraw(ByVal sender As Object, ByVal g As Steema.TeeChart.Drawing.Graphics3D) Handles TChart1.AfterDraw   
        Dim Width3D As Integer = TChart1.Aspect.Width3D   
        Dim P1 As New Point(TChart1.Axes.Left.Position, TChart1.Axes.Top.Position)   
        Dim P2 As New Point(TChart1.Axes.Right.Position, TChart1.Axes.Bottom.Position)   
        g.MoveTo(P1, Width3D)   
        g.LineTo(P2, Width3D)   
End Sub

在运行时若要为3D图表设计显示属性，则可使用Aspect接口（\*注.不是所有的选项对正交图表都是可用的）.   
  
**3D“标准”Windows图表**   
标准的Windows 3D 模式提供了图表的旋转和90°立体效果。您可以将图表进行整体缩放（图表也可进行部分缩放，请参考[Zoom/Scroll](#Bookmark12)(See 1.1.11)教程）  
  
在设计时要选择TeeChart的标准Windows 3D模式只需在3DEditor页面取消Orthogonal复选框。

[C#]   
tChart1.Aspect.Orthogonal = false;   
[VB.Net]   
TChart1.Aspect.Orthogonal = False

在Chart Editor中，非正交时可以同时调整Elevation和Rotation的微调控件，他们都允许在设计时改变他们的显示属性  
对TeeChart Canvases 更进一步的说明请参考 [Custom drawing 教程](#Bookmark14)(See 1.1.13).   
  
[](#Bookmark15)(See 1.1.14)     [](#Bookmark17)(See 1.1.16)

### 1.1.16 教程16 - TeeChart附加面板组件

**教程16 - 附加面板组件  
Contents**

[**使用附加的TeeChart控件**](#Bookmark17_AddControls)   
     [选择与新增组件](#Bookmark17_SelectControl)   
     [将组件与Tchart关联](#Bookmark17_AssocToChart)   
     [ButtonColor](#Bookmark17_ButtonColor)   
     [ButtonPen](#Bookmark17_ButtonPen)   
     [ChartListBox](#Bookmark17)   
     [Commander](#Bookmark17_TeeCommander)   
     [Editor](#Bookmark17_EditorComponent)   
     [GalleryPanel](#Bookmark17_GalleryPanel)

**使用附加的TeeChart控件**

TeeChart for.Net包含了许多附加控件，将它们添加到工程中可以增强图表应用程序在运行时的功能.   
  
ButtonColor    
ButtonPen    
ChartListBox  
Commander    
Editor    
GalleryPanel    
  
**选择并嵌入组件**   
当您将TeeChart for .Net组件添加到Visual Studio .Net 工具箱时，附加控件也会自动进行添加。  
  
**将组件与TeeChart相关联**   
 在一个TChart中，可以运用Chart属性将Commander与其他非图表组件相关联。  
例如

[C#]   
commander1.Chart = tChart1;   
[VB.Net]   
Commander1.Chart = TChart1

**ButtonColor**ButtonColor是一个颜色编辑器，它可以与TeeChart对象的Color属性相关联，修改图表的颜色和透明度。  
  
例如

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        tChart1.Aspect.View3D = false;   
        line1.FillSampleValues(20);   
        colorBand1.Axis = tChart1.Axes.Left;   
        colorBand1.Start = line1.YValues.Maximum \* 0.8;   
        colorBand1.End = line1.YValues.Minimum \* 1.2;   
        buttonPen1.Pen = colorBand1.Pen;   
}   
   
private void buttonColor1\_Click(object sender, System.EventArgs e) {   
        colorBand1.Color = buttonColor1.Color;   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        TChart1.Aspect.View3D = False   
        Line1.FillSampleValues(20)   
        ColorBand1.Axis = TChart1.Axes.Left   
        ColorBand1.Start = Line1.YValues.Maximum \* 0.8   
        ColorBand1.End = Line1.YValues.Minimum \* 1.2   
        ButtonPen1.Pen = ColorBand1.Pen   
End Sub   
           
Private Sub ButtonColor1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonColor1.Click   
        ColorBand1.Color = ButtonColor1.Color   
End Sub

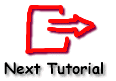
**ButtonPen**ButtonPen是一个线条编辑器，它能与TeeChart的Pen属性相关联，修改它的可见度，颜色，结尾，样式，宽度及透明度。  
例如

[C#]   
private void Form1\_Load(object sender, System.EventArgs e) {   
        tChart1.Aspect.View3D = false;   
        line1.FillSampleValues(20);   
        colorBand1.Axis = tChart1.Axes.Left;   
        colorBand1.Start = line1.YValues.Maximum \* 0.8;   
        colorBand1.End = line1.YValues.Minimum \* 1.2;   
        buttonPen1.Pen = colorBand1.Pen;   
}   
   
private void buttonColor1\_Click(object sender, System.EventArgs e) {   
        colorBand1.Color = buttonColor1.Color;   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        TChart1.Aspect.View3D = False   
        Line1.FillSampleValues(20)   
        ColorBand1.Axis = TChart1.Axes.Left   
        ColorBand1.Start = Line1.YValues.Maximum \* 0.8   
        ColorBand1.End = Line1.YValues.Minimum \* 1.2   
        ButtonPen1.Pen = ColorBand1.Pen   
End Sub   
           
Private Sub ButtonColor1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonColor1.Click   
        ColorBand1.Color = ButtonColor1.Color   
End Sub

**TeeListBox**   
ChartListBox在运行时会显示一列Series。ChartListBox可提供用户Chart Series部分自定义及显示/隐藏功能。  
  
**Commander**   
Commander在运行时为用户提供导航和参数变化控制功能（通过TeeChart Editor）。选择导航按钮并按下鼠标左键，然后在图表面板上通过拖动来进行图表的移动。  
  
**Editor**Editor允许自定义 TeeChart 的编辑器:

[C#]

private void Form1\_Load(object sender, System.EventArgs e) {   
        editor1.Chart = tChart1;   
        line1.FillSampleValues(20);   
}   
   
private void button1\_Click(object sender, System.EventArgs e) {   
        editor1.DefaultTab = Steema.TeeChart.Editors.ChartEditorTabs.Legend;   
        editor1.Title = "My Application Title";   
        editor1.ShowModal();   
}   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Editor1.Chart = TChart1   
        Line1.FillSampleValues(20)   
End Sub   
           
Private Sub ButtonColor1\_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles ButtonColor1.Click   
        Editor1.DefaultTab = Steema.TeeChart.Editors.ChartEditorTabs.Legend   
        Editor1.Title = "My Application Title"   
        Editor1.ShowModal()   
End Sub

**GalleryPanel**   
GalleryPanel显示了 TeeChart Series的一个自定义的Gallery。  
  
[](#Bookmark16)(See 1.1.15)     [](#Bookmark18)(See 1.1.17)

### 1.1.17 教程17 - 设计时 运行时及授权规定

**Tutorial 17 - 设计时，运行时及授权规定   
目录**

[**TeeChart 运行时规定**](#Bookmark18_Runtime)   
     [TeeChart运行时文件](#Bookmark18_Files)   
  
[**运行时编辑器帮助支持**](#Bookmark18_EditorHelp)  
     [配置](#Bookmark18_EditorHelpCfg)   
  
[**TeeChart语言支持**](#Bookmark18_Lang)   
     [配置](#Bookmark18_LangCfg)   
     [设计时语言](#Bookmark18_DesignLang)   
  
[**TeeChart自定义主题支持**](#Bookmark18_Theme)   
  
[**ASP.NET 应用程序支持**](#Bookmark18_ASPNET)   
     [配置](#Bookmark18_ASPCfg)   
  
[**设计及运行时授权设置**](#Bookmark18)   
     [说明](#Bookmark18_LICDESC)   
     [操作注意事项](#Bookmark18_LICOPSETUP)   
     [在dll中使用TeeChart](#Bookmark18_USEINDLL)   
     [在插件dll中使用TeeChart](#Bookmark18_USEINPLUGIN)   
     [在VC++中使用TeeChart](#Bookmark18_USEINVC)   
     [运行时分发授权](#Bookmark18_LICRUNTIME)   
     [授权问题疑难解答](#Bookmark18_LICERRORS)

**TeeChart运行时规定**

本教程主要从几个方面介绍了TeeChart应用程序运行时安装的问题 。  
  
**TeeChart运行时文件**您的TeeChart应用程序需要通过TeeChart.dll来进行分发，但是如果您的应用程序是使用Steema.TeeChart.Languages来命名的，那么就要使用TeeChart.Languages.dll进行分发。您还需要决定是否将这些dll文件安装在本机上，例如，在您的应用程序目录下的bin目录中，用AssemblyFolders注册表项注册这些文件，或将它们安装在Global Assembly Cache (GAC)中。请通过[此链接](ms-help://MS.VSCC/MS.MSDNVS/cpguide/html/cpconglobalassemblycache.htm) 学习GAC是怎样使您的TeeChart.Net 程序集可以在不同的应用程序中进行轻松共享。

**运行时编辑器帮助支持**

在运行时操作TeeChart编辑器，可以使用户修改您应用程序的图表设置。TeeChart自带的设计时程序设计帮助并不是运行时的扩展，您只是可以在运行时，通过Help按钮添加自己的帮助文档到图表编辑器中，。  
  
**配置**

 您的帮助文档必须是chm格式（HTML Help v1）

 当您将帮助文档安装在以下注册位置时，该文档必须是在运行时已进行了注册。如下所示:

     [HKEY\_LOCAL\_MACHINE\\SOFTWARE\\Microsoft\\Windows\\HTML Help]   
        "YourHelpfile.chm"="drive:\directory"

**TeeChart语言支持**

TeeChart的语言，换句话说，也就是TeeChart编辑器及其他TeeChart对话框用不同语言进行显示的能力，目前它只支持在运行时进行注册。  
  
**配置**  
如果要在运行时要改变语言，您只需在您的项目中引用TeeChart.Languages.dll，然后使用如下代码即可：  
  
fnnnt 范例  
这里需放置一个Combo box来进行语言选择:

private void Form1\_Load(object sender, System.EventArgs e)   
        commander1.Chart = tChart1;   
        tChart1.Aspect.View3D = false;   
        line1.FillSampleValues(20);   
        comboBox1.Items.Add("Arabic");   
        comboBox1.Items.Add("Brazilian");   
        comboBox1.Items.Add("Catalan");   
        comboBox1.Items.Add("Chinese Simple");   
        comboBox1.Items.Add("Chinese Traditional");   
        comboBox1.Items.Add("Danish");   
        comboBox1.Items.Add("Dutch");   
        comboBox1.Items.Add("English");   
        comboBox1.Items.Add("Finnish");   
        comboBox1.Items.Add("French");   
        comboBox1.Items.Add("Galician");   
        comboBox1.Items.Add("German");   
        comboBox1.Items.Add("Hebrew");   
        comboBox1.Items.Add("Hellenic");   
        comboBox1.Items.Add("Hungarian");   
        comboBox1.Items.Add("Indonesian");   
        comboBox1.Items.Add("Italian");   
        comboBox1.Items.Add("Japanese");   
        comboBox1.Items.Add("Korean");   
        comboBox1.Items.Add("Norwegian");   
        comboBox1.Items.Add("Polish");   
        comboBox1.Items.Add("Portuguese");   
        comboBox1.Items.Add("Russian");   
        comboBox1.Items.Add("Slovak");   
        comboBox1.Items.Add("Slovene");   
        comboBox1.Items.Add("Spanish");   
        comboBox1.Items.Add("Swedish");   
        comboBox1.Items.Add("Turkish");   
        comboBox1.Items.Add("Ukranian");   
        comboBox1.SelectedItem = "English";   
   
   
private void comboBox1\_SelectedIndexChanged(object sender, System.EventArgs e)   
        switch(comboBox1.SelectedIndex)   
            case 0: Steema.TeeChart.Languages.Arabic.Activate(); break;   
            case 1: Steema.TeeChart.Languages.Brazilian.Activate(); break;   
            case 2: Steema.TeeChart.Languages.Catalan.Activate(); break;   
            case 3: Steema.TeeChart.Languages.ChineseSimp.Activate(); break;   
            case 4: Steema.TeeChart.Languages.ChineseTrad.Activate(); break;   
            case 5: Steema.TeeChart.Languages.Danish.Activate(); break;   
            case 6: Steema.TeeChart.Languages.Dutch.Activate(); break;   
            case 7: Steema.TeeChart.Languages.English.Activate(); break;   
            case 8: Steema.TeeChart.Languages.Finnish.Activate(); break;   
            case 9: Steema.TeeChart.Languages.French.Activate(); break;   
            case 10: Steema.TeeChart.Languages.Galician.Activate(); break;   
            case 11: Steema.TeeChart.Languages.German.Activate(); break;   
            case 12: Steema.TeeChart.Languages.Hebrew.Activate(); break;   
            case 13: Steema.TeeChart.Languages.Hellenic.Activate(); break;   
            case 14: Steema.TeeChart.Languages.Hungarian.Activate(); break;   
            case 15: Steema.TeeChart.Languages.Indonesian.Activate(); break;   
            case 16: Steema.TeeChart.Languages.Italian.Activate(); break;   
            case 17: Steema.TeeChart.Languages.Japanese.Activate(); break;   
            case 18: Steema.TeeChart.Languages.Korean.Activate(); break;   
            case 19: Steema.TeeChart.Languages.Norwegian.Activate(); break;   
            case 20: Steema.TeeChart.Languages.Polish.Activate(); break;   
            case 21: Steema.TeeChart.Languages.Portuguese.Activate(); break;   
            case 22: Steema.TeeChart.Languages.Russian.Activate(); break;   
            case 23: Steema.TeeChart.Languages.Slovak.Activate(); break;   
            case 24: Steema.TeeChart.Languages.Slovene.Activate(); break;   
            case 25: Steema.TeeChart.Languages.Spanish.Activate(); break;   
            case 26: Steema.TeeChart.Languages.Swedish.Activate(); break;   
            case 27: Steema.TeeChart.Languages.Turkish.Activate(); break;   
            case 28: Steema.TeeChart.Languages.Ukrainian.Activate(); break;   
        

**设计时语言**   
TeeChart能设置一种默认的设计时语言，当TeeChaart被添加到一个Form中时便可立即生效，而不需要引用TeeChart.Languages.dll。在您第一次打开图表项目时，语言检索表中活动的Language将会生效，或者也可以通过单击图表右键菜单中的'Options' 选项来设定语言。  
  
**Ie.**

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Steema Software\TeeChart.NET]   
"Language"=dword:00000011  //Hex 11 is decimal 17 (Japanese)

**ASP.NET应用程序支持**

本节包括ASP.NET WebForm应用程序分发的相关资料。  
**配置**只有当WebChart的TempChart属性设为'File'时，WebChart才可以通过注册表设置将图表存储在临时文件夹中。而如果 WebChart的TempChart属性设为 'Session'，那么在运行时服务器上是禁止注册的。关于WebChart应用程序配置的更多信息，请参考教程9，ASP.NET应用程序。  
  
**当TempChart设为File时:**   
如果您希望将临时文件夹写到硬盘，File的设置是非常有用的。默认情况下，当TeeChart安装时，将对临时文件夹位置及对应的虚拟文件夹名创建注册表项:  其注册表项如下所示：

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Steema Software\TeeChart.NET]    
"VirtualShare"="/TeeChartForNET"    
"ShareFolder"="C:\Program Files\Steema Software\TeeChart for .NET v3\TeeChartForNET"

如果已有默认的安装位置，那么以上的设置将在以英语为标准语言的机器中进行创建。TeeChartNET虚拟文件夹将通过TeeChart的安装程序来创建。您只有创建必要的IIS虚拟共享位置来反映新的注册表值时，您才可以更改这些注册设置。  
  
WebChart是将文件保存在ShareFolder注册表中以\_Chart\_tmp命名的文件夹下。  
  
TeeChart中的临时文件是不能删除的，尽管它可以自动执行一些类似Windows计划任务的操作，但还是请检查当前所用版本TeeChart中包含的东西。

**设计及运行时授权设置**

本节主要讲述配置原理及按照TeeChart设计时与运行时授权架构所要执行的设置步骤。  
  
**说明**   
TeeChart是使用基于Microsoft System.Component.LicenseProvider 类的授权控件。当用TeeChart安装程序安装TeeChart时，会添加一个注册表授权许可，其可见的部分在HKEY\_LOCAL\_MACHINE\SOFTWARE\Steema Software\TeeChart.NET下找到，授权许可如下所示 :

"DesignKey"="Steema.TeeChart.TChart is a licensed component."

