目录

1. ComponentOne WinForms 图表一览

使用 **ComponentOne Chart for WinForms** 可以有效地创建具有专业外观的 2D 和 3D 图表.使用 Visual Studio .NET 内置的最新技术,C1Chart 可以完全兼容.NET Frameworks 的 2.0,3.5,4.0 版本. ComponentOne 完全封装了图表组件的底层复杂性,从而使开发人员只需重点关注应用程序中更重要的部分-特定的功能业务。

ComponentOne Chart™ for WinForms 含有两个在 Microsoft Visual Studio .NET 中创建 2D 和 3D 图表的图表控件:C1Chart 和 C1Chart3D.其中 **C1Chart** 控件是一个二维的图表控件,它允许您为任何图表类型应用程序创建多种类型的动态 2D 图表. **C1Chart3D** 是一个三维的图表控件,它用来创建三维表面,三维条形图,三维散点图表,四维条形图,四维表面图表。

ComponentOne Chart™ for WinForms 的全面和详细的文档帮助您在创建 C1Chart 和 C1Chart3D 图表时更加的得心应手.为了更加方便您的阅读, **ComponentOne Chart™ for WinForms** 的帮助文档有两部分:

- **C1Chart2D** 帮助– 介绍有关 **C1Chart** 控件的文档.
- **C1Chart3D** 帮助– 介绍有关 **C1Chart3D** 控件的文档.

为了让您在任何图表类型应用程序中创建具有漂亮的外观,理想的性能的图表. **ComponentOne Chart for WinForms** 给您提供了以下的功能特性:

超过 80 种的 2D 和 3D 图表类型,灵活的可以自定义的图表元素,零代码的交互式设计器,增强的视觉效果(光效果和阴影等),终端用户交互,高级鼠标跟踪能力.

关于 ComponentOne Chart for WinForms 中最新新增的功能,请访问

Chart for WinForms中的新功能.

入门

如果您是刚接触 **Chart for WinForms,**那么从以下主题开始:

- WinForms图表快速入门(27页)
- 特殊的2D图表 (80页)
- 创建2D图表的设计时工具(118页)
- 图表区域和绘制区域对象(page 204)
- WinForms图表指南(268页)

1.1 安装 Chart for WinForms

本章节介绍 **Chart for WinForms** 的安装文件和系统要求.并且也介绍了如何安装 ComponentOne 产品的一个演示版本,还有如何卸载 Chart for WinForms.

1.1.1 Chart for WinForms 安装文件

ComponentOne Chart for WinForms 安装程序会创建以下的目录: C:\Program Files\ComponentOne\Studio for Winforms.该目录含有以下的子目录:

Bin	含有ComponentOne的所有的二进制文件的拷贝(DLL,EXE).
H2Help	包含所有Chart for WinForms控件的在线文档.

C1Chart 包含Chart for WinForms产品的相关文件(至少含有readme.txt).

ComponentOne Studio for WinForms 帮助安装程序完整地安装了 Microsoft Help 2.0 和 Microsoft Help Viewer help,它们被安装在 **C:\Program Files\ComponentOne\Studio for WinForms** 目录,并含有以下的目录:

H2Help 含有所有的Studio组件的Microsoft Help 2.0完整的文档.

HelpViewer 含有所有的Studio组件的Microsoft Help Viewer Visual Studio 2010完整的文档.

示例

产品的实例程序默认安装在 ComponentOne Samples 目录下. ComponentOne Samples 的目录在 Windows XP 和 Windows Vista 的机器上是有略微的不同的:

Windows XP 下的路径: C:\Documents and Settings\<username>\My Documents\ComponentOne Samples

Windows Vista 下的路径: C:\Users\<username>\Documents\ComponentOne Samples

ComponentOne Samples目录含有以下的子目录:

Common 提供供大部分的示例程序使用的帮助和数据文件.

C1Chart 提供Chart for WinForms的示例和指南.

可以使用**ComponentOne Sample Explorer**来访问示例.在您的电脑桌面上,点击**Start**菜单,然后依次点击**ComponentOne | Studio for WinForms| Samples |Chart Samples.**

1.1.2 系统要求:

含有以下的系统要求:

Operating Systems: Windows 7
Windows® 2000
Windows Server® 2003
Windows Server 2008
Windows XP SP2
Windows Vista™

Environments: .NET Framework 2.0 或更高级版本. C# .NET Visual Basic .NET

Disc Drive: 如果从CD安装，那么需要CD或者DVD-ROM

1.1.3 安装演示版本

如果您想试用**ComponentOne Chart for WinForms,**但是又没有一个序列号.按照以下步骤完成安装向导并使用默认的序列号.

我们的产品在非注册(演示版)和注册(付费版)两个版本中的唯一的区别是注册版会标记您编译的每一个应用程序,然后在您的程序运行时不会弹出我们的 ComponentOne 提示对话框.

1.1.4 卸载 Chart for WinForms

如果您需要卸载**ComponentOne Chart for WinForms**,那么完成以下步骤:

a. 打开**控制面板,**并选择**添加或删除程序**(在Windows 7/vista中是**程序和特性**).
b. 选择**ComponentOne Studio for WinForms**,并点击**Remove**按钮.
c. 点击**Yes**来删除程序.

1.2 ComponentOne 终端用户许可协议

所有的ComponentOne许可信息,包括ComponentOne终端用户许可协议, ComponentOne 授权模型,和常见的许可问题.都可以在从 http://www.componentone.com/SuperPages/Licensing/获取.

1.3 许可常见问题和解答

本章节介绍了许可在的主要技术特性.这可能帮助您理解和解决在.NET 和 ASP.NET 中使用 ComponentOne 时会碰到的一些许可问题.

1.3.1 什么是许可?