授权许可加上对应的‘未公开’关键代码，设计时就可在Visual Studio.NET中使用TeeChart，同时也允许在运行时编译可执行文件（.exe）和库文件（.dll），所以TeeChart.dll与TeeChart.languages.dll文件都可以通过已编译的应用程序来进行分发，但不会为安装用于设计时组件的TeeChart提供选项。   
**操作注意事项**   
一个包含TChart，Chart或WebChart组件的项目必须包含一个Licenses.licx文件。在添加任何一个图表时，该文件会通过Visual Studio IDE自动添加到TChart (Windows Form Chart)或WebChart (Web Form Chart)项目中。对于非设计时图表应用程序，在运行时手动创建图表的方法如下所示：

"Steema.TeeChart.Chart c = new Steema.TeeChart.Chart();"

然后您必须对该项目手动添加授权文件，在Utils文件夹中包含一个您可能会用到的示例文件。您可以在VS.NET的解决方案资源管理器中右键单击工程，然后选择'**添加 | 添加现有项**' ，其文件类型选为“所有文件” ，再选择 'Utils\Licenses.Licx' 来添加授权文件。   
  
**在dll中使用TeeChart**  
如果您已经成功的将具有授权的TeeChart导入到dll中，那么请注意，如果你在其他应用程序中使用该dll文件，你必须确保**将TeeChart授权添加到应用程序（最终版本）的Licenses.licx文件中**，即使该应用程序本身没有直接引用TeeChart。如果您最终用户的应用程序不在Visual Studio .NET中开发，那么授权文件和授权本身就不需要在最后进行编译。   
在一台拥有开发者授权的机器中，如果您要在开发环境中复用您的dll文件，那么请确认该复用操作是否包含在下面的内容中<a href="#USEINPLUGIN">在插件的dll文件中使用TeeChart</a>如果您认为您的组件（使用TeeChart的组件）要被一个拥有TeeChart类和方法的容器复用或引用，但是这台机器上没有开发者授权，那么请确认该机器已在设计时被授权使用TeeChart。一些分发或集体作业的环境问题需要通过一个特殊的授权来解决，通过info@steema.com联系Steema 工作人员，我们可以帮您解决授权问题。  
  
**在插件dll中使用TeeChart**   
TeeChart .NET的标准版支持在动态下载插件中使用授权控件。该插件是一个包含TeeChart的典型组件，它可以在运行时通过容器应用程序，并使用NET'Assembly.Load'或是类似的方法来进行动态下载。为了在容器应用程序中无需licenses.licx文件便可下载及运行TeeChart，TeeChart提供了一个可供选择的默认构造函数，在编译dll文件集时作为一个自变量，并在dll文件上（并非容器应用程序）进行授权检测。请注意该容器应用程序不能通过编写代码来直接调用Teechart。它只可以打开及运行TeeChart dll文件，而实际上dll文件能自动调用TeeChart。可从容器应用程序中进行图表调用的特例是，如果TeeChart的方法是通过一些方法进行'封装’的（第二次被调用），那么将会在一个无关的接口中来进行定义。对于所提到的这些调用，例子中所描述的内容可以进行修改。  
  
  
插件  
创建时嵌入在TeeChart中的插件，需要通过将其作为参数来加载TeeChart。为了顺利进行授权检查，重载的构造函数仅能在第一次打开TeeChart时使用。此后TeeChart可通过常规方法来使用。下面是一个使用的示例。在这个示例中，创建表单时为了促使初始化的授权检查可创建一个TeeChart的‘虚拟体’。  
范例:   
在这个示例中， PluginShared.Factory是一个主机应用程序引用的接口，它提供了一个叫'CreateForm'的方法。因为嵌入在TeeChart中的组件需用PluginShared.Factory接口，因此也提供了CreateForm方法，但是它可以自行引用（在导入TeeChart时）。该主机应用程序允许没有TeeChart相关内容的特殊组件调用CreateForm方法。  
  
请注意。嵌入TeeChart中的插件必须包含Licenses.licx文件，并能在具有一个有效TeeChar开发者授权的机器上进行编译。  
  
嵌入TeeChart中的插件(eg. PluginWithTeeChart.dll)   
-------------------------------------------------------------------------------------------   
Class:

namespace Plugin   
{   
   public class FactoryImpl : PluginShared.Factory     
    {   
        public System.Windows.Forms.Form CreateForm()   
        {   
            try   
            {   
                return new PluginWithTeeChart.FormWithChart();   
            }   
            catch (Exception e)   
            {   
                MessageBox.Show("failed to create plugin with error: " + e.ToString());   
                return null;   
            }   
        }   
    }   
}

Form:

namespace PluginWithTeeChart   
{   
    public partial class FormWithChart : Form   
    {   
        public FormWithChart()   
        {   
            //discardable Chart forces License check   
            Steema.TeeChart.TChart tempChart = new Steema.TeeChart.TChart(this);   
            tempChart = null;   
            InitializeComponent(); //<---- Normal application code. Could run here various TeeCharts without need for overloaded constructor   
        }   
    }   
}

容器应用程序(eg. HostApp.exe)   
----------------------------------------------------------------------   
容器可以通过以下的方法来导入TeeChart嵌套组件。HostApp引用常见的接口插件是非常重要的。FactoryImpl提供了CreateForm方法。如下所示：

void LoadTeeChartPlugin(string pluginName)   
{   
  string pluginName = @".dll";   
  try   
  {   
    Debug.WriteLine("Loading plugin assembly " + pluginName);   
   
    Assembly assembly = Assembly.LoadFrom(pluginName);   
   
    string factoryTypeName = "Plugin.FactoryImpl";   
   
    Type t = assembly.GetType(factoryTypeName);   
    if (t == null)   
    {   
      MessageBox.Show("Type: " + factoryTypeName + " not found in plugin assembly");   
      return;   
    }   
   
    PluginShared.Factory factory = (PluginShared.Factory)assembly.CreateInstance(t.ToString());   
    System.Windows.Forms.Form f = factory.CreateForm();   
    f.ShowDialog();   
  }   
  catch (Exception ex)   
  {   
    MessageBox.Show(ex.ToString());   
  }   
}

**在 Visual C++.NET中使用TeeChart**  
授权在VC++中的操作相对于其他.NET编程语言而言，并没有如期望的那么透明化 。您可以按下面所描述的步骤为VC++项目顺利的配置授权文件，它是由microsoft.public.dotnet.framework新闻组的 Tim Adler提供的。  
**引用:**   
“我的工作区很简单：在一个新的C#工程（与您的C++应用程序同名）中，创建一个新的解决方案。接着创建一个虚拟表单并删除其中所有需要注册的组件。或者也可以对每个组件自动生成的licenses.licx 文件进行编辑 。然后再创建应用程序，接着跳转到debug目录下，将"\*.licenses"文件复制到您的C++应用程序的基目录中。并在您C++ app中按Project Options->Linker->Input->Embed Managed Resource进行文件设置，最后指到该授权文件，执行'重构'。”  
  
**Licenses.licx 文件**   
该文件是一个文本文件，它包含了至少一个或多个通用文件，以下三行内容请参考:

Steema.TeeChart.Chart, TeeChart   
Steema.TeeChart.TChart, TeeChart   
Steema.TeeChart.Web.WebChart, TeeChart

其内容应包括至少一个以您的项目创建的类名，代表Steema.TeeChart.TChart基于项目的一个窗体表单 ， 对于WebForm项目中您通过代码和Steema.TeeChart.Web.WebChart创建的项目，Steema.TeeChart.Chart包括了TeeChart中的WebChart组件。  
当向一个Form NET添加图表时，它会自动添加一个稍微完整且适合的版本，并包括了组件版本号和公共密匙。

Steema.TeeChart.TChart, TeeChart, Version=1.1.1864.22788, Culture=neutral, PublicKeyToken=9c8126276c77bdb7

**将授权添加到旧的‘未授权’项目中**  
在使用无授权文件的老版本TeeChart创建Form应用程序时应注意，您可以从工具面板的Steem项中拖拽图表到项目表单中，从而快速的生成该授权文件，然后再删除它。最新生成的图表将使IDE添加Licenses.licx文件。  
**运行时分发**   
已编译Licenses.licx文件的项目.exe或 .dll文件，可能会被作为一个运行时应用程序来进行分发，除了TeeChart.dll（如果有必要还有 TeeChart.Languages.dll ）以外，该应用程序不需要安装任何TeeChart运行时授权文件或注册密钥。  
  
**授权问题疑难解答**   
我们已出现可能会发生的错误如下:   
- 在编译时的错误信息:

"Could not transform licenses file 'licenses.licx' into a binary resource."

一方面可能是开发者授权注册表密匙的位置不对。请检查在本章节中讲述的密匙条目，看其中是否有任何明显的问题或遗漏。解决该问题的另一方法是使用TeeChart安装程序来重装TeeChart，安装人员需设置注册密匙。如果该问题依然存在，那么请通过技术支持论坛联系TeeChart技术支持团队。  
  
- 当在没有安装设计表项的机器上，运行编译（运行时）输出应用程序时出现的授权错误 。   
检查包含Licenses.licx文件的项目。它可能在向表单内添加图表时已自动添加该文件。无论哪种情况，请确定Licenses.licx文件是作为嵌入源存在于您的项目中。您需要在项目的解决方案资源管理器窗口中，激活“查看所有文件”来查看Licenses.licx。  
  
- 该问题出现的另一个原因可能是licenses.licx文件的内容是错误的。请参考本章节前面的描述来检查其内容。

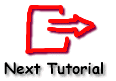
**如何在设计时和运行时设定自定义主题**

自定义主题是通过几行简单代码创建小的\*.xml 文件来实现的:

tChart1.Export.Theme.Save(@"C:\Steema\MyTheme.xml");

自定义主题囊括了许多图表的绘图设置，其涉及到字体，格式刷与画笔颜色及面板风格，轴，页眉，页脚，图例等。图表一旦创建完成，这些主题将会在图表编辑器的Theme标签中显示出来，它在这儿显示的名字与文件中给出的名字是一样的。另外这里还有在本地机器上指定的文件夹注册设置， 所以图表编辑器清楚在哪里查找这些文件：

[HKEY\_LOCAL\_MACHINE\SOFTWARE\Steema Software\TeeChart.NET]    
"ThemeFolder"="C:\Program Files\Steema Software\TeeChart for .NET v3\Themes"

在这个密匙中，将您自定义的主题轻松地添加（\*.xml文件）到指定文件夹中，并通过图表编辑器读取和显示他们.。  
    
  
[](#Bookmark17)(See 1.1.16)     [](#Bookmark19)(See 1.1.18)

**可参考**

[TeeChart Pro .Net 教程:目录:教程9 - ASP.NET 应用程序](#Bookmark10)(See 1.1.9)

### 1.1.18 教程18 - TeeChart 工具集

**教程18 - TeeChart 工具集   
目录**

[**引言**](#Bookmark19_Introduction)   
  
[**Series工具**](#Bookmark19_Series)    [**Cursor工具**](#Bookmark19_Cursor)         [设计时](#Bookmark19_Design4)   
         [运行时](#Bookmark19_Run4)   
  
    [**Drag Marks 工具**](#Bookmark19_DragMarks)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Drag Point 工具**](#Bookmark19_DragPoint)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Draw Line 工具**](#Bookmark19_DrawLine)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Gantt Drag 工具**](#Bookmark19_GanttDrag)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Image 工具**](#Bookmark19_Image)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)    [**Mark Tips 工具**](#Bookmark19_MarkTips)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Nearest Point 工具**](#Bookmark19_NearestPoint)         [Design-time](#Bookmark19_Design9)   
         [Run-time](#Bookmark19_Run9)    [**Pie Slices 工具**](#Bookmark19_PieSlices)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Series Animation 工具**](#Bookmark19_SeriesAnimation)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Extra Legend 工具**](#Bookmark19_ExtraLegend)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Surface Nearest工具**](#Bookmark19_SurfaceNearest)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Fibonacci工具**](#Bookmark19_Fibonacci)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Series Region 工具**](#Bookmark19_SeriesRegion)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Legend Palette 工具**](#Bookmark19_LegendPalette)   
        [设计时](#Bookmark19)   
        [运行时](#Bookmark19)   
  
     [**Series Statistics 工具**](#Bookmark19_SeriesStatistics)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Clip Series 工具**](#Bookmark19_ClipSeries)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
  
[**Axis工具**](#Bookmark19_Axis)   
  
    [**Axis Arrows工具**](#Bookmark19_AxisArrows)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Color Band 工具**](#Bookmark19_ColorBand)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Color Line 工具**](#Bookmark19_ColorLine)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Grid Band 工具**](#Bookmark19_GridBand)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Axis Scroll 工具**](#Bookmark19_AxisScroll)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**WebChart 工具**](#Bookmark10)**(See 1.1.9)**   
  
[**Hotspot 工具**](#Bookmark10)**(See 1.1.9)**   
  
[**Scroll 工具**](#Bookmark10)**(See 1.1.9)**   
  
[**Zoom 工具**](#Bookmark10)**(See 1.1.9)**   
  
[**其他工具**](#Bookmark19_Other)[**3D Grid Transpose 工具**](#Bookmark19_3DGrid)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Annotation 工具**](#Bookmark19_Annotation)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Page Number 工具**](#Bookmark19_Page)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Rotate 工具**](#Bookmark19_Rotate)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Legend ScrollBar工具**](#Bookmark19_LegendScrollBar)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**2D Lighting 工具**](#Bookmark19_2DLighting)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**SubChart 工具**](#Bookmark19_SubChart)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Fader 工具**](#Bookmark19_Fader)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Rectangle 工具**](#Bookmark19_Rectangle)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Selector 工具**](#Bookmark19_Selector)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
[**Transpose Series 工具**](#Bookmark19_TransposeSeries)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**DataTable 工具**](#Bookmark19_DataTable)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Banner 工具**](#Bookmark19_Banner)   
        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)  
  
    [**Magnify 工具**](#Bookmark19_Magnify)        [设计时](#Bookmark19_Design4)   
        [运行时](#Bookmark19_Run4)

**引言**

本教程主要介绍TeeChart for.Net工具，简单的代码操作便可使您或您应用程序的用户在TeeCharts中添加附加的功能。这里有10种新的工具供您使用:

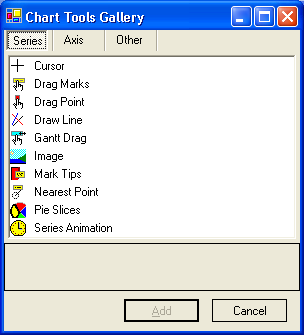
**Series 工具**

**Cursor 工具**   
Cursor 工具可在图表中添加光标，其可与个别的数据序列或整个图表相关联。光标可能是横向的，纵向的或两者皆有（十字的）。  
**Drag Marks 工具**  
Drag Marks 工具可以对已选定的Series（数据序列）标记鼠标拖拽。  
**Drag Point 工具**   
Drag Marks 工具可以对已选定的Series（数据序列）标记鼠标拖拽。  
**Draw Line 工具**   
Draw Line 工具可完全自定义在图表中通过鼠标来对直线进行绘制，选择，拖动及删除的操作。  
**Gantt Drag 工具**   
Gantt Drag 工具能拖动和/或调整Gantt Series的滚动条。  
**Image 工具**   
Image 工具可在特定图表的Series背后绘制图片（bitmap, jpeg, gif, png, pcx）。  
**Mark Tips 工具**   
Mark Tips 工具可用来显示鼠标滑过（或点击）一个Series点时默认的Hint窗口。  
**Nearest Point 工具**   
Nearest Point 工具可将标记移到最接近光标的点。该工具可与Series相关联或应用于整个图表。  
**Pie Slices 工具**   
不管是在扇形区域周围绘制边界或是在饼状Series外部延伸，Pie Slices工具强调的是鼠标下的扇形区。  
**Series Animation 工具**   
Series Animation 工具可形成一个Series Points（数据序列点）的渐增式动画。  
**Extra Legend 工具**   
Extra Legend 工具可在图表里的任何位置显示自定义图例。  
**Surface Nearest 工具**   
Surface Nearest 工具可在用户将鼠标悬停在数据数组上时，会直观显示所选表面单元。  
**Fibonacci工具**Fibonacci 工具显示Fibonacci弧形或基于特定趋势线的扇形。  
**Series Region 工具**   
Series Region 工具可在数据序列曲线下描绘一块区域。  
**Legend Palette 工具**Legend Palette 工具显示由3D数据序列彩色调色板生成的图例。  
**Series Statistics 工具**Series Statistics 工具可对特定的数据序列估算出许多标准的统计量度。  
**Clip Series 工具**Clip Series工具限制与Series相关联的轴边界外Series（数据数列）的绘制。

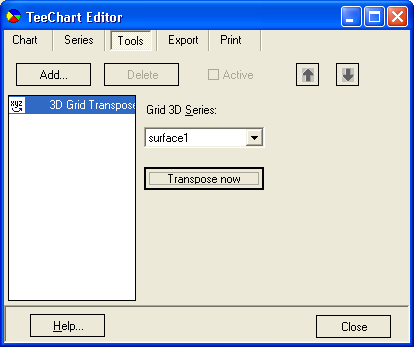
**Axis 工具**

**Axis Arrows 工具**   
Axis Arrows 工具在坐标轴的始末显示可编辑的箭头。当点击这些箭头时，可滚动图表的坐标轴。  
**Color Band 工具**   
Color Band 工具可在图表坐标轴的内部区域添加可编辑的颜色带。可对这些颜色带设置它们的横纵坐标值。  
**Color Line 工具**   
Color Line 工具可在图表中添加有色的直线，可对它们设置横纵坐标值。  
**Grid Band 工具**   
Grid Band 工具可在特定轴及位置的网格线上显示有色的矩形（带）。   
**Axis Scroll 工具**   
Axis Scroll 工具可通过拖动鼠标来实现图表坐标轴的滚动。

**其他工具**

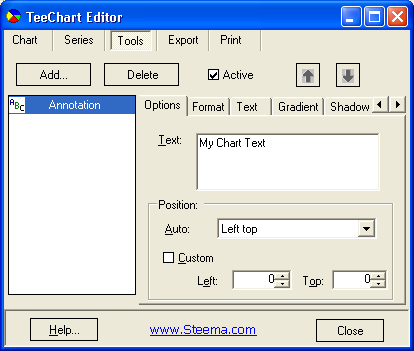
**3D Grid Transpose 工具**   
3D Grid Transpose 工具可通过交换X与Z轴使基于网格的数据序列旋转90度。   
**Annotation 工具**   
Annotation 工具可允许在图表中显示文本。  
**Page Number 工具**   
Page Number 工具可显示一个包含页码选择系统的可编辑文本框。  
**Rotate 工具**   
在图表上拖拽鼠标时，Rotate工具可改变图表的立体度及旋度。  
**Legend ScrollBar 工具**   
Legend ScrollBar工具可通过拖拽鼠标来滚动坐标轴。  
**2D Lighting 工具**   
2D Lighting 工具可在图表Canvas上显示一个2维的“灯光”效果。  
**SubChart工具**   
SubChart工具允许在一个单一的图表内包含多个图表。  
**Fader 工具**   
Fader工具可实现图表淡入淡出的动画效果。  
**Rectangle 工具**   
Rectangle 工具是一个可拖动及调整大小的注释工具。  
**Selector 工具**   
Selector工具能使终端用户点击或拖拽图表的数据序列，坐标轴，图例，标题等元素。  
**Transpose Series 工具**   
Transpose Series 工具可用来交换多个数据序列的行和列。   
**Data Table 工具**   
Data Table 工具可显示一个类似于Excel有横轴和纵轴的表格。  
**Banner 工具**   
Banner工具可制作一个滚动且闪烁的文字动画。  
**Magnify 工具**   
Magnify 工具可用来放大图表的某一部分。  
  
在设计时，这些工具可通过点击图表编辑器中Tool标签下的 Add...按钮进行添加  
  
  
在运行时只需要几行简单的代码就可以进行添加。

**3D Grid Transpose 工具**

3D Grid Transpose 工具可通过交换X与Z轴使基于网格的数据序列旋转90度。   
  
**设计时**  
使用Grid3D Series 的组合框来选择要旋转的数据序列。  
  
**运行时**在运行时，要添加一个3D Grid Transpose 工具是非常简单的操作，可参考以下范例：

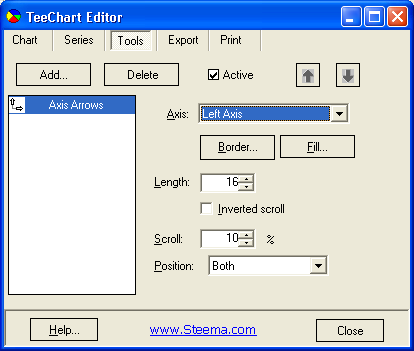
[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        Surface surface1 = new Surface(tChart1.Chart);   
        GridTranspose gridTranspose1 = new GridTranspose(tChart1.Chart);   
        gridTranspose1.Series = surface1;   
        surface1.FillSampleValues(20);   
   
   
private void button1\_Click(object sender, System.EventArgs e)   
        ((GridTranspose)tChart1.Tools[0]).Transpose();   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Surface1 As New Steema.TeeChart.Styles.Surface(TChart1.Chart)   
        Dim GridTranspose1 As New Steema.TeeChart.Tools.GridTranspose(TChart1.Chart)   
        Surface1.FillSampleValues(20)   
        GridTranspose1.Series = Surface1   
End Sub   
   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        CType(TChart1.Tools(0), Steema.TeeChart.Tools.GridTranspose).Transpose()   
End Sub

**Annotation 工具**

Annotation 工具可以在图表上显示文本。  
  
**设计时**  
  
  
Annotation 工具的Text及其文本框都是可以编辑的。Options标签可让您选择想进行设定的文本，并通过默认或自定义的值来设定文本位置。Format标签可对文本框进行编辑，同时修改文本框的颜色，框架，样式，倾斜度，大小，圆度及透明度。Text标签可设定文本的字体，颜色，字形及间距，另外也可以设定文本阴影区的颜色及大小。Gradient标签可编辑文本框的渐变属性。Shadow标签可编辑文本框的阴影区属性，而Beve标签则是设定注释工具的斜面属性。  
  
**运行时**   
在运行时添加一个Annotation 工具是非常简单的操作，可参考以下范例：

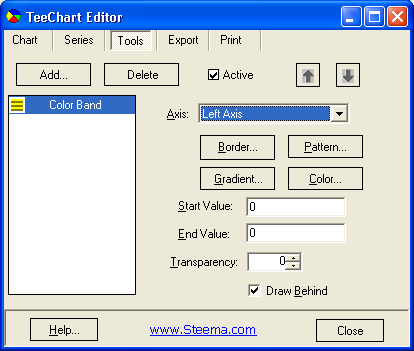
[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        Bar bar1 = new Bar(tChart1.Chart);   
        Steema.TeeChart.Tools.Annotation annotation1 = new Steema.TeeChart.Tools.Annotation(tChart1.Chart);   
        annotation1.Active = true;   
        annotation1.Position = AnnotationPositions.LeftTop;   
        annotation1.Text = "My Chart Text";   
   
        bar1.FillSampleValues(20);   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Dim Annotation1 As New Steema.TeeChart.Tools.Annotation(TChart1.Chart)    
        Annotation1.Active = True   
        Annotation1.Position = Steema.TeeChart.AnnotationPositions.LeftTop   
        Annotation1.Text = "My Chart Text"   
   
        Bar1.FillSampleValues(20)   
End Sub

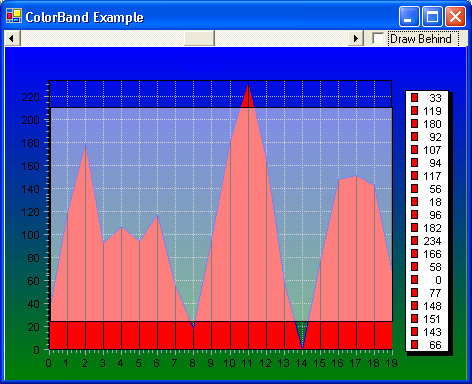
**Axis Arrows 工具**

Axis Arrows 工具在坐标轴的始端与末端显示可编辑的箭头。当点击这些箭头时，可滚动图表的坐标轴。  
**设计时**  
  
一旦您添加一个Arrows工具，那么许多选项卡就都会打开。Axis可以选择与Axis工具相关联的坐标轴（上，下，左，右或自定义）。您可以添加多个Axis工具，将他们分别与不同的坐标轴相关联。Border (Pen Editor)可编辑绘制箭头形状的画笔样式、颜色、宽度，末端，透明度及可见性，而Fill (Hatch Brush Editor)可编辑箭头的主要部分，包括它的颜色，透明度，可见性，样式填充，渐变填充或图像填充。Length描述了以像素计算箭头的长度；Position可设置是否在轴的始端，末端或是两端绘制箭头； Scroll可用来设置点击箭头时可滚动的坐标轴在轴长上所占的比例，Inverted Scroll则是设定图表在箭头的哪个方向上滚动。  
  
**运行时**   
在运行时添加一个Axis Arrows 工具是非常简单的操作，可参考以下范例：

[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        Bar bar1 = new Bar(tChart1.Chart);   
        AxisArrow axisArrow1 = new AxisArrow(tChart1.Chart);   
        axisArrow1.Active = true;   
        axisArrow1.Axis = tChart1.Axes.Left;   
   
        bar1.FillSampleValues(20);   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Dim AxisArrow1 As New Steema.TeeChart.AxisArrow(TChart1.Chart)   
        AxisArrow1.Active = True   
        AxisArrow1.Axis = TChart1.Axes.Left   
   
        Bar1.FillSampleValues(20)   
End Sub

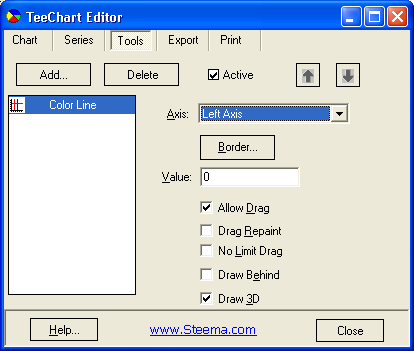
**Color Band 工具**

Color Band 工具可在图表坐标轴的内部区域添加可编辑的颜色带。可对这些颜色带设置它们的横纵坐标值。  
  
**设计时**  
只要将Color Band 工具添加在您的图表中，就会有很多相关可编辑的选项卡。Axis可以选择您想与 Color Band相关联的坐标轴（上，下，左，右或自定义）。Border (Pen 编辑器)可编辑箭头形状的画笔样式、颜色、宽度，末端，透明度及可见性，而Fill (Hatch Brush 编辑器)可设定箭头的主要部分，包括它的颜色，透明度，可见性，样式填充，渐变填充或图像填充。Gradient用来定义Color Band Gradient的可见性，方向及始端、中间和尾端的颜色而当没有渐变或图像填充时，Color可定义它的颜色。Start value用来设定Color Band 工具坐标轴的起始值，而End value设定终止值。Transparency是一个1到100的值，它可用来设定Color Band 的透明度，而Draw Behind 可设定在图表数据序列后面可用/禁用绘制Color Band。

**运行时**   
在运行时添加一个Color Band 工具是非常简单的操作，可参考以下范例：  
范例  
Color Band 是非常有用的工具，它可以在您的数据序列中突出显示一些特定的设置。在这个范例中我们使用一个Color Band工具，它在两个指定的动态Y值间延伸，同时可通过Scroll Bar来改变它的透明度。可在您的TeeChart数据序列的前面或后面来绘制Color Bands。  
  


[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        AreaSeries area1 = new AreaSeries(tChart1.Chart);   
        ColorBand colorband1 = new ColorBand(tChart1.Chart);   
        tChart1.Aspect.View3D = false;   
        tChart1.Panel.Gradient.Visible = true;   
        tChart1.Panel.Gradient.StartColor = Color.Blue;   
        tChart1.Panel.Gradient.MiddleColor = Color.Gray;   
        tChart1.Panel.Gradient.EndColor = Color.Green;   
   
        area1.LinePen.Color = Color.Blue;   
        area1.FillSampleValues(20);   
   
        double offSet = area1.YValues.Maximum \* 0.1;   
        colorband1.Active = true;   
        colorband1.Axis = tChart1.Axes.Left;   
        colorband1.Transparency = 50;   
        colorband1.Start = area1.YValues.Minimum + offSet;   
        colorband1.End = area1.YValues.Maximum - offSet;   
        colorband1.DrawBehind = false;   
        hScrollBar1.Value = 50;   
   
   
private void hScrollBar1\_Scroll(object sender, System.Windows.Forms.ScrollEventArgs e)   
        ((ColorBand)tChart1.Tools[0]).Transparency = e.NewValue;   
       
   
private void checkBox1\_CheckedChanged(object sender, System.EventArgs e)   
        ((ColorBand)tChart1.Tools[0]).DrawBehind = !((ColorBand)tChart1.Tools[0]).DrawBehind;   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Area1 As New Steema.TeeChart.Styles.Area(TChart1.Chart)   
        Dim Colorband1 As New Steema.TeeChart.ColorBand(TChart1.Chart)   
        TChart1.Aspect.View3D = False   
        TChart1.Panel.Gradient.Visible = True   
        TChart1.Panel.Gradient.StartColor = Color.Blue   
        TChart1.Panel.Gradient.MiddleColor = Color.Gray   
        TChart1.Panel.Gradient.EndColor = Color.Green   
   
        Area1.LinePen.Color = Color.Blue   
        Area1.FillSampleValues(20)   
   
        Dim OffSet As Double = Area1.YValues.Maximum \* 0.1   
        Colorband1.Active = True   
        Colorband1.Axis = TChart1.Axes.Left   
        Colorband1.Transparency = 50   
        Colorband1.Start = Area1.YValues.Minimum + OffSet   
        Colorband1.End = Area1.YValues.Maximum - OffSet   
        Colorband1.DrawBehind = False   
        HScrollBar1.Value = 50   
End Sub   
   
Private Sub HScrollBar1\_Scroll(ByVal sender As Object, ByVal e As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll   
        CType(TChart1.Tools(0), Steema.TeeChart.ColorBand).Transparency = e.NewValue   
End Sub   
   
Private Sub CheckBox1\_CheckedChanged(ByVal sender As Object, ByVal e As System.EventArgs) Handles CheckBox1.CheckedChanged   
        CType(TChart1.Tools(0), Steema.TeeChart.ColorBand).DrawBehind = Not CType(TChart1.Tools(0), Steema.TeeChart.ColorBand).DrawBehind   
End Sub