许可是一种通过保证用户有权使用软件产品,来达到保护知识产权的目的的机制.

许可不仅仅是用来防止非法销售软件产品.许多软件厂商,包括 ComponentOne 在内,使用许可来让潜在用户在决定是否购买产品之前先测试产品是否满足他们的需求.

离开了许可,这种形式的分发对生产商是不实际的,同时对用户也很不方便.生产商将不得不采取以下两种方式之一:分发一个仅有有限功能的评估版,或者将管理软件许可的责任转嫁给终端用户,用户很容易忘记他们正在使用的是一个评估板本,而他们并没有购买.

1.3.2 许可如何工作?

ComponentOne 使用的授权模型基于 Microsoft 提供的行业标准.它适用在任何的组件上.

注意: Compact Framework 组件相比于其它的 ComponentOne 组件,因为平台差异而使用了一种稍微不同的运行时许可机制.

当用户购买了产品以后,他会收到一个安装程序和一个序列号.用户在安装过程中会被提示输入保存在系统中的序列号(用户还可以通过点击任何 ComponentOne 产品的 About Box 菜单中的 License 按钮来输入序列号,或者也可以通过重新运行安装向导并在许可对话框中输入序列号来完成).

当一个许可过的组件被添加到一个窗体或者 Web 页面中时, Visual Studio 会从新创建的组件中获取版本和许可信息.当 Visual Studio 查询时,组件会查找存储在系统中的许可信息,并且生成一个运行时的许可和版本信息,同时, Visual Studio 会把这些信息存储在以下两个文件中.

- 一个含有真正的运行时许可的程序集资源文件.
- 一个" licenses.licx"文件,它含有已许可的组件的强命名和版本信息.

Microsoft Visual Studio 会自动添加这两个文件到工程中去.

在 WinForms 和 ASP.NET 1.x 程序中,运行时许可会由 Visual Studio 将其作为嵌入资源保

存到包含控件或者组件的程序集中.在 ASP.NET 2.x 应用程序中,运行时许可信息可能被当做嵌入资源存储在 App_Licenses.dll 程序集中,该程序集用来保存所有本应用程序中寄宿在 WebForms 上的所有控件的运行时许可.**所以,App_licenses.dll 程序集应该总是随着应用程序一起部署**.

licenses.licx 文件是一个含有所有应用程序中用到的已许可的组件的强命名和版本信息的简单的文本文件.每次使用 Visual Studio 进行重新编译资源时,该文本被读取,来给一系列的控件提供运行时许可,并将许可嵌入到适当的程序集资源中.请注意在该文件中编辑或者添加一行能够导致 Visual Studio 给其它控件添加运行时许可.

请注意 licenses.licx 文件通常不出现在 Solution Explorer 中,但是当您点击 Solution Explorer 中的 **Show All Files** 按钮后它就会出现.或者您也可以通过在 Visual Studio 的主菜单中,从 **Project** 菜单中选择 **Show All Files** 来显示该文件.

之后,当控件在运行时被创建时,它从适当的程序集资源中获得运行时许可.该资源在设计时创建,并能够决定是否简单地接受运行时许可,或者抛出一个异常并导致失败,或者给用户提示一些信息来告诉他们用到的控件没有被许可.

所有的 ComponentOne 产品都被设计成当产品没有许可时就弹出许可信息.而不会抛出异常或者阻止应用程序的执行.

1.3.3 常见场景

以下的部分描述了您可能会碰到的许可问题的场景.

1.3.3.1 在设计时创建控件

这是最常见同时也是最简单的场景:用户给窗体添加了一个或者多个控件,同时许可信息被存储在 licenses.licx 文件中,然后,控件能够工作.

请注意这种机制在 Windows Forms 和 Web Forms (ASP.NET)工程中完全一致.

1.3.3.2 在运行时创建控件

这也是相当常见的场景.您不需要在窗体中含有控件的一个实例,但是您可能想在运行时创建一个或者多个控件的实例.

在这种场景下,工程将不会含有 Licenses.licx 文件(或者文件中将不会含有该控件对应的运行时许可),所以许可将会失败.

为了修正这个问题,在工程中给窗体添加一个控件的实例.这会导致 licenses.licx 文件被同步添加,同时一切都会按照预期工作(控件的实例可以在 licenses.licx 文件创建处理后从窗体上删除).

添加一个控件的实例到窗体上,之后再将其删除,这只是一种在 licenses.licx 文件中添加一行关于控件强签名的简单方式.您也可以通过使用记事本或者 Visual Studio 本身来打开文件并添加对应的行.当 Visual Studio 创建应用程序资源时,会查询控件,并将其运行时许可添加到适当的的程序集资源中.

1.3.3.3 从许可过的组件中继承

当一个继承自许可过的控件的控件被创建时,仍然需要在窗体中保存许可信息.可以通过以下两个方式完成:

- 给控件添加一个LicenseProvider属性.
  这会让派生组件类被认为是认证过的.Visual Studio会创建并管理licenses.licx文件,然后基类会照常处理许可过程,而不需要做任何的附件工作.例如:

```
[LicenseProvider(typeof(LicenseProvider))]
class MyGrid: C1.Win.C1FlexGrid.C1FlexGrid
{
// ...
}
```

- 在窗体上添加一个基类组件的实例

这会导致在 licenses.licx 文件中嵌入许可信息.类似于前一个场景,基类组件会查找并使用许可.这个额外的基类组件可以在许可被添加到 licenses.licx 文件中后删除掉.

请注意,C1 许可不会在以下场景下接受一个派生组件的运行时许可:如果运行时许可没有被嵌入在和派生类定义的相同程序集中,并且该程序集是一个 DLL.这个制限是为了防止在没有设计时许可的情况下,一个派生控件类程序集被用在另外一个程序集中.如果您创建了一个这样的程序集,那么您需要做前述动作中的一个来在运行时创建一个组件.