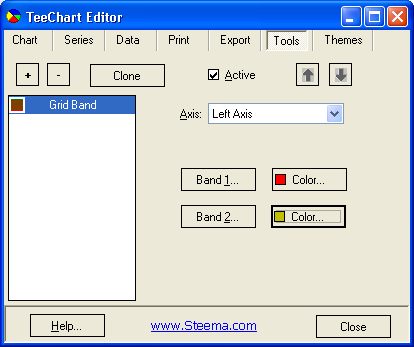
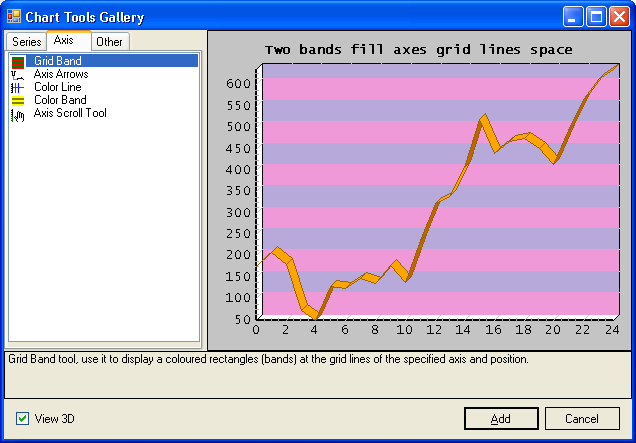
**Color Line 工具**

Color Line 工具可在图表中添加彩色的直线，并设置它们的横纵坐标值。  
  
**设计时**  
  
  
ColorLines可在特定的数据序列Value上与特定坐标轴相关联。Border (Pen Editor)可用来编辑画笔的样式、颜色、宽度，末端，透明度及可见性，Allow Drag可实现拖拽能，Drag Repaint可在拖拽ColorLine工具时重绘图表，No Drag Limit可允许ColorLine在图表的坐标轴外进行拖拽，Draw Behind可在图表Series背面绘制ColorLine，然而Draw3D可在3D中绘制ColorLine。

**运行时**   
在运行时添加一个Color Line工具是非常简单的操作，可参考以下范例：

[C#]   
Bar bar1 = new Bar(tChart1.Chart);   
ColorLine colorLine1 = new ColorLine(tChart1.Chart);   
   
bar1.FillSampleValues(20);   
colorLine1.Active = true;   
colorLine1.AllowDrag = true;   
colorLine1.Axis = tChart1.Axes.Left;   
colorLine1.Pen.Color = Color.Blue;   
colorLine1.Value = bar1.YValues.Maximum / 2;   
   
[VB.Net]   
Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
Dim ColorLine1 As New Steema.TeeChart.ColorLine(TChart1.Chart)   
   
Bar1.FillSampleValues(20)   
ColorLine1.Active = True   
ColorLine1.AllowDrag = True   
ColorLine1.Axis = TChart1.Axes.Left   
ColorLine1.Pen.Color = Color.Blue   
ColorLine1.Value = Bar1.YValues.Maximum / 2

**Grid Band 工具**

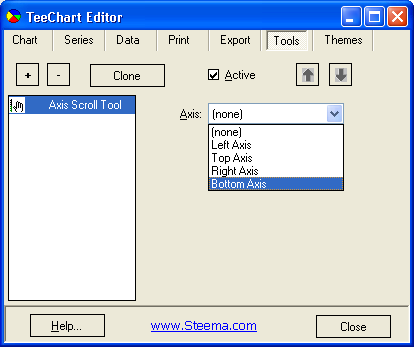
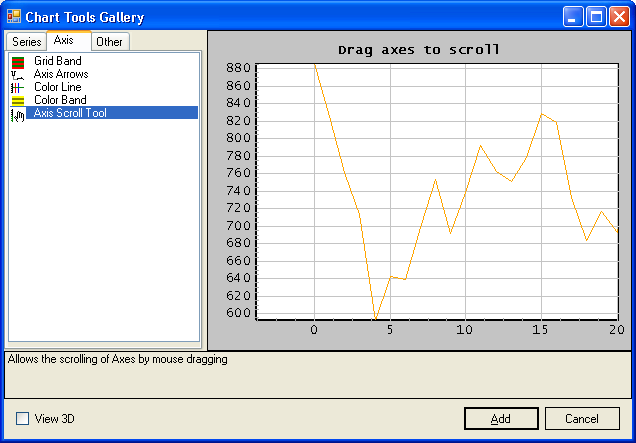
Grid Band 工具可在网格线特定的轴及位置上显示彩色矩形（带）。  
  
**设计时**   
  
  
  
使用Axis组合框选择您想与该工具相关联的坐标轴。每个带的格式刷和颜色特性都可使用Band 和 Color按钮来分别指定。  
  
  
  
**运行时**   
在运行时添加一个 Grid Band工具是非常简单的操作，可参考以下范例：  
  
[C#]

GridBand tool = new GridBand(tChart1.Chart);   
Line line = new Line(tChart1.Chart);   
line.FillSampleValues();   
   
tool.Axis = tChart1.Axes.Left;   
tool.Band1.Color = Color.Red;   
tool.Band2.Color = Color.White;

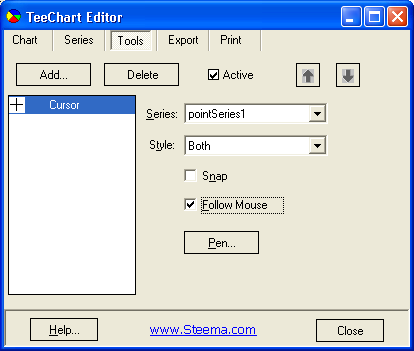
[VB.Net]

Dim tool As GridBand = New GridBand(TChart1.Chart)   
Dim line As Line = New Line(TChart1.Chart)   
line.FillSampleValues()   
   
tool.Axis = TChart1.Axes.Left   
tool.Band1.Color = Color.Red   
tool.Band2.Color = Color.White

**Axis Scroll 工具**

Axis Scroll 工具可通过鼠标拖拽来实现图表坐标轴的滚动。   
  
**设计时**   
  
  
  
使用Axis组合框可选择您想与该工具相关联的坐标轴。  
  
  
  
**运行时**   
在运行时添加一个Axis Scroll工具是非常简单的操作，可参考以下范例：  
  
[C#]   
AxisScroll tool = new AxisScroll(tChart1.Chart);   
Line line = new Line(tChart1.Chart);   
line.FillSampleValues();   
  
tool.Axis = tChart1.Axes.Left;   
  
[VB.Net]   
Dim tool As AxisScroll =  New AxisScroll(tChart1.Chart)    
Dim line As Line =  New Line(tChart1.Chart)    
line.FillSampleValues()   
    
tool.Axis = tChart1.Axes.Left

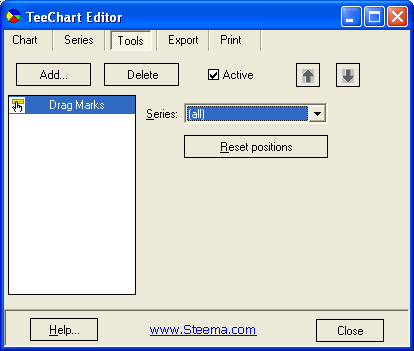
**Cursor工具**

Cursor 工具可在图表中添加光标，其可与个别的数据序列或与整个图表相关联。光标可以是横向的，纵向的或两者兼有（十字的）。  
  
  
**设计时**  
  
  
一旦添加了Cursor工具，不一定非要将它与图表Series相关联，您可以在Style中编辑光标的样式，它可以是横向的，纵向的或两者兼有。如果Cursor工具与数据序列相关联，那么当光标移动时，它可以将数据序列的Y值设为Snap。Follow Mouse可让Cursor工具随鼠标进行移动，而 Pen (Pen Editor) 可编辑Cursor工具的样式，颜色，宽度，末端，透明度及可见度。  
  
**运行时**

在运行时添加一个Cursor工具是非常简单的操作，可参考以下范例：

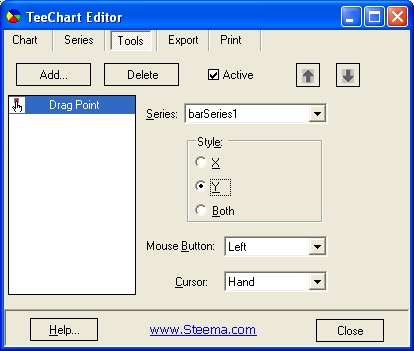
[C#]   
Points pointSeries1 = new Points(tChart1.Chart);   
CursorTool cursorTool1 = new CursorTool(tChart1.Chart);   
   
pointSeries1.FillSampleValues(20);   
cursorTool1.Active = true;   
cursorTool1.FollowMouse = true;   
cursorTool1.Series = pointSeries1;   
cursorTool1.Style = CursorToolStyles.Both;   
   
[VB.Net]   
Dim PointSeries1 As New Steema.TeeChart.Styles.Points(TChart1.Chart)   
Dim CursorTool1 As New Steema.TeeChart.Tools.CursorTool(TChart1.Chart)   
   
PointSeries1.FillSampleValues(20)   
CursorTool1.Active = True   
CursorTool1.FollowMouse = True   
CursorTool1.Series = PointSeries1   
CursorTool1.Style = Steema.TeeChart.Tools.CursorToolStyles.Both

**Drag Marks 工具**

Drag Marks 工具可对已选定的Series（数据序列）标记鼠标拖拽。  
  
**设计时**  
  
  
Drag Marks工具可用来设置特定的Series，或通过调用来断开与所有图表数据序列的关联。Reset Positions可让所有已移动的标记返回到它们原始的默认位置。  
  
**运行时**   
在运行时添加一个Drag Marks工具是非常简单的操作，可参考以下范例：

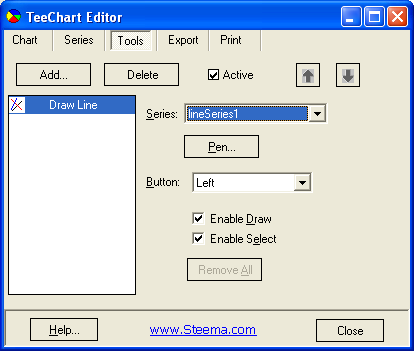
[C#]   
Bar bar1 = new Bar(tChart1.Chart);   
DragMarks dragMarks1 = new DragMarks(tChart1.Chart);   
   
bar1.FillSampleValues(20);   
dragMarks1.Active = true;   
dragMarks1.Series = bar1;   
   
[VB.Net]   
Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
Dim DragMarks1 As New Steema.TeeChart.DragMarks(TChart1.Chart)   
   
Bar1.FillSampleValues(20)   
DragMarks1.Active = True   
DragMarks1.Series = Bar1

**Drag Point 工具**

Drag Point 工具可对已选定的Series（数据序列）标记鼠标拖动。  
  
**设计时**  
  
  
Drag Point 工具可用来设置特定Series，或通过调用来断开与所有图表数据序列的关联。Style可使Drag Point工具在X方向，Y方向或两个方向上进行拖拽。Mouse Button可选择活动的鼠标按键而Cursor可选择当在Series点上进行拖拽时光标的样式。  
  
**运行时**   
在运行时添加一个Drag  Point工具是非常简单的操作，可参考以下范例：

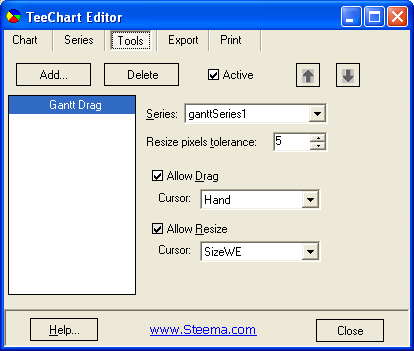
[C#]   
Bar bar1 = new Bar(tChart1.Chart);   
DragPoint dragPoint1 = new DragPoint(tChart1.Chart);   
   
bar1.FillSampleValues(10);   
dragPoint1.Active = true;   
dragPoint1.Series = bar1;   
dragPoint1.Style = DragPointStyles.Y;   
   
 [VB.Net]   
Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
Dim DragPoint1 As New Steema.TeeChart.DragPoint(TChart1.Chart)   
   
Bar1.FillSampleValues(10)   
DragPoint1.Active = True   
DragPoint1.Series = Bar1   
DragPoint1.Style = Steema.TeeChart.DragPointStyles.Y

**Draw Line 工具**

在图表中，Draw Line 工具可完全自定义使用鼠标完成直线的绘制，选择，拖动及删除等操作。  
  
  
**设计时**  
  
  
Draw Line工具可用来设置特定Series，或可通过Series 组合框来断开与所有图表数据序列的关联。Button组合框可设置Draw Line工具的特定鼠标按钮。Pen (Pen Editor) 允许您对Draw Lines的风格，颜色，宽度，尾端，透明度及可见度进行编辑，而Enable Drawing和Enable Select可对直线的绘制及选择进行可用/禁用操作。  
  
**运行时**   
在运行时添加一个Draw Line工具是非常简单的操作，可参考以下范例：

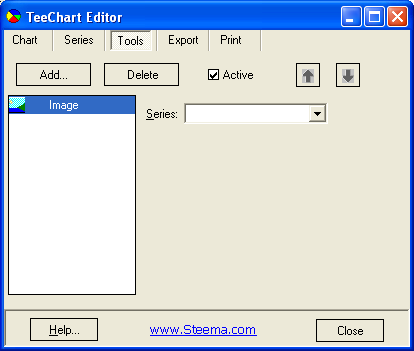
[C#]   
Bar bar1 = new Bar(tChart1.Chart);   
DrawLine drawLine1 = new DrawLine(tChart1.Chart);   
   
bar1.FillSampleValues(20);   
drawLine1.Series = bar1;   
drawLine1.Button = MouseButtons.Left;   
drawLine1.EnableDraw = true;   
drawLine1.EnableSelect = true;   
drawLine1.Pen.Color = Color.AliceBlue;   
   
[VB.Net]   
Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
Dim DrawLine1 As New Steema.TeeChart.DrawLine(TChart1.Chart)   
   
Bar1.FillSampleValues(20)   
DrawLine1.Series = Bar1   
DrawLine1.Button = MouseButtons.Left   
DrawLine1.EnableDraw = True   
DrawLine1.EnableSelect = True   
DrawLine1.Pen.Color = Color.AliceBlue

**Gantt Drag 工具**

Gantt Drag 工具能拖拽和/或调整Gantt Series的滑动条。  
**设计时**  
  
  
Gantt Drag 已和一个特定的Gantt Series相关联。Resize Pixel Tolerance可指定按像素拖拽的精确度，Allow Drag表示可对其进行拖拽，而Cursor可设定在拖拽时光标的显示样式。 Allow Resize可使Gantt Bar恢复到所要求的尺寸，而 Cursor可设定在恢复时光标的显示样式。  
  
**运行时**   
在运行时添加一个 Image工具是非常简单的操作，可参考以下范例：

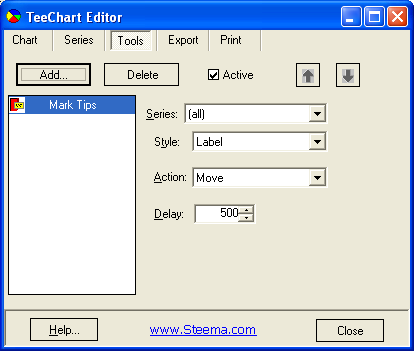
[C#]   
GanttSeries ganttSeries1 = new GanttSeries(tChart1.Chart);   
GanttTool ganttTool1 = new GanttTool(tChart1.Chart);   
   
ganttSeries1.FillSampleValues(10);   
ganttTool1.AllowResize = false;   
ganttTool1.AllowDrag = true;   
ganttTool1.MinPixels = 3;   
ganttTool1.Series = ganttSeries1;   
ganttTool1.Active = true;   
   
tChart1.Legend.Visible = false;   
   
[VB.Net]   
Dim GanttSeries1 As New Steema.TeeChart.GanttSeries(TChart1.Chart)   
Dim GanttTool1 As New Steema.TeeChart.GanttTool(TChart1.Chart)   
   
GanttSeries1.FillSampleValues(10)   
GanttTool1.AllowResize = False   
GanttTool1.AllowDrag = True   
GanttTool1.MinPixels = 3   
GanttTool1.Series = GanttSeries1   
GanttTool1.Active = True   
   
TChart1.Legend.Visible = False

**Image 工具**

Image 工具可在特定图表的Series背后绘制图片（bitmap, jpeg, gif, png, pcx）。  
  
  
**设计时**  
  
  
Image 工具可与特定的Series相关联，或者可断开与任何图表数据序列的关联。使用Image工具可浏览或清除所有格式的图像。  
  
**运行时**在运行时添加一个 Image工具是非常简单的操作，可参考以下范例：

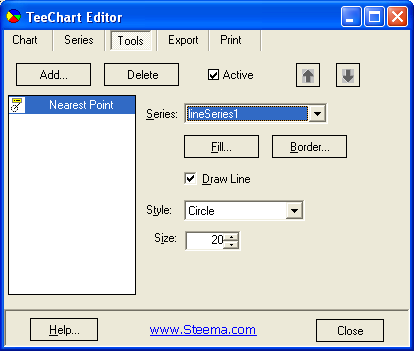
[C#]   
Bar bar1 = new Bar(tChart1.Chart);   
ChartImage chartImage1 = new ChartImage(tChart1.Chart);   
string imagePath = @"C:.bmp";   
Bitmap myBitMap = new Bitmap(imagePath);   
   
bar1.FillSampleValues(20);   
chartImage1.Active = true;   
chartImage1.Image = myBitMap;   
   
[VB.Net]   
Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
Dim ChartImage1 As New Steema.TeeChart.ChartImage(TChart1.Chart)   
Dim ImagePath As String = "C:.bmp"   
Dim MyBitMap As New Bitmap(ImagePath)   
   
Bar1.FillSampleValues(20)   
ChartImage1.Active = True   
ChartImage1.Image = MyBitMap

**Mark Tips 工具**

Mark Tips 工具是当鼠标滑过（或点击）一个Series点时，可用来显示默认的Hint窗口。  
  
**设计时**  
  
  
Mark Tips工具可与特定的Series相关联，或者可断开与任何图表数据序列的默认关联。Style可定义在Mark Tips中显示数据序列的数据类型，而Mouse Action可设置在鼠标点击或移动时是否显示Mark Tips**。** Delay可定义在 Mark Tips显示的时间间隔（毫秒级）。  
  
**运行时**   
在运行时添加一个Mark Tips 工具是非常简单的操作，可参考以下范例：

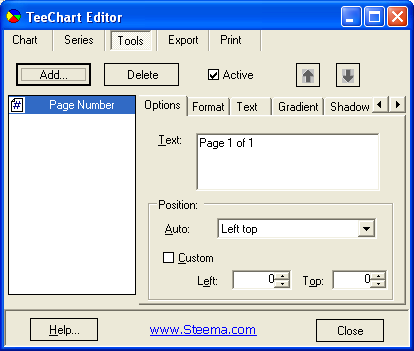
[C#]   
PointSeries pointSeries1 = new PointSeries(tChart1.Chart);   
MarksTip marksTip1 = new MarksTip(tChart1.Chart);   
        tab   
pointSeries1.FillSampleValues(20);   
marksTip1.Active = true;   
marksTip1.MouseDelay = 500;   
marksTip1.MouseAction = MarksTipMouseAction.Move;   
marksTip1.Style = MarksStyles.XY;   
   
[VB.Net]   
Dim PointSeries1 As New Steema.TeeChart.PointSeries(TChart1.Chart)   
Dim MarksTip1 As New Steema.TeeChart.MarksTip(TChart1.Chart)   
   
PointSeries1.FillSampleValues(20)   
MarksTip1.Active = True   
MarksTip1.MouseDelay = 500   
 MarksTip1.MouseAction = Steema.TeeChart.MarksTipMouseAction.Move   
 MarksTip1.Style = Steema.TeeChart.MarksStyles.XY

**Nearest Point 工具**

Nearest Point 工具可将标记移到最接近光标的点。该工具可与Series相关联或应用于整个图表。  
  
**设计时**   
  
  
Nearest Point工具需要与特定的Chart Series相关联。它的外观可使用Fill (Hatch Brush Editor)来定义Nearest Point的颜色，透明度，可见度，延时，样式填充，渐变填充及图像填充，Border (Pen Editor)可定义Nearest Point 工具的绘图笔样式，颜色，宽度，末端，透明度及可见度。Draw Line能可用/禁用在光标位置与Nearest Point 工具的图形之间绘制连接线， Size可定义形状的大小，而Style可定义所使用形状的类型。  
  
**运行时**   
在运行时添加一个Nearest Point工具是非常简单的操作，可参考以下范例：

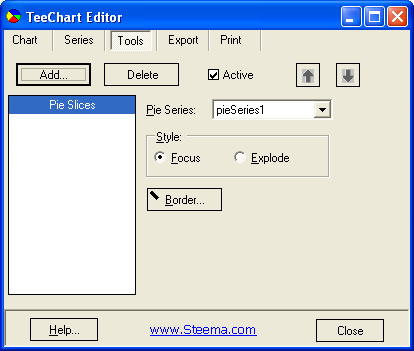
[C#]   
Line line1 = new Line(tChart1.Chart);   
NearestPoint nearestPoint1 = new NearestPoint(tChart1.Chart);   
   
line1.FillSampleValues(20);   
nearestPoint1.Series = line1;   
nearestPoint1.Pen.Color = Color.Blue;   
nearestPoint1.Size = 25;   
nearestPoint1.Style = NearestPointStyles.Diamond;   
   
[VB.Net]   
Dim Line1 As New Steema.TeeChart.Styles.Line(TChart1.Chart)   
Dim NearestPoint1 As New Steema.TeeChart.NearestPoint(TChart1.Chart)   
   
Line1.FillSampleValues(20)   
NearestPoint1.Series = Line1   
NearestPoint1.Pen.Color = Color.Blue   
NearestPoint1.Size = 25   
NearestPoint1.Style = Steema.TeeChart.NearestPointStyles.Diamond

**Page Number 工具**

Page Number 工具可显示一个包含页码选择系统的可编辑文本框。  
  
**设计时**  
  
  
Page Number 工具的Text和它的文本框都是完全可编辑的。Options标签允许您选择想设置的文本，并通过默认或自定义的值来设定文本位置。Format标签可编辑文本框，允许您改变它的颜色（Color Editor），框架（Pen Editor），样式（HatchBrush Editor），圆度及透明度。Text标签可设定文本字体（Font Editor），填充（Hatch Brush Editor），同时也可设定文本阴影的颜色(Color Editor)和样式 (HatchBrush Editor) 。Gradient标签可配置文本框渐变的属性。Shadow标签可编辑文本框阴影的属性而Bevels选项卡则可配置文本框斜角的属性。  
  
**运行时**   
在运行时添加一个Page Number工具是非常简单的操作，可参考以下范例：

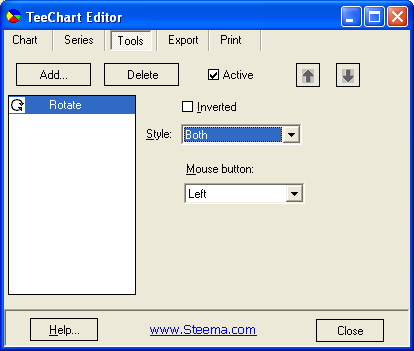
[C#]   
private void Form1\_Load(object sender, System.EventArgs e)   
        Bar bar1 = new Bar(tChart1.Chart);   
        PageNumber pageNumber1 = new PageNumber(tChart1.Chart);   
   
        bar1.FillSampleValues(200);   
        tChart1.Page.MaxPointsPerPage = 10;       
   
   
private void button1\_Click(object sender, System.EventArgs e)   
        tChart1.Page.Previous();   
   
   
private void button2\_Click(object sender, System.EventArgs e)   
        tChart1.Page.Next();   
   
   
[VB.Net]   
Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load   
        Dim Bar1 As New Steema.TeeChart.Styles.Bar(TChart1.Chart)   
        Dim PageNumber1 As New Steema.TeeChart.PageNumber(TChart1.Chart)   
   
        Bar1.FillSampleValues(200)   
        TChart1.Page.MaxPointsPerPage = 10   
End Sub   
   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
        TChart1.Page.Previous()   
End Sub   
   
Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click   
        TChart1.Page.Next()   
End Sub

**Pie Slices 工具**

不管是在扇形区域周围绘制边界或是在饼状Series外部进行延伸，Pie Slices工具强调的是鼠标下的扇形区。  
  
  
**设计时**  
  
  
Pie Slices工具已与一个特定的Pie Series相关联。Style设定Pie Slice的边界是否突显（可通过Border按钮（Pen Editor）来进行编辑），或是否可在Pie Series以外的区域进行延伸。  
  
**运行时**   
在运行时添加一个Pie Slices工具是非常简单的操作，可参考以下范例：

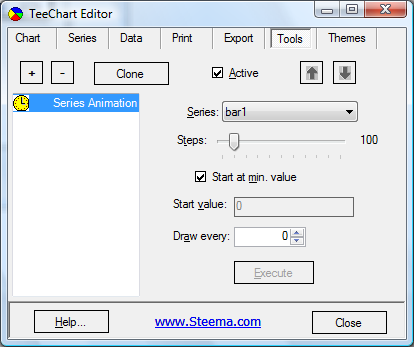
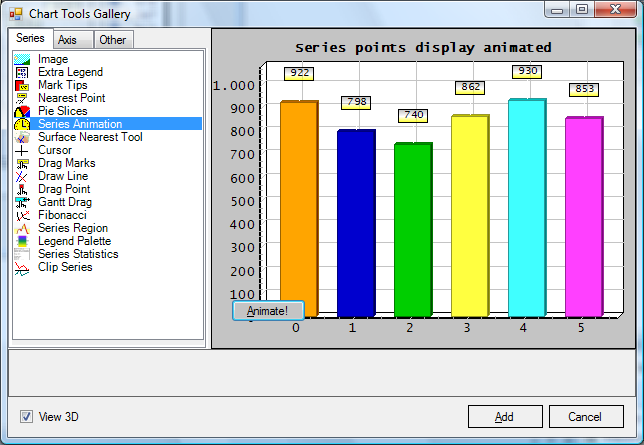
[C#]   
PieSeries pieSeries1 = new PieSeries(tChart1.Chart);   
PieTool pieTool1 = new PieTool(tChart1.Chart);   
   
pieSeries1.FillSampleValues(10);   
pieTool1.Series = pieSeries1;   
pieTool1.Style = PieToolStyle.Explode;   
   
[VB.Net]   
Dim PieSeries1 As New Steema.TeeChart.PieSeries(TChart1.Chart)   
Dim PieTool1 As New Steema.TeeChart.PieTool(TChart1.Chart)   
   
PieSeries1.FillSampleValues(10)   
PieTool1.Series = PieSeries1   
PieTool1.Style = Steema.TeeChart.PieToolStyle.Explode

**Rotate 工具**

在图表上拖拽鼠标时，Rotate工具可改变图表的立体度及旋度。  
  
**设计时**  
  
  
Rotat是可以编辑的，它可以使用Style 组合框来实现旋转（沿图表的纵轴旋转），抬高（沿横轴旋转）或同时实现两种情况。Mouse Button 可用来旋转图表也可以用来选择图表。  
  
**运行时**   
在运行时添加一个Rotate 工具是非常简单的操作，可参考以下范例：

[C#]   
Line line1 = new Line(tChart1.Chart);   
Rotate rotate1 = new Rotate(tChart1.Chart);   
   
line1.FillSampleValues(20);   
rotate1.Button = MouseButtons.Right;   
rotate1.Style = RotateStyles.Elevation;   
   
[VB.Net]   
Dim Line1 As New Steema.TeeChart.Styles.Line(TChart1.Chart)   
Dim Rotate1 As New Steema.TeeChart.Rotate(TChart1.Chart)   
   
Line1.FillSampleValues(20)   
Rotate1.Button = MouseButtons.Right   
Rotate1.Style = Steema.TeeChart.RotateStyles.Elevation

**Series Animation 工具**

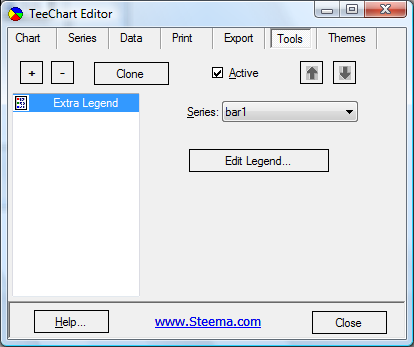
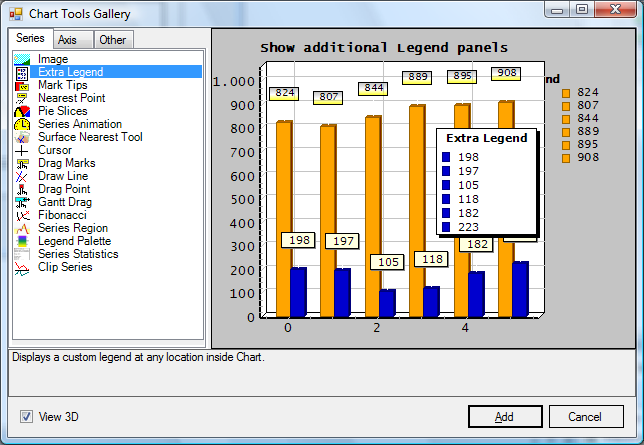
Series Animation 工具可生成一个Series Points（数据序列点）的渐增式动画。  
  
**设计时**   
  
  
  
为了使Series Animation工具生效，它已和数据序列相关联。动画的速度是可以通过逐步跟踪条和绘图的每个属性来控制的，而动画的起始点可通过数组序列的最小值或文本框的起始值来设定。  
  
  
  
**运行时**   
在运行时添加一个Series Animation工具是非常简单的操作，可参考以下范例：   
  
[C#]

private SeriesAnimation tool;   
private void InitializeChart()   
   
     Bar bar = new Bar(tChart1.Chart);   
     tool = new SeriesAnimation(tChart1.Chart);   
     tool.Series = bar;   
     bar.FillSampleValues();   
   
   
private void button1\_Click(object sender, EventArgs e)   
   
     tool.Execute();

[VB.Net]

Dim tool As SeriesAnimation   
   
Public Sub InitializeChart()   
 Dim bar As Bar = New Bar(TChart1.Chart)   
 tool = New SeriesAnimation(TChart1.Chart)   
 tool.Series = bar   
 bar.FillSampleValues()   
End Sub   
   
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
 tool.Execute()   
End Sub

**Extra Legend工具**

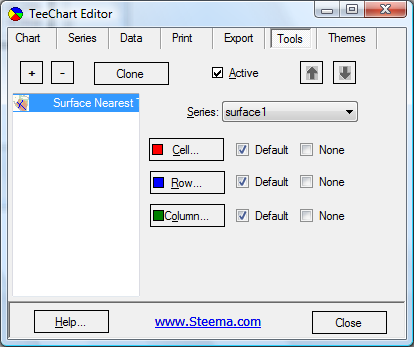
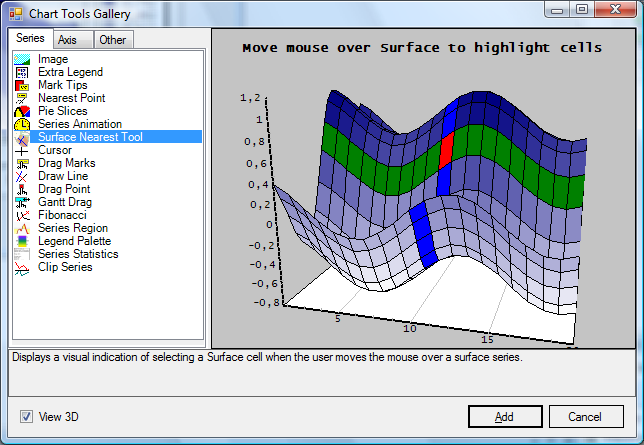
Extra Legend工具可在图表内的任意位置显示自定义的图例。  
  