1.3.3.4 在控制台中使用许可过的控件

当构建控制台应用程序时,没有窗体可以用来添加控件,所以 Visual Studio 不会创建 licenses.licx 文件.

在这种情况下,创建一个 Windows Forms 应用程序,并添加所有需要许可的组件到一个窗体中.然后关闭该 Windows Forms 应用程序,并将 licenses.licx 文件拷贝到控制台应用程序的工程中.

请确认 licenses.licx 文件在工程中是嵌入资源.为了做到这一点,在 Solution Explorer 窗体中的 licenses.licx 文件上右键,并选择 **Properties.**在属性窗口中,将 **Build Action** 属性设置为 **Embedded Resource.**

1.3.3.5 在 Visual C++应用程序中使用许可过的控件

在 VC++ 2003 中有一个问题,就是 licenses.licx 文件在构建过程中会被忽略.所以 VC++应用程序中不含有许可的信息.

为了修正这个问题,需要做额外的步骤来编译许可资源并将它们链接到工程中.如下所述:

1. 像通常一样构建C++应用程序,这样会创建一个exe文件,还有含有许可信息的licenses.licx文件.
2. 拷贝licenses.licx文件到目标目录(Debug或者Release).
3. 拷贝C1Lc.exe工具和许可过的DLL到目标目录(不要使用标准的Lc.exe,它有Bug).
4. 使用C1Lc.exe工具来编译licenses.licx文件,命令的形式如下:

c1lc /target:MyApp.exe /complist:licenses.licx /i:C1.Win.C1FlexGrid.dll

5. 将许可链接到工程.为了做到这一步.回到Visual Studio中,在工程上右键点击,选择属性.然后进入Linker/Commond Line选项.然后输入以下命令:
6. /ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
7. 重新编译可执行文件,这样就会在应用程序中包含许可信息.

1.3.3.6 在自动测试中使用许可过的组件

动态装载程序集的自动测试产品可能会导致弹出许可对话框.这是预期行为,因为通常测试程序不会含有必须的许可信息,并且没有简单的方式添加它.

这可以通过在含有或继承自 ComponentOne 控件的程序集的 AssemblyConfiguration 属性中添加 C1CheckForDesignLicenseAtRuntime 字符串来解决.该属性指定 ComponentOne 控件在运行时使用设计时的许可信息.

例如:

```
#if AUTOMATED_TESTING
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime")]
#endif
public class MyDerivedControl : C1LicensedControl
{
// ...
}
```

请注意 AssemblyConfiguration 字符串可以在给定的字符串之前或者之后含有附件的文字.所以 AssemblyConfiguration 字符串也可以用作其他用途.例如:

```
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime,BetaVersion")]
```

这个方法必须用在前述场景中.它需要在测试机器上安装一个设计时许可.给其他机器分发或者安装许可是对最终用户授权协议的侵犯.

1.3.4 移动应用程序上的常见场景

下面的主题描述了您在移动应用程序中可能会遇到的许可场景.

1.3.4.1 升级或者重建许可

如果您重建了您的订阅,那么需要安装新的许可.

如果一个移动控件是通过 Studio 订阅来进行许可的,那么打开一个 ASP.NET 控件或者.NET Windows 窗体控件的 **About Box** 来更新您的许可,这通过点击 **License** 按钮并输入序列号来完成.

如果一个移动控件是通过 Studio for Mobile Devices 订阅来进行许可的,那么打开一个 1.x 版本的移动控件的 **About Box** 或者运行安装程序来输入序列号.目前, 2.x 版本的移动控件不能从 **About Box** 中进行许可或者注册.所以,必须通过另外一个控件或者安装程序来进行许可动作.

通过安装程序来进行 2.X 移动控件的许可

为了在 Studio for Mobile Devices 2.0 安装程序中输入序列号(许可).需要进行以下步骤:

1. 运行ComponentOne Studio for Mobile Devices 2.0安装程序.
2. 按照安装过程中的描述,在提示输入时输入序列号.

在一些情况下安装程序可能不会提示.如果您已经安装了一个合法的许可.或者您在安装一个已经安装过的控件版本(维护模式),这样您将不会被提示来输入序列号.如果您想在维护模式中输入一个序列号.那么您首先需要进行卸载.一旦您完成卸载,重新运行按照程序.

如果您还是在安装程序的安装过程中没有被提示输入序列号,那么您一定已经安装过了一个合法的许可.如果您还是想输入一个新的序列号,那么删除旧的序列号并重新运行安装程序.这里有一个帮助您删除旧的序列号的工具,如下所述.

许可删除器

许可删除器会查找您机器中以下产品的许可信息: Studio Enterprise, Studio for Mobile Devices, 和 individual Studio Mobile.所有找到的许可会在一个列表中显示,这样方便您选择您想要删除的许可.为了在许可删除器中删除一个 Studio for Mobile Devices 许可.按照以下步骤进行:

1. 解压缩并运行C1LicBomb.exe.
2. 您将会看到一个显示了您机器上所有的安装过许可的应用程序的列表.如果您在列表中同时看到了Studio Enterprise和Studio for Mobile devices,选择你看到的其中一个.如果您希望为Studio for Mobile Devices重新输入的新的序列号，不是一个Studio Enterprise序列号,您需要重新输入您的Studio Enterprise序列号.但是这可以通过除了Mobile Studio About Boxes之外的所有C1chart控件的About Boxes来完成.
3. 选择任何您想删除的许可,然后点击**Remove Selected Licenses**按钮
4. 您将会被询问是否确认要删除许可,当该询问对话框出现时,点击**OK**.
5. 删除过程中也会出现一个对话框来提示您某一个特定的许可已经被成功删除,点击OK按钮来继续进行删除动作.
6. 现在,您可以关闭许可删除器,然后运行您的Studio for Mobile Devices安装程序,并在安装过程中输入您的序列号.