**设计时**   
  
  
  
为了能显示Extra工具，它已和一个数据序列相关联。"Edit Legend"对话框可对图例进行编辑。  
  
  
  
**运行时**   
在运行时添加一个Extra工具是非常简单的操作，可参考以下范例：  
  
[C#]

Bar bar = new Bar(tChart1.Chart);   
ExtraLegend tool = new ExtraLegend(tChart1.Chart);   
tool.Series = bar;   
bar.FillSampleValues();

[VB.Net]

Dim bar As Bar = New Bar(TChart1.Chart)   
Dim tool As ExtraLegend = New ExtraLegend(TChart1.Chart)   
tool.Series = bar   
bar.FillSampleValues()

**Surface Nearest 工具**

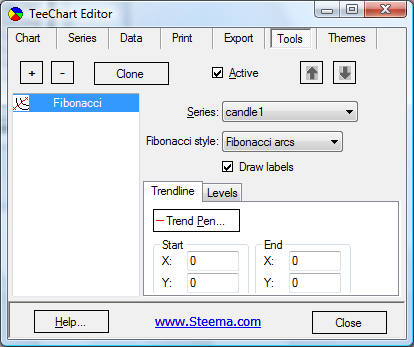
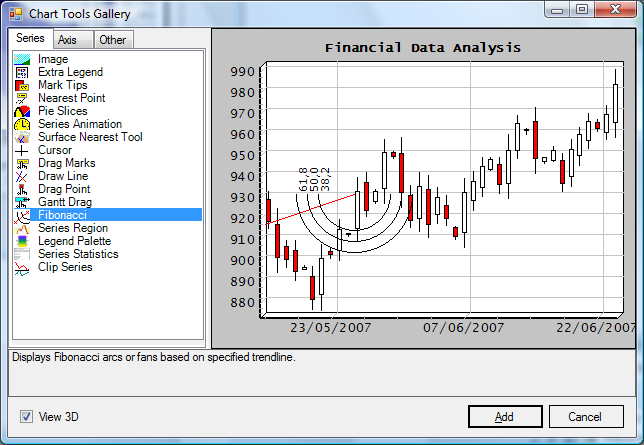
Surface Nearest工具可在图表内的任意位置显示自定义的图例。  
  
**设计时**   
  
  
  
 使用Series组合框可选择您想与之相关联的外观数组序列。通过使用相关的颜色编辑器可对单元格，行及列的颜色特性进行编辑。  
  
  
  
**运行时**   
在运行时添加一个Surface Nearest工具是非常简单的操作，可参考以下范例：   
  
[C#]

Surface surface = new Surface(tChart1.Chart);   
SurfaceNearestTool tool = new SurfaceNearestTool(tChart1.Chart);   
tool.Series = surface;   
surface.FillSampleValues();

[VB.Net]

Dim surface As Surface = New Surface(TChart1.Chart)   
Dim tool As SurfaceNearestTool = New SurfaceNearestTool(TChart1.Chart)   
tool.Series = surface   
surface.FillSampleValues()

**Fibonacci工具**

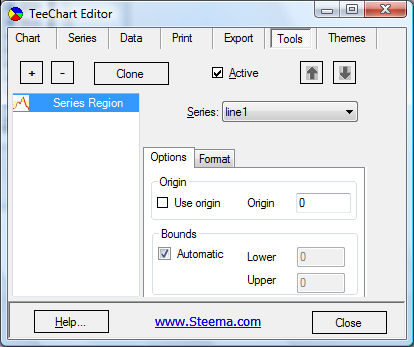
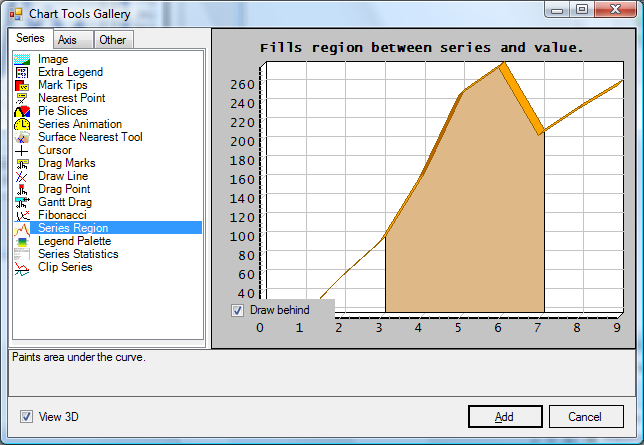
Fibonacci 工具显示Fibonacci弧形或基于特定趋势线的扇形。  
  
**设计时**   
  
  
  
使用Series组合框可选择您想与之相关联的数组序列。Fibonacci的绘图特性与它的算法变量都可以在上面的编辑器中进行编辑。  
  
  
  
**运行时**   
在运行时添加一个Fibonacci工具是非常简单的操作，可参考以下范例：   
  
[C#]

Candle candle = new Candle(tChart1.Chart);   
FibonacciTool tool = new FibonacciTool(tChart1.Chart);   
candle.FillSampleValues();   
tool.Series = candle;   
tool.StartX = candle.DateValues[0];   
tool.StartY = candle.CloseValues[0];   
tool.EndX = candle.DateValues[10];   
tool.EndY = candle.CloseValues[10];

[VB.Net]

Dim candle As Candle = New Candle(tChart1.Chart)   
Dim tool As FibonacciTool = New FibonacciTool(tChart1.Chart)   
candle.FillSampleValues   
tool.Series = candle   
tool.StartX = candle.DateValues(0)   
tool.StartY = candle.CloseValues(0)   
tool.EndX = candle.DateValues(10)   
tool.EndY = candle.CloseValues(10)

**Series Region工具**

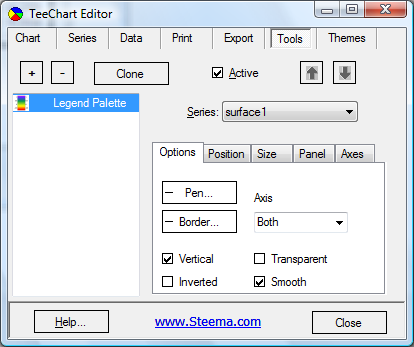
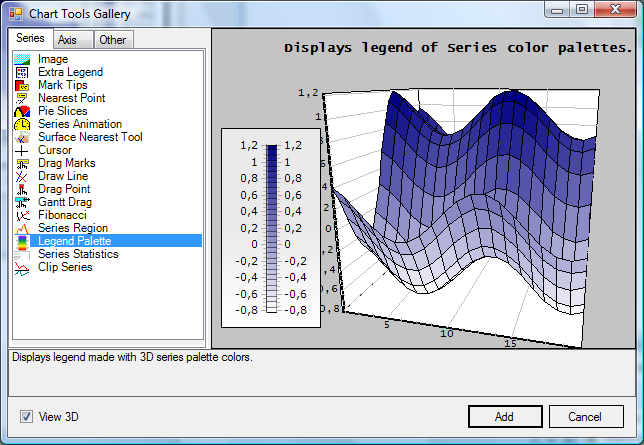
Series Region工具可在点或直线的数据序列下描绘一块区域。  
  
**设计时**   
  
  
  
使用Series组合框可选择您想与之相关联的3D数据序列。Origin可指定区域内y值（水平）的基值而Bounds可指定x值（垂直）的起始与结束点。使用格式标签可定义区域内画笔及格式刷的特性。  
  
  
  
**运行时**   
在运行时添加一个Series Region工具是非常简单的操作，可参考以下范例：   
  
[C#]

Line line = new Line(tChart1.Chart);   
SeriesRegionTool tool = new SeriesRegionTool(tChart1.Chart);   
line.FillSampleValues();   
tool.Series = line;

[VB.Net]

Dim line As Line = New Line(TChart1.Chart)   
Dim tool As SeriesRegionTool = New SeriesRegionTool(TChart1.Chart)   
tool.Series = line   
line.FillSampleValues()

**Legend Palette 工具**

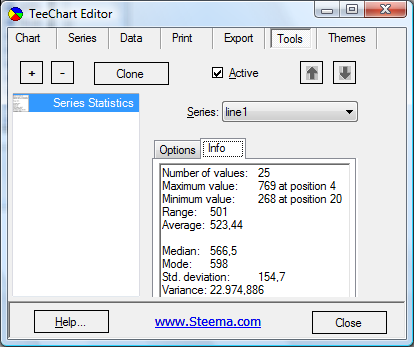
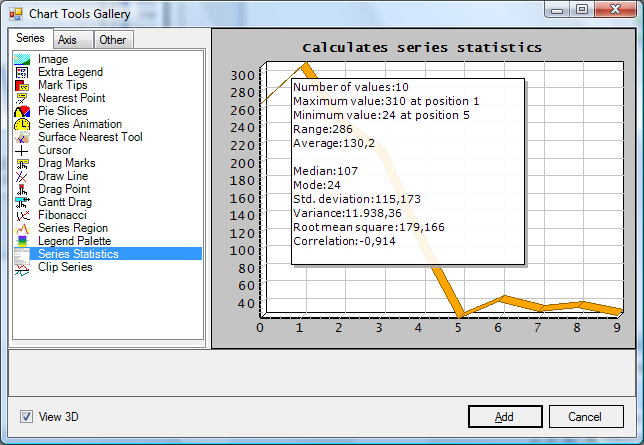
Legend Palette 工具显示由3D数据序列彩色调色板生成的图例。  
  
  
**设计时**   
  
  
  
使用Series组合框可选择您想与之相关联的3D数据序列。该工具的所有元素都是可编辑的。  
  
  
  
**运行时**   
在运行时添加一个Legend Palette工具是非常简单的操作，可参考以下范例：  
  
[C#]

Surface surface = new Surface(tChart1.Chart);   
LegendPalette tool = new LegendPalette(tChart1.Chart);   
surface.FillSampleValues();   
tool.Series = surface;

[VB.Net]

Dim surface As Surface = New Surface(TChart1.Chart)   
Dim tool As LegendPalette = New LegendPalette(TChart1.Chart)   
surface.FillSampleValues()   
tool.Series = surface

**Series Statistics 工具**

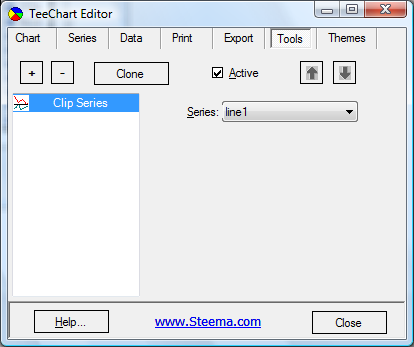
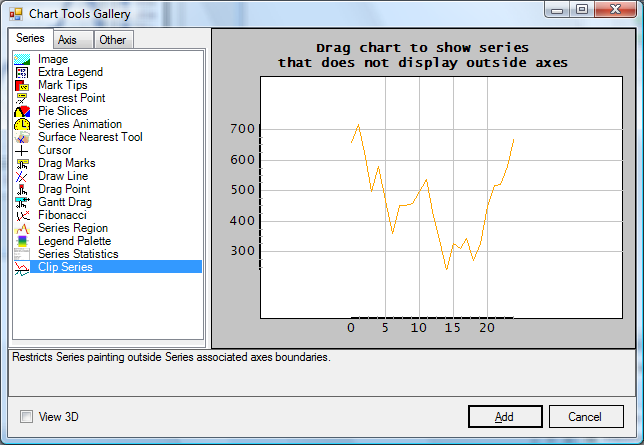
Series Statistics 工具可对给定的数据序列估算大量标准的统计量度。  
  
**设计时**   
  
  
  
使用Series 组合框可选择您想与工具相关联的数据序列。  
  
  
  
**运行时**   
在运行时添加一个Series Statistics 工具是非常简单的操作，可参考以下范例：  
  
[C#]

SeriesStats tool = new SeriesStats(tChart1.Chart);   
Line line = new Line(tChart1.Chart);   
line.FillSampleValues();   
tool.Series = line;   
   
Annotation at = new Annotation(tChart1.Chart);   
at.Shape.Transparency = 10;   
at.Left = 80;   
at.Top = 50;   
at.Text = tool.Statistics;

[VB.Net]

Dim line As Line = New Line(TChart1.Chart)   
Dim tool As SeriesStats = New SeriesStats(TChart1.Chart)   
line.FillSampleValues()   
tool.Series = line   
   
Dim at As Annotation = New Annotation(TChart1.Chart)   
at.Shape.Transparency = 10   
at.Left = 80   
at.Top = 50   
at.Text = tool.Statistics

**Clip Series 工具**

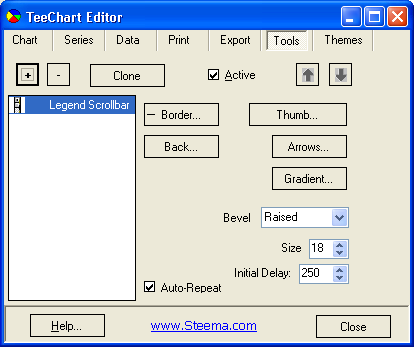
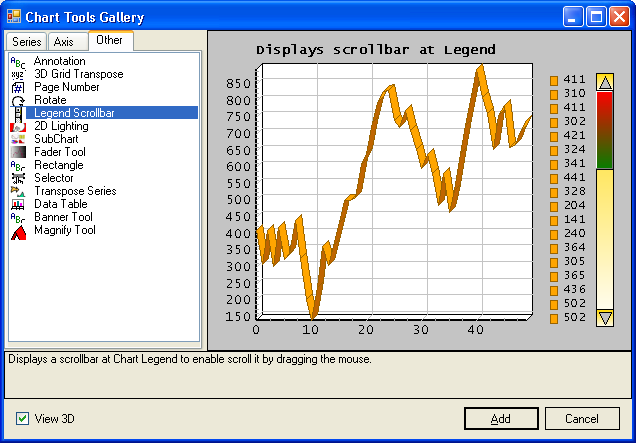
Clip Series工具限制与Series相关联的轴边界外Series（数据序列）的绘制。在使用自定义坐标轴时该工具是非常有用的。  
  
**设计时**   
  
  
  
使用Series组合框来选择您想要与工具相关联的数据序列。   
  
  
  
**运行时**   
在运行时添加一个Series Statistics工具是非常简单的操作，可参考以下范例：   
  
[C#]

Axis vert = new Axis(false, false, tChart1.Chart);   
Axis horiz = new Axis(true, false, tChart1.Chart);   
tChart1.Axes.Custom.Add(vert);   
tChart1.Axes.Custom.Add(horiz);   
   
Line line = new Line(tChart1.Chart);   
line.FillSampleValues();   
horiz.StartPosition = 20;   
horiz.EndPosition = 80;   
vert.StartPosition = 20;   
vert.EndPosition = 80;   
line.CustomHorizAxis = horiz;   
line.CustomVertAxis = vert;   
   
ClipSeries tool = new ClipSeries(tChart1.Chart);   
tool.Series = line;   
   
tChart1.Panel.MarginBottom = 10;   
tChart1.Panel.MarginLeft = 10;

[VB.Net]

Dim vert As Axis = New Axis(False, False, TChart1.Chart)   
Dim horiz As Axis = New Axis(True, False, TChart1.Chart)   
TChart1.Axes.Custom.Add(vert)   
TChart1.Axes.Custom.Add(horiz)   
   
Dim line As Line = New Line(TChart1.Chart)   
line.FillSampleValues()   
horiz.StartPosition = 20   
horiz.EndPosition = 80   
vert.StartPosition = 20   
vert.EndPosition = 80   
line.CustomHorizAxis = horiz   
line.CustomVertAxis = vert   
   
Dim tool As ClipSeries = New ClipSeries(TChart1.Chart)   
tool.Series = line   
   
TChart1.Panel.MarginBottom = 10   
TChart1.Panel.MarginLeft = 10

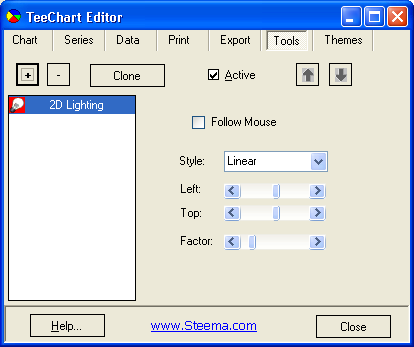
**Legend ScrollBar 工具**

Legend ScrollBar工具可通过鼠标的拖拽来滚动图例滚动条。  
  
**设计时**   
  
  
  
Legend ScrollBar编辑器中的不同按钮可用于编辑图像不同方面的内容，而initial delay 可设定工具点击与图表重绘间所需的时间，  
  
  
  
**运行时**   
在运行时添加一个ScrollBar工具是非常简单的操作，可参考以下范例：  
  
[C#]

Line line = new Line(tChart1.Chart);   
LegendScrollBar tool = new LegendScrollBar(tChart1.Chart);   
line.FillSampleValues();

[VB.Net]   
Dim line As Line =  New Line(TChart1.Chart)    
Dim tool As LegendScrollBar =  New LegendScrollBar(TChart1.Chart)    
line.FillSampleValues()

**2D Lighting工具**

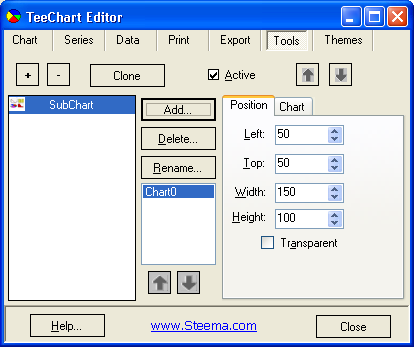
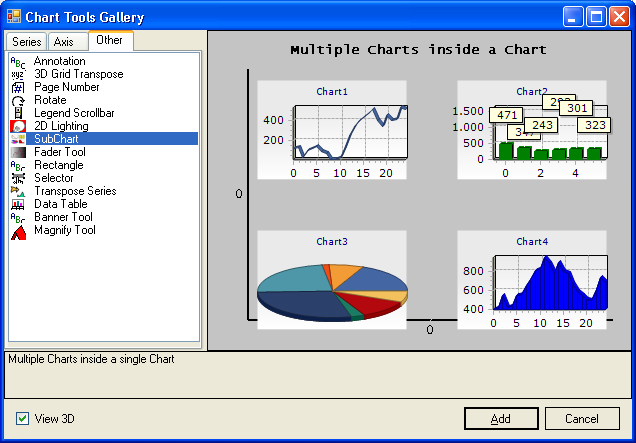
2D Lighting 工具可在图表Canvas上显示一个2维的“灯光”效果。  
**设计时**   
  
  
  
该照明工具（左上）的位置可以指定，或者您可以选择根据鼠标来使用工具。   
  
  
  
**运行时**   
在运行时添加一个2D Lighting工具是非常简单的操作，可参考以下范例：  
  
[C#]

Line line = new Line(tChart1.Chart);   
LightTool tool = new LightTool(tChart1.Chart);   
tool.FollowMouse = true; //slow in Debug build   
line.FillSampleValues();

[VB.Net]

Dim line As Line =  New Line(TChart1.Chart)    
Dim tool As LightTool =  New LightTool(TChart1.Chart)    
tool.FollowMouse = True 'slow in Debug build   
line.FillSampleValues()

**SubChart工具**

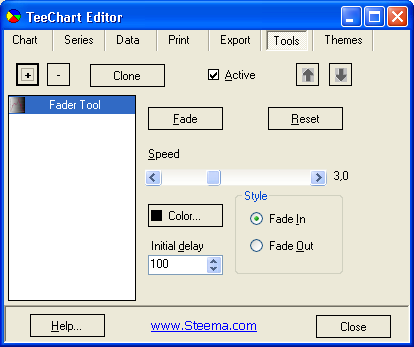
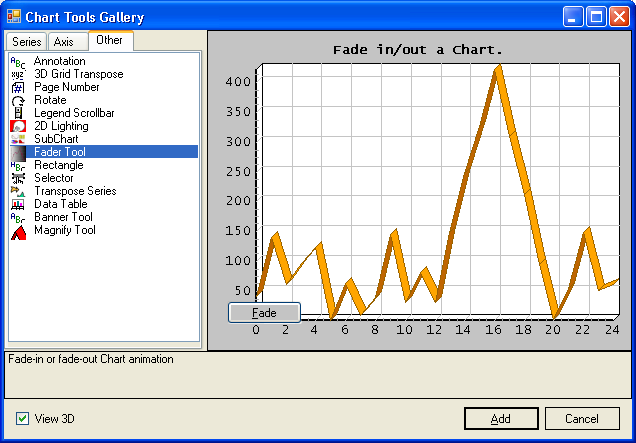
SubChart工具允许在一个单一的图表内包含多个的图表。  
  
**设计时**   
  
  
  
添加SubCharts可通过图表中的Add按钮进行，同时可通过使用嵌套图表编辑器来进行编辑。  
  
  
  
**运行时**   
在运行时添加一个SubChart工具是非常简单的操作，可参考以下范例：  
  
[C#]

Line line = new Line(tChart1.Chart);   
SubChartTool tool = new SubChartTool(tChart1.Chart);   
line.FillSampleValues();   
TChart chart1 = tool.Charts.AddChart("Chart1");   
chart1.Series.Add(typeof(Bar));   
chart1[0].FillSampleValues();

[VB.Net]

Dim line As Line = New Line(TChart1.Chart)   
Dim tool As SubChartTool = New SubChartTool(TChart1.Chart)   
line.FillSampleValues()   
Dim chart1 As TChart = tool.Charts.AddChart("Chart1")   
chart1.Series.Add(GetType(Bar))   
chart1(0).FillSampleValues()

**Fader工具**

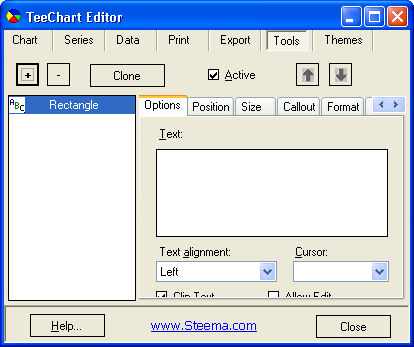
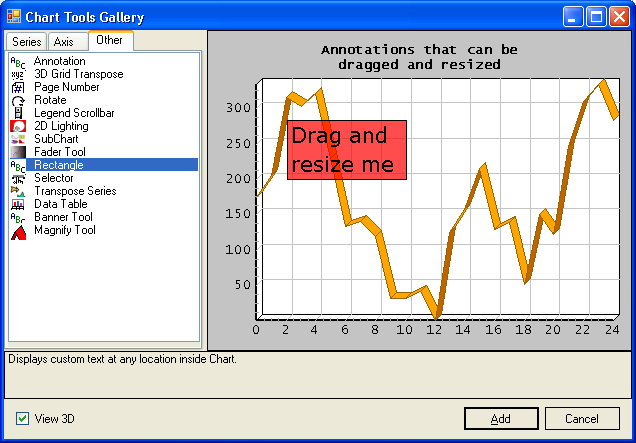
Fader工具可实现图表淡入淡出的动画效果。  
  
**设计时**   
  
  
  
通过使用Fader工具编辑器可设定动画的速度，“无填充”颜色，样式及起始延时。  
  
  
  
**Run-time**   
在运行时添加一个Fader工具是非常简单的操作，可参考以下范例：  
  
[C#]

private FaderTool tool;   
private void InitializeChart()   
   
     Line line = new Line(tChart1.Chart);   
     tool = new FaderTool(tChart1.Chart);   
     tool.Color = Color.Yellow;   
     line.FillSampleValues();   
   
   
private void button1\_Click(object sender, EventArgs e)   
   
     tool.Start();

[VB.Net]

Private tool As FaderTool   
Private  Sub InitializeChart()   
     Dim line As Line =  New Line(TChart1.Chart)    
     tool = New FaderTool(TChart1.Chart)   
     tool.Color = Color.Yellow   
     line.FillSampleValues()   
End Sub   
    
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
     tool.Start()   
End Sub

**Rectangle工具**

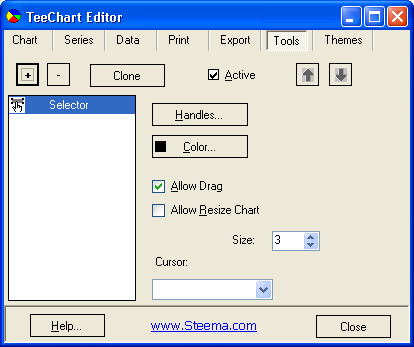
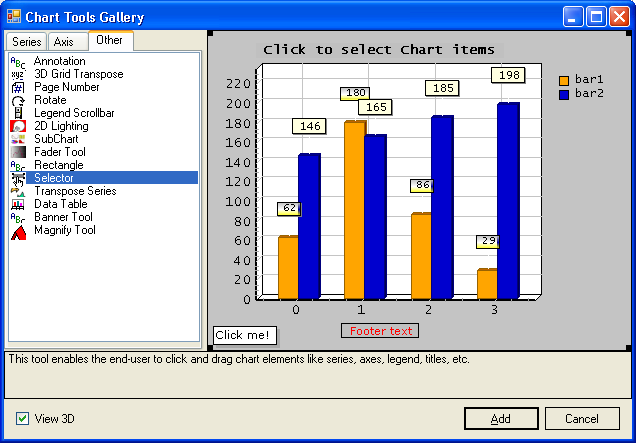
Rectangle 工具是一个可拖拽及调整大小的注释工具。  
  
**设计时**   
  
  
  
该编辑器可对文本及其边界区域进行编辑。   
  
  
  
**设计时**   
在运行时添加一个Rectangle工具是非常简单的操作，可参考以下范例：  
  
[C#]

Line line = new Line(tChart1.Chart);   
RectangleTool tool = new RectangleTool(tChart1.Chart);   
line.FillSampleValues();   
tool.Text = "Text";

[VB.Net]

Dim line As Line =  New Line(TChart1.Chart)    
Dim tool As RectangleTool =  New RectangleTool(TChart1.Chart)    
line.FillSampleValues()   
tool.Text = "Text"

**Selector 工具**

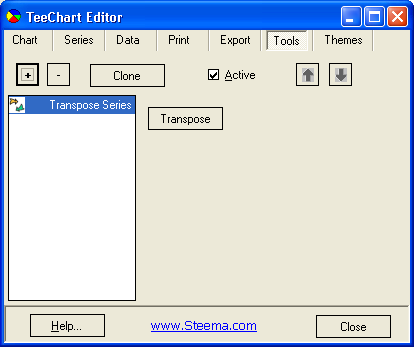
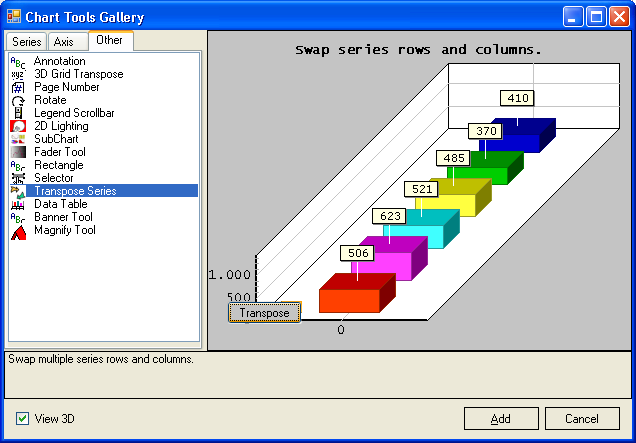
Selector工具能使终端用户点击或拖拽图表的数据序列，坐标轴，图例，标题等元素。  
  
**设计时**   
  
  
  
Selector工具编辑器是用来编辑被选项目的句柄大小及格式刷特性，也能设定可拖动的对象（例如图例）是否能进行拖动操作。在使用工具时的光标类型也能进行设置。  
  
  
  
**运行时**   
在运行时添加一个Selector工具是非常简单的操作，可参考以下范例：   
  
[C#]

Bar bar = new Bar(tChart1.Chart);   
Selector tool = new Selector(tChart1.Chart);   
bar.FillSampleValues();   
tool.Cursor = Cursors.IBeam;

[VB.Net]

Dim bar As Bar =  New Bar(TChart1.Chart)    
Dim tool As Selector =  New Selector(TChart1.Chart)    
bar.FillSampleValues()   
tool.Cursor = Cursors.IBeam

**Transpose Series工具**

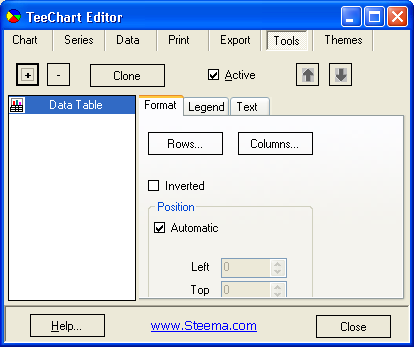
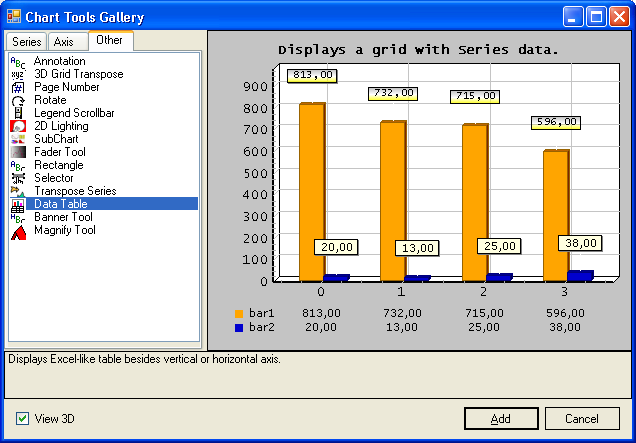
Transpose Series 工具可用来交换多个数据序列的行和列。  
  
**设计时**   
  
  
  
点击Transpose按钮即可执行该操作  
  
  
  
**Run-time**   
在运行时添加一个Transpose Series工具是非常简单的操作，可参考以下范例：   
  
[C#]

private SeriesTranspose tool;   
private void InitializeChart()   
   
     Bar bar = new Bar(tChart1.Chart);   
     bar.MultiBar = MultiBars.None;   
     tool = new SeriesTranspose(tChart1.Chart);   
     bar.FillSampleValues();   
   
   
private void button1\_Click(object sender, EventArgs e)   
   
     tool.Transpose();

[VB.Net]

Private tool As SeriesTranspose   
Private  Sub InitializeChart()   
     Dim bar As Bar =  New Bar(tChart1.Chart)    
     bar.MultiBar = MultiBars.None   
     tool = New SeriesTranspose(tChart1.Chart)   
     bar.FillSampleValues()   
End Sub   
    
Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click   
     tool.Transpose()   
End Sub