按照安装过程中的步骤,当被提示时输入您的序列号.然后 ComponentOne Studio for Mobile Devices 将会在您的机器中被许可.

1.3.4.2 重建许可后更新您的工程

一旦您安装了新的许可,那么每一个工程都应该进行更新,从而能够使用新控件.重新编译控件是不够的.还需要控件重新生成嵌入在它的 **SupportInfo** 属性中的许可信息.为了做到这一点,必须要强制 Visual Studio 更新存储在窗体中的控件属性.最简单的做到这一点的方法是通过简单地更改一个属性.最简单的选择是切换一个布尔属性,例如 **Visible** 属性.然后切换它的值到原始值.这不会导致控件配置的任何改变或者有任何的副作用.但是这会强制 IDE 更新 **SupportInfo** 并嵌入最新的运行时许可.

1.3.4.3 在运行时安装一个移动控件

当移动控件在设计时没有被添加到窗体中,而在运行时被创建时,它表现得跟其它的

ComponentOne 控件没有差别.因为 IDE 在这种情况下没有机会自己获取运行时许可的信息,所以需要强制 IDE 包含一个许可信息.为了做到这一点,在动态创建的组件被创建之前,至少在含有窗体的程序集中包含一个控件的实例.一旦一个相同类型的控件被许可了,窗体上所有相同类型的控件都被认为是许可的.

1.3.5 疑难排解

我们非常用心地让许可机制尽可能的简单明了.但是因为一些原因问题还是会出现.

下面的章节描述了最常见的场景和它们的解决方案

1.3.5.1 我有一个许可过的 ComponentOne 控件版本的产品,但是当我运行工程时仍会出现闪屏

如果出现了这种情况,那么可能是工程中的 licenses.licx 文件出了问题.它可能不存在,或者含有错误的信息,或者没有被正确的配置.

首先,尝试一次完全的重新编译(Visual Studio 中的 Build 菜单中的 Rebuild All 子菜单).这通常会重新编译正确的许可资源.

如果这也失败了,尝试以下的动作:

1. 打开受影响的工程.
2. 选择一个更新过的控件的一个实例
3. 在Visual Studio Properties窗口,改动任何的一个属性.改变了哪一个属性是没有关系的,因为您可以把它的值重新修改为原有值.
4. 使用**Rebuild All**(而不是**Rebuild**)菜单来重新编译工程.然后运行工程.

可选方案1,按照以下的步骤:

1. 打开一个新的Visual Studio.NET工程.
2. 添加一个更新过的控件到窗体上.
3. 编译并运行工程.
4. 打开新工程的licenses.licx文件.
5. 拷贝以更新过的组件的命名空间(例如,C1.Win.C1Report)开头的信息行,并以公钥标记结束.
6. 打开现有的不正确的工程.
7. 打开新工程的licenses.licx文件.
8. 将从第5步中得到的信息行拷贝到这个文件中(使用新的信息替换旧信息).
9. 使用**Rebuild All**(而不是**Rebuild**)菜单来重新编译工程.然后运行工程.

可选方案2,按照以下的步骤:

1. 打开受影响的工程.
2. 从工程中删除licenses.licx 文件.
3. 在窗体上添加一个更新过的控件的实例.
4. 重新编译并运行工程,这次闪屏不会再出现.
5. 在窗体上删除第2步新添加的控件.

如果需要,可以多次尝试可选方案.如果还是没有解决问题,请检查下您是不是在代码中而不

是在设计时创建了一个控件.如果是的话,您必须在 licenses.licx 文件中添加对应的条目(参考 http://helpcentral.componentone.com/PrintableView.aspx?ID=1886 来获取更多的信息).同时如果您的项目是一个网站,而不是 ASP.NET web 应用程序.请在 licenses.licx 文件上右键点击,并且在上下文菜单中选择"Build Runtime Licenses".

1.3.5.2 我在我的 Web 服务器上有一个许可过的 ComponentOne 控件版本的产品,但是表现的像没有许可一样

没有必要在被当做服务器而不是开发使用的机器上安装任何的许可.

控件必须在开发机器上被许可,这样当工程编译时许可信息将会被保存到可执行文件中(.exe 或者.dll).在这之后,应用程序可以被部署到包含服务器在内的任何机器上.

对于 ASP.NET 2.x 应用程序,请确保 App_Licenses.dll 程序集在开发的应用程序被部署到 web 服务器的 bin 目录时被创建.

如果您的 ASP.NET 应用程序以组合许可控件形式使用 WinForms 用户控件.运行时许可信息被嵌入在 WinForms 用户控件程序集中.您必须在嵌入的许可过的控件被更新时重新编译并更新用户控件.

1.3.5.3 我下载了一个许可过的控件的最新版本,但是当我编译我的工程时出现了闪屏

请先检查序列号是否还有效.如果您在一年前对控件进行的许可动作.那么您的分发很可能已经失效.在这这种情况下,您有两种选择:

可选方案1—重建您的分发并获得一个新的序列号

如果您选择这样做,那么您会获得一个新的用来对控件进行许可的序列号(从安装工具或者直接从 **About Box** 获取).

新的分发将会授权您获取所有最新的产品更新。

可选方案2—继续使用您已有的控件

分发过期,但是产品没有过期.您可以在您的分发失效时继续使用您收到或者下载的组件.