**Data Table工具**

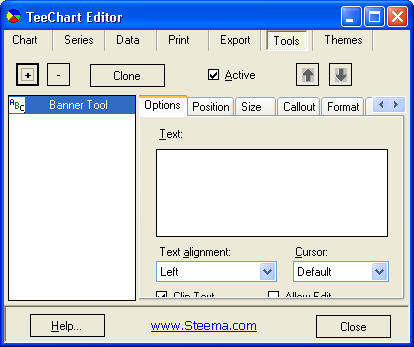
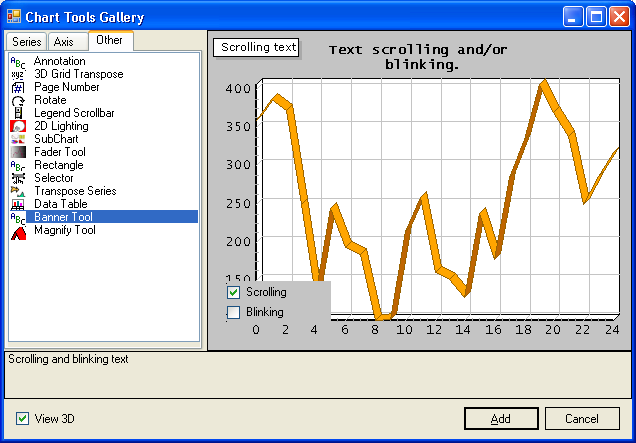
Data Table 工具可显示一个与Excel类似的有横轴和纵轴的表格。  
  
**设计时**   
  
  
  
使用Data Table可对表格的边界（行与列），位置及文本格式进行编辑。  
  
  
  
**Run-time**   
在运行时添加一个Data Table工具是非常简单的操作，可参考以下范例：  
  
[C#]

Bar bar = new Bar(tChart1.Chart);   
DataTableTool tool = new DataTableTool(tChart1.Chart);   
bar.FillSampleValues();

[VB.Net]

Dim bar As Bar =  New Bar(TChart1.Chart)    
Dim tool As DataTableTool =  New DataTableTool(TChart1.Chart)    
bar.FillSampleValues()

**Banner工具**

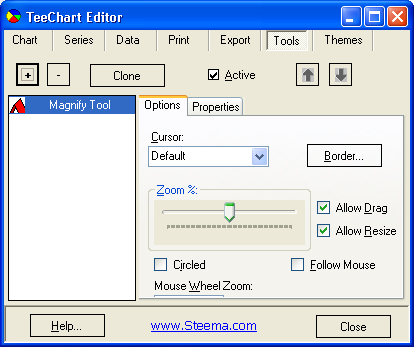
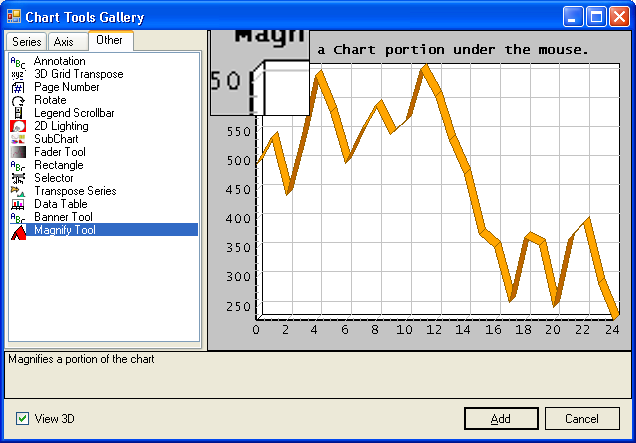
Banner 工具可生成一个滚动且闪烁的文字动画。  
  
**设计时**   
  
  
  
使用Banner工具编辑器可对工具的文本及数据绑定框进行编辑。  
  
  
  
**Run-time**   
在运行时添加一个Banner工具是非常简单的操作，可参考以下范例：  
  
[C#]

Bar bar = new Bar(tChart1.Chart);   
BannerTool tool = new BannerTool(tChart1.Chart);   
bar.FillSampleValues();   
tool.Text = "Text";

[VB.Net]

Dim bar As Bar =  New Bar(TChart1.Chart)    
Dim tool As BannerTool =  New BannerTool(TChart1.Chart)    
bar.FillSampleValues()   
tool.Text = "Text"

**Magnify 工具**

Magnify 工具可放大图表的某一部分。  
  
**设计时**   
  
  
  
使用Magnify工具可它的图像及功能性特征进行编辑。  
  
  
  
**运行时**   
在运行时添加一个Magnify工具是非常简单的操作，可参考以下范例：   
  
[C#]

Bar bar = new Bar(tChart1.Chart);   
Magnify tool = new Magnify(tChart1.Chart);   
bar.FillSampleValues();

[VB.Net]

Dim bar As Bar =  New Bar(TChart1.Chart)    
Dim tool As Magnify =  New Magnify(TChart1.Chart)    
bar.FillSampleValues()

[](#Bookmark18)(See 1.1.17)

### 1.1.19 教程19 - 自定义序列化

**教程19 - 自定义序列化 (自定义元素的序列化)**  
[**引言**](#Bookmark20_Intro)[**序列化的定制Series**](#Bookmark20_Series)[**已知类型的序列化 (eg. string, int, double)**](#Bookmark20_SaveKnownTypes)   
     [定义一个定制Series](#Bookmark20_CustomSeries)   
     [保存及导入](#Bookmark20_StdSaveAndImport)   
  
[**自定义（未知）类型的序列化**](#Bookmark20_SaveUnKnownTypes)   
     [通过定制对象来定义一个定制Series](#Bookmark20_CustomSeriesUnKnown)   
     [保存及导入](#Bookmark20_CustSaveAndImport)   
  
[**自定义图表的序列化**](#Bookmark20_Chart)   
  
**请注意.**    
自定义序列化是一个领先的想法，它能正确保存与恢复开发者设计的自定义元素，该元素也许是相关联的，但它却是在TeeChart自身功能性设置之外。Steema在本教程讲述的技术范围内提供了定制序列化的有限技术支持。

**引言**

**自动式序列化**   
 TeeChart可自动的处理所有必须的Serialization步骤，它允许对TeeChart进行保存和调用。它包括图表及其Series所有修正的元素。  
  
**自定义序列化(自定义元素的序列化)**   
TeeChart Series提供了很多您所需的功能，但是在使用Series时，您可能想要将它修改为包括了您自己信息的附加部分，以便将Series公开、生效或被其他应用程序所引用，为了达到此目的，您也许希望从一个TeeChart Serie创建您自己的SeriesType，并添加您自己定义的元素。TeeChart的定制序列化可保存（序列化）及恢复这些元素。  
  
请注意: 自定义的保存分为2种技术：  
- **已知类型的序列化 (例如：string, int, double)**   
- **自定义(未知)类型的序列化**   
  
本章节将讲解其如何实现

**序列化定制Series****已知类型的序列化 (eg. string, int, double)****定义一个定制Series**

现在假设您希望从TeeChart的Line Series中得到一个Series。Line Series包括了您所需的功能，但是您希望添加一些元素来实现追踪目的或作为增值的项目与任何特定数据集相关联。  
以下是关于如何定义从一个Line Series得到Custom Series的范例：

public class MyLine : Steema.TeeChart.Styles.Line   
{   
  public MyLine(Chart c) : base(c)   
  {   
    myStrVar = "default";   
  }   
   
  public MyLine() : this((Chart)null) { }   
   
  private string myStrVar = "";   
  private int myIntVar = 0;   
   
  public string MyStrProp   
  {   
    get { return myStrVar; }   
    set { myStrVar = value; }   
  }   
   
  public int MyIntProp   
  {   
    get { return myIntVar; }   
    set { myIntVar = value; }   
  }   
}

**保存及导入**

您可以填充您的自定义变量并保存图表，当重新导入已保存的图表时这些自定义变量也会被导入。如果该变量是已知系统类型那么则不需要进行额外的步骤。TeeChart将会处理序列化和反序列化。  
  
Eg.   
保存已修改Series的图表

private void button1\_Click(object sender, EventArgs e)   
{   
  MyLine mLine=new MyLine();   
   
  mLine.MyStrProp = "now set";   
  mLine.MyIntProp = 43;   
   
  tChart1.Series.Add(mLine);   
  mLine.FillSampleValues();   
   
  tChart1.Export.Template.Save(@"c:\files\customLine.ten");   
}

导入已修改Series的图表

private void button2\_Click(object sender, EventArgs e)   
{   
  tChart1.Clear();   
  tChart1.Import.Template.Load(@"c:\files\customLine.ten");   
  label1.Text = ((MyLine)(tChart1[0])).MyStrProp;   
  label2.Text = ((MyLine)(tChart1[0])).MyIntProp.ToString();   
}

Label1 和Label2 将在保存图表前显示设定的值。

**自定义(未知)类型的序列化**

**通过定制对象来定义Custom Series**

通过使用定制项目的元素创建Custom Series类似于使用已知类型的元素来创建Serie。

下面是从Line Series得到一个Series的范例，Line Series包括了已知及未知类型(MyObj)的新元素。由于TeeChart不知道如何序列化和反序列化未知类型的Serie，所有我们需要提供一个您可通过写程序来实现它的方法，并在序列化/反序列化时调用该方法。 该方法是通过继承**Steema.TeeChart.Export.TemplateExport.ICustomSerialization 接口中的类**来实现. 该方法要求在你的类中添加两个方法：  
**- public void Serialize(SerializationInfo info)   
- public void DeSerialize(SerializationInfo info)**

using Steema.TeeChart.Export;   
   
// Custom Line class   
public class MyLine : Steema.TeeChart.Styles.Line, TemplateExport.ICustomSerialization   
{   
  public MyLine(Chart c) : base(c)   
  {   
    myStrVar = "default";   
  }   
   
  public MyLine() : this((Chart)null) { }   
   
  private string myStrVar = "";   
  private int myIntVar = 0;   
  private MyObj myMyObj;   
   
  public string MyStrProp   
  {   
    get { return myStrVar; }   
    set { myStrVar = value; }   
  }   
   
  public int MyIntProp   
  {   
    get { return myIntVar; }   
    set { myIntVar = value; }   
  }   
   
  /// <summary>   
  /// Method to ease population of the myMyObj variable.   
  /// </summary>   
  public void setMyObj(int i, string s)   
  {   
    myMyObj = new MyObj(i, s);   
  }   
   
  /// <summary>   
  /// Property of Custom type, MyObj   
  /// </summary>   
  public MyObj MyObj   
  {   
    get { return myMyObj; }   
    set { myMyObj = value; }   
  }   
   
  /// <summary>   
  /// Obligatory method (TemplateExport.ICustomSerialization interface)   
  /// </summary>   
  public void Serialize(SerializationInfo info)   
  {   
    //save custom elements    
    info.AddValue("myObjStr", myMyObj.MyString, myMyObj.MyString.GetType());   
    info.AddValue("myObjInt", myMyObj.MyInt);   
  }   
   
  /// <summary>   
  /// Obligatory method (TemplateExport.ICustomSerialization interface)   
  /// </summary>   
  public void DeSerialize(SerializationInfo info)   
  {   
    //recover custom elements into the myMyObj variable   
    myMyObj = new MyObj(info.GetInt32("myObjInt"), info.GetString("myObjStr"));   
  }   
}   
   
//custom object   
public class MyObj   
{   
  private string myString;   
  private int myInt;   
   
  public MyObj(int i, string s)   
  {   
    myString = s;   
    myInt = i;   
  }   
   
  public int MyInt   
  {   
    get { return myInt; }   
    set { myInt = value; }   
  }   
   
  public string MyString   
  {   
    get { return myString; }   
    set { myString = value; }   
  }   
}

注意序列化使用 'info.AddValue'来保存自定义元素的值。反序列化根据所要保存变量的类型使用'info.Getxxxx'。

**保存及导入**

一旦您已经在您定制的类中填充程序，用来实现序列化及反序列化您自定义元素，那么就前面所描述的未更改图表来说，图表保存及加载可通过相同的方式来调用，  
  
Eg.   
保存已修改Series的图表

private void button1\_Click(object sender, EventArgs e)   
{   
  MyLine mLine=new MyLine();   
   
  mLine.MyStrProp = "now set";   
  mLine.MyIntProp = 43;   
  mLine.setMyObj(43, "set val 43");   
   
  tChart1.Series.Add(mLine);   
  mLine.FillSampleValues();   
   
  tChart1.Export.Template.Save(@"c:\files\customLine.ten");   
}

导出已修改Series的图表

private void button2\_Click(object sender, EventArgs e)   
{   
  tChart1.Clear();   
  tChart1.Import.Template.Load(@"c:\files\customLine.ten");   
  label1.Text = ((MyLine)(tChart1[0])).MyStrProp;   
  label2.Text = ((MyLine)(tChart1[0])).MyIntProp.ToString();   
  label3.Text = ((MyLine)(tChart1[0])).MyObj.MyInt.ToString();   
  label4.Text = ((MyLine)(tChart1[0])).MyObj.MyString;   
}

Label1, Label2, Label3 及Label4 将在保存图表前显示设定的值。

**自定义图表的序列化**

该步骤必须从这些适用的Custom Series来自定义的图表。  
  
从Steema.TeeChart.Chart派生您的自定义图表，将它标记为Serializable，并将其设为ICustomSerialization类（见以下示例）。就如上面的Series范例，对已知类型的序列化不需要进行任何步骤。对于未知的类型，我们将使用与Series 范例MyObj中相同的对象，

例如：有已知和未知自定义变量的示例图表

[System.Serializable()]   
public class MyChart : Steema.TeeChart.Chart, TemplateExport.ICustomSerialization    
{   
  private MyObj myMyObj;   
   
  public MyChart() : base() { }   
   
  //required constructor   
  protected MyChart(SerializationInfo info, StreamingContext context) : base(info, context) { }   
   
  private string myStrVar = "";   
  private int myIntVar = 0;   
   
  public string MyStrProp   
  {   
    get { return myStrVar; }   
    set { myStrVar = value; }   
  }   
   
  public int MyIntProp   
  {   
    get { return myIntVar; }   
    set { myIntVar = value; }   
  }   
   
  public void setMyObj(int i, string s)   
  {   
    myMyObj = new MyObj(i, s);   
  }   
   
  public MyObj MyObj   
  {   
    get { return myMyObj; }   
    set { myMyObj = value; }   
  }   
   
  //required method   
  public void Serialize(SerializationInfo info)   
  {   
    object o = myMyObj.MyString;   
    info.AddValue("myObjStr", o, o.GetType());   
    info.AddValue("myObjInt", myMyObj.MyInt);   
  }   
   
  //required method   
  public void DeSerialize(SerializationInfo info)   
  {   
    myMyObj = new MyObj(info.GetInt32("myObjInt"), info.GetString("myObjStr"));   
  }   
}

在保存图表时Serialisation可自动进行处理。  
  
例如：序列化   
private void button1\_Click(object sender, EventArgs e)

{   
  MyChart myChart = new MyChart();   
  myChart.Series.Add(new Steema.TeeChart.Styles.Bar());   
  myChart[0].FillSampleValues();   
  myChart.setMyObj(22, "set 22");   
  myChart.MyStrProp = "set String Prop";   
  myChart.Export.Template.IncludeData = true;   
  myChart.Export.Template.Save(@"c:\files\customChart.ten");   
}

反序列化要求一个额外的步骤，TeeChart的反序列要求MyChart与序列化图表绑定，而不是默认类型的图表。  
  
注意设置绑定的代码：

myChart.Import.Template.CustomType = myChart.ToString();

例如：反序列化   
private void button2\_Click(object sender, EventArgs e)

{   
  MyChart myChart = new MyChart();   
  myChart.Import.Template.CustomType = myChart.ToString();   
  myChart = (MyChart)(myChart.Import.Template.Load(@"c:\files\customChart.ten"));   
  myChart.Header.Text = myChart.MyObj.MyString;   
  myChart.Footer.Text = myChart.MyStrProp;   
  myChart.Footer.Visible = true;   
  myChart.Export.Template.Save(@"c:\files\testCustomChartmodded.ten");   
  myChart.Export.Image.PNG.Save(@"c:\files\testCustomChart.png");   
}