1.4 技术支持

葡萄城遵循“专业服务，快速响应”的服务理念，向客户提供与葡萄城控件产品相关的控件选型、试用支持、产品使用金牌服务、项目部署等全程服务，为用户的控件使用保驾护航。

更多详情，请参见葡萄城控件官方网站有关技术服务的介绍。

1.5 再发行文件

ComponentOne Chart for WinForms 基于 ComponentOne LLC 开发和发布.你可以 Microsoft Visual Studio 或者任何支持用户使用和集成控件的开发环境里使用它来开发应用程序.您还可以在这种场景中免费的分发以下的可再分发文件:开发在网络的客户端/工作站端中运行在单独一个 CPU 上的应用程序.

C1.Win.C1Chart.2.dll

C1.Win.C1Chart3D.2.dll

C1.CF.C1Chart.2.dll (Chart for Mobile Devices)

C1.Win.C1Chart.4.dll

C1.Win.C1Chart3D.4.dll

1.6 关于本文档

致谢

Microsoft, Windows, Windows Vista, and Visual Studio 是注册商标或者微软公司在美国和/或者其他国家的商标.

ComponentOne

如果您有关于新特性或者控件的任何建议.请给我们致电或者使用以下地址写信:

西安葡萄城

葡萄城成立于 1980 年，是全球最大的控件提供商、微软公司认证的金牌合作伙伴和 Visual Studio Industry Partner 计划合作伙伴。葡萄城致力于为全球客户提供先进的控件产品和服务，其开发的多代控件产品在日本和欧美市场占据领先地位，在全球拥有 850 多名员工和数十万用户。

更多信息，请访问葡萄城控件产品网站： http://www.gcpowertools.com.cn。

在新浪微博关注 http://e.weibo.com/powertools ，可了解葡萄城控件最新动态。

1.7 命名空间

命名空间是用来组织在一个程序集中定义的对象的.程序集可以含有多个命名空间,而且命名空间允许嵌套定义.

命名空间防止歧义产生,并在诸如类库之类的大量使用对象的程序中简化了引用. ComponentOne Windows产品总的命名空间是**C1.Win**.ComponentOne Web-based产品总的命名空间是**C1.Web**. C1Chart控件的命名空间是**C1.Win.C1Chart**,同时其Web程序集的命名空间是**C1.Web.C1WebChart.**下述的代码片段展示了如何在一个类中使用完整的命名空间来声明一个**C1Chart**控件.

- Visual Basic

```
Dim chart As C1.Win.C1Chart.C1Chart
```

- C#

```
C1.Win.C1Chart.C1Chart chart;
```

命名空间有时候会导致一种叫做命名空间污染的问题.当开发人员在开发类库时使用了另一个使用了相同命名空间的库时会出现这种问题.与现有组件的这种冲突有时被称为命名冲突.

例如,一个名为 **ChartLabels** 的类被创建出来.它能够在工程内部无限制地被使用.但是 **C1Chart** 程序集也包含了一个名为 **ChartLabels** 的实现类.所以,当这两个 **C1Chart** 类被用在同一工程中,必须适用完全命名空间来指定一个确定的引用.如果引用不唯一,那么 Visual Studio .NET 会产生一个错误并提示命名有歧义.

完全有效名称是那些以对象被定义的程序集的命名空间名字开头的对象引用.在其它工程中定义的对象在添加了对象所在程序集的引用后就可以使用(通过在工程菜单中选择添加引用完成).然后在代码中使用完全有效名称来使用对象.

完全有效名称能够让编译器决定正在使用哪个对象而避免了命名冲突.但是,名称本身可能会变得很长或者臃肿.为了解决这个问题,您可以使用 Import 语句(C#中是**using** 语句)来定义一个别名-一个可以用来替代完全有效名称的略名.例如,下面的代码片段为两个对象定义了别名,并在定义这两个对象时使用了别名.

- Visual Basic

```
Imports MyChart = C1.Win.C1Chart
Dim chart As MyChart
```

- C#

```
using C1.Win.C1Chart = MyChart;
MyChart chart;
```

如果没有使用 Import 语句定义别名,那么使用不需要限制来指定在工程中唯一的名称.

作为一个警告,除非显式在代码中指定,它用来确保如下的语句被指定:

- Visual Basic

```
Imports C1.Win.C1Chart
```

- C#

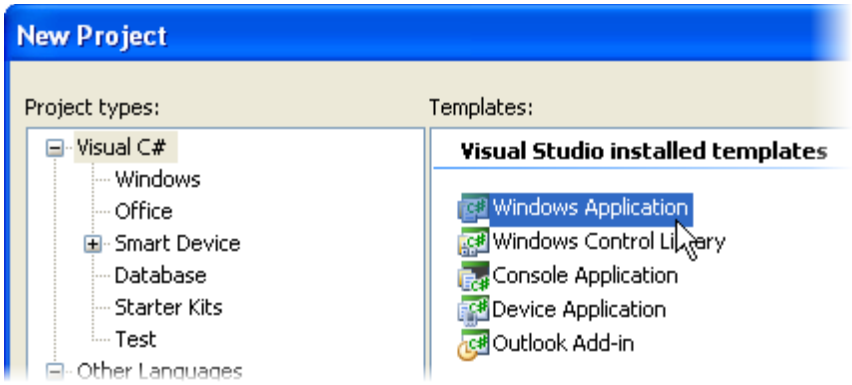
```
using C1.Win.C1Chart;
```

这仅在简单示例代码中适用.指南或者其他更复杂的示例中会指定全名称.

1.8 创建一个.Net2.0 工程

为了创建一个.Net 工程,完成以下步骤:

1. 在Microsoft Visual Studio .NET中的**File**菜单中,选择**New Project**.来打开**New Project**对话框.
2. 在**Project Types**项目中,选择**Visual Basic**或者**Visual C#**.请注意其中的一个会位于**Other Languages**项目下.
3. 在右边的**Templates**选择中选择**Windows Application**



4. 通过输入或者浏览来在**Location**项目中为您的应用程序选择一个目录.然后点击**OK**按

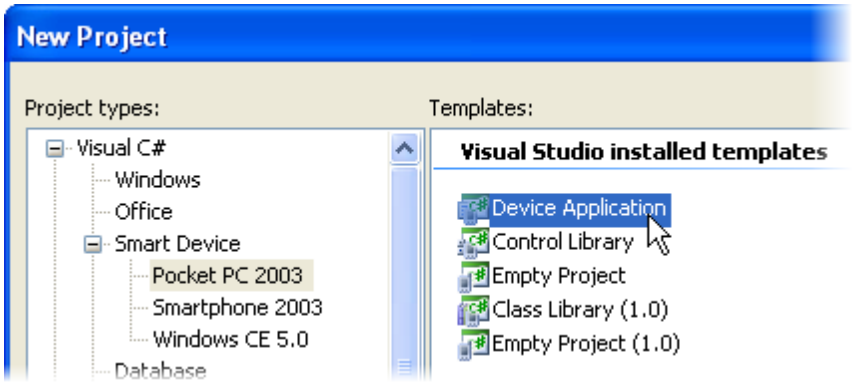
钮.

5. 一个新的Microsoft Visual Studio .NET工程会在制定的目录地址中被创建出来.额外地,一个新的Form1会出现在设计器视图中.
6. 在ToolBar中双击**C1Chart**控件来将其添加到Form1上.关于如何将一个控件添加到**ToolBar**上的更多信息,请参考**在工程中添加一个C1Chart组件**(13页)

1.9 创建一个移动设备应用程序

为了创建一个.Net2.0 移动设备应用程序工程,完成以下步骤:

1. 在Microsoft Visual Studio .NET 2005中的File菜单中,选择**New Project**.来打开**New Project**对话框.
2. 在**Project Types**项目中,选择**Visual Basic**或者**Visual C#**.请注意其中的一个会位于**Other Languages**项目下.
3. 展开**Smart Device**节点,并从列出的设备类型中选择一个.
4. 在右边的**Templates**选择中选择**Device Application.**



5. 在**Name**文本框中输入名字,然后点击OK按钮.一个新的工程会被创建出来.额外地,一个新的Form1会出现在设计器视图中.
6. 在您的工程中添加C1Chart程序集的引用,关于此的更多信息,请参见

在一个工程中添加C1Chart控件

在您安装.Net2.0 的 ComponentOne Studio 时,在安装过程中 **Create a ComponentOne Visual Studio 2005 Toolbox Tab** 复选框默认情况下是选中的.当您打开 Visual Studio 2005 时,一个含有 ComponentOne 控件的 **ComponentOne Studio for .NET 2.0** 选项卡会自动添加到 ToolBox 中.如果您在安装过程中不选中 **Create a ComponentOne Visual Studio 2005 Toolbox Tab** 复选框,那么您可以在安装之后,手动地添加 ComponentOne 控件到 ToolBox 中.

ComponentOne Chart for .NET 2.0 提供了以下的控件:

- C1Chart
- C1Chart3D

如果您需要使用 **C1Chart,**添加这些控件到您的工程中,或者在您的工程中引用 C1.Win.C1Chart 命名空间.

1.10 手动添加 C1Chart 到您的工程中

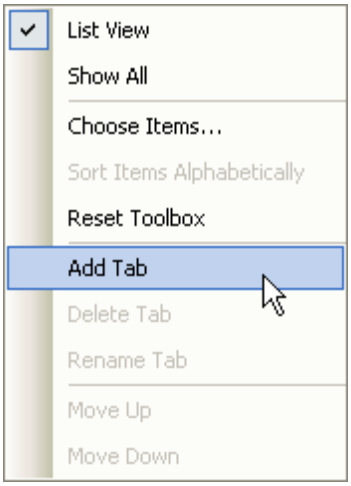
当您安装过 **C1Chart** 后,以下的 **C1Chart** 控件会自动地出现在 Visual Studio Toolbox 自定

义对话框中:

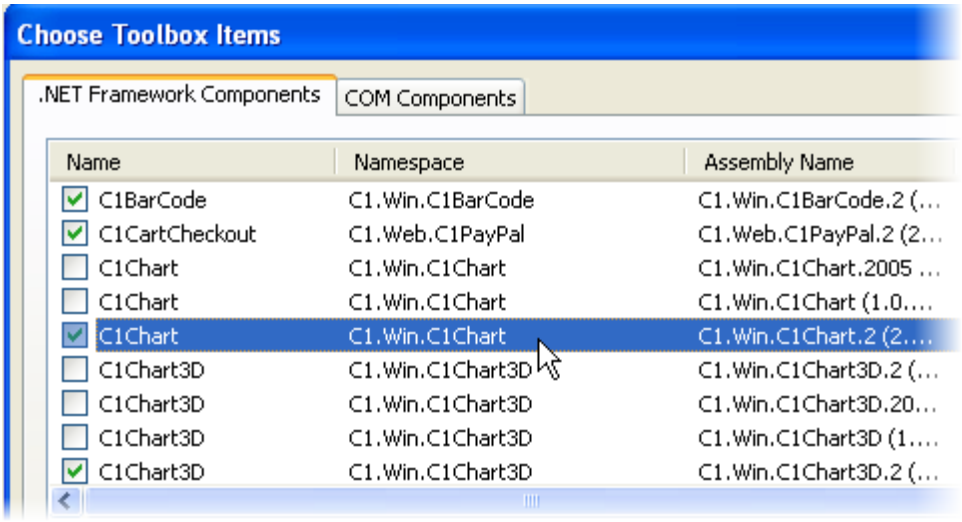
- C1Chart
- C1Chart3D

为了手动添加 **C1Chart** 控件到 Visual Studio Toolbox 中去,需要完成以下的步骤:

1. 打开Visual Studio IDE (Microsoft开发环境).确认ToolBox是可见的(如果需要在**View**菜单中选择**Toolbox**).然后右键点击,来展开它的上下文菜单.
2. 为了让**C1Chart**控件在ToolBox中出现在自己应在的选项卡内,在上下文菜单中选择**Add Tab**,并输入名字,例如**C1Chart.**



3. 在选项卡上右键点击,然后在出现的上下文菜单中选择**Choose Items.**
4. **Choose Toolbox Items**对话框将会出现**.**
5. 在对话框中,选择**.NET Framework Components**选项卡.按照Namespace将项目进行排序(通过在表格的表头上点击Namespace).然后将所有属于C1.Win.C1Chart命名空间的控件前面的复选框都选中.请注意一个命名空间内可能含有多个控件.



在窗体上添加 C1Chart 控件

完成以下步骤来在窗体上添加 C1Chart 控件:

1. 添加**C1Chart**控件到Visual Studio Toolbox中去.
2. 双击控件,或者将其拖拽到窗体上.

添加对程序集的引用

为了添加对程序集的引用,需要完成以下步骤:

1. 在您的工程的Project菜单中选择**Add Reference.**
2. 在**.Net**选项卡中选择**ComponentOne C1Chart**程序集.或者通过浏览来找到C1.Win.C1Chart.2.dll文件并点击**OK**按钮.
3. 双击窗体标题区域来打开代码.在代码文件的开头部分,加入以下**Imports**语句(C#中是**using**):

```
Imports C1.Win.C1Chart
```

注意:这将会使得在 **C1Chart** 程序集中定义的对象在工程中都是可见的.更多信息请参见命名空间(11 页).

1.11 迁移一个 C1Chart 工程到到 Visual Studio 2005

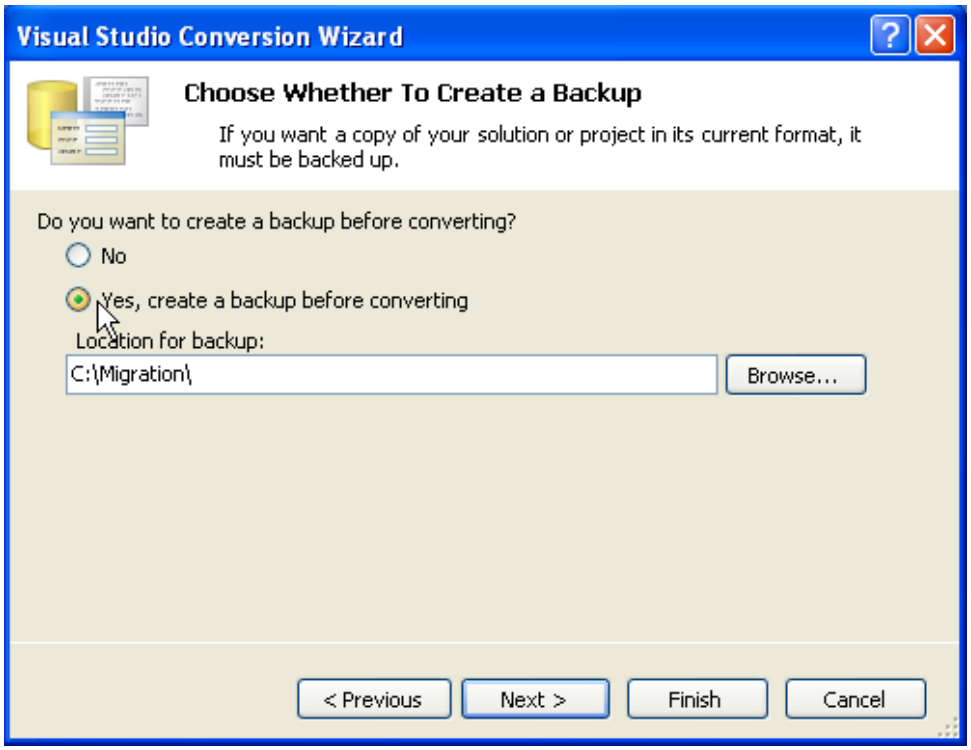
为了迁移一个使用了 ComponentOne 控件的工程到 Visual Studio 2005 中,有两个步骤是必须进行的.首先,您必须将您的工程转换为 Visual Studio 2005 工程.这将会包括将原有引用删除,并指向一些新的引用的过程.第二, .licx 文件必须被更新,才能够让您的工程能够正常运行.

进行工程转换:

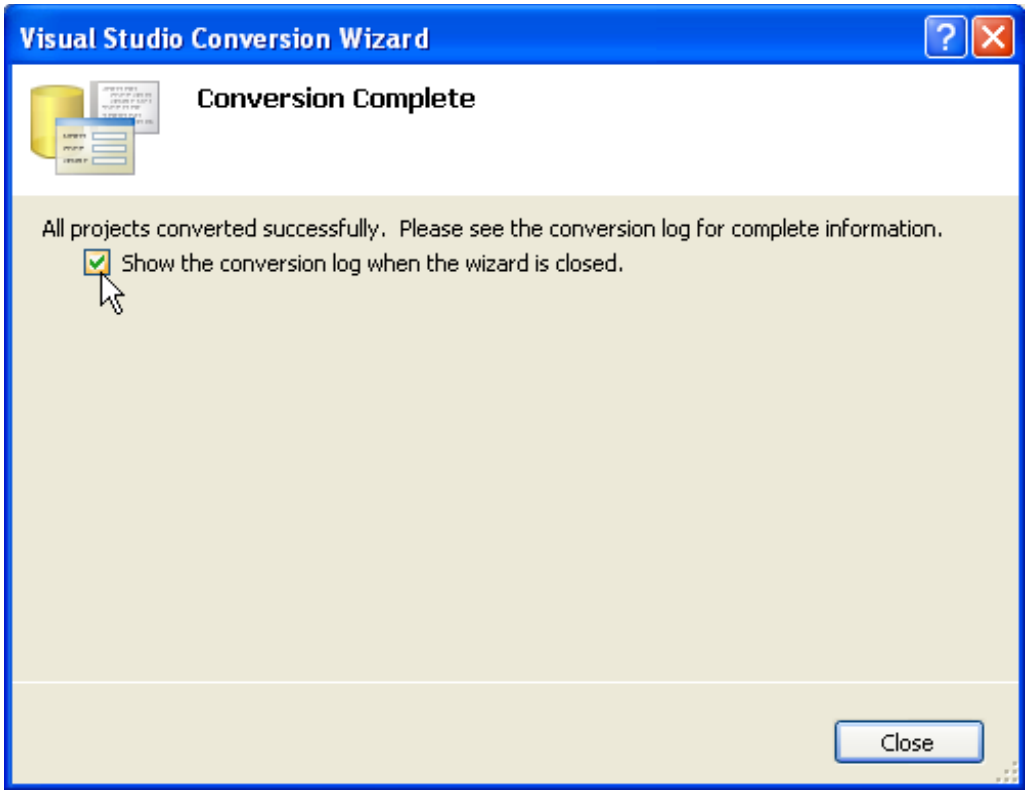
1. 打开Visual Studio 2005 ,然后选择**File, Open Project.**
2. 定位您想要转换到Visual Studio 2005的工程的.sln文件.选择它并点击**Open.Visual Studio Conversion Wizard**对话框将会出现.

3. 点击**Next**.
4. 选择**Yes, create a backup before converting**来给您现有的工程创建一个备份.然

后点击**Next.**

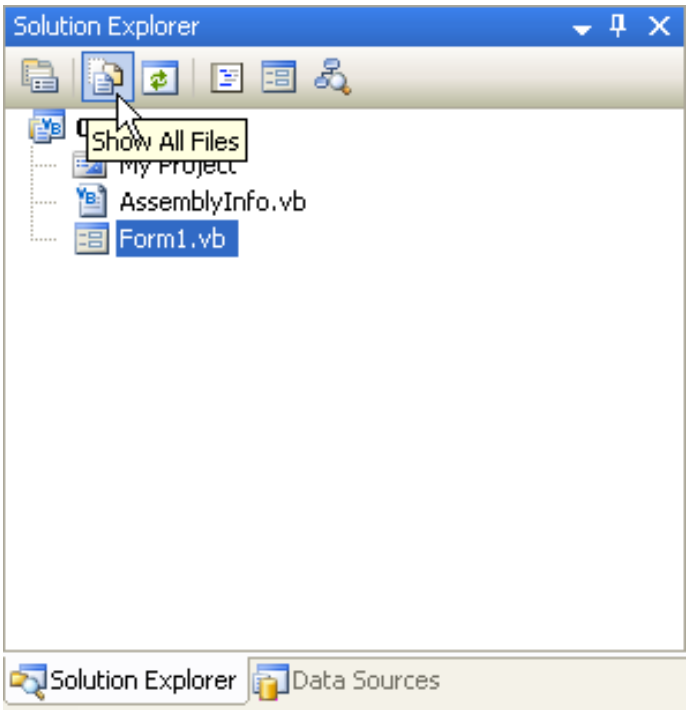


5. 点击**Finish**按钮来完成到Visual Studio 2005的转换. **Conversion Complete**窗口将会出现**.**
6. 如果您想查看转换的Log,那么点击**Show the conversion log when the wizard is closed**复选框**.**



7. 点击**Close**按钮,工程将会出现.现在您必须将工程中原先引用的所有ComponentOne的.dll文件删除,并重新指向新的.
8. 到Solution Explorer (**View, Solution Explorer**)中.选择工程,并点击**Show All Files**按钮.

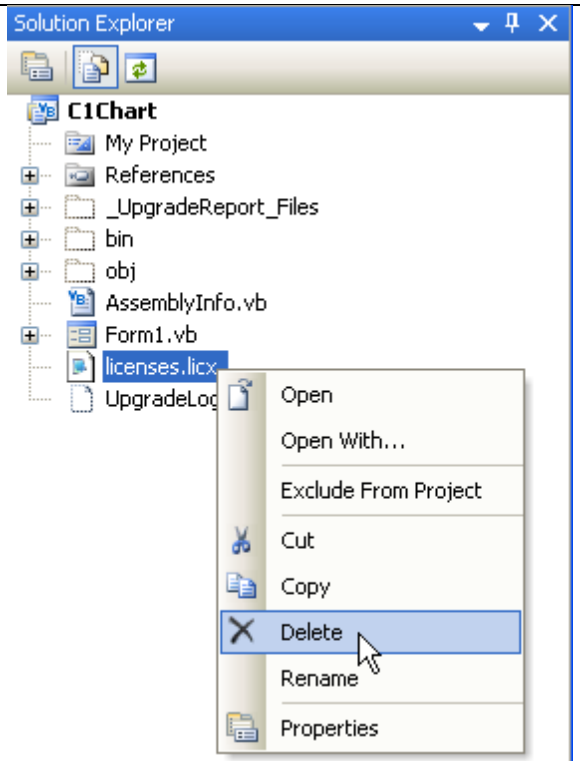
注意: **Show All Files** 按钮在解决方案工程节点没有选中时不会出现在 Solution Explorer toolbar 中.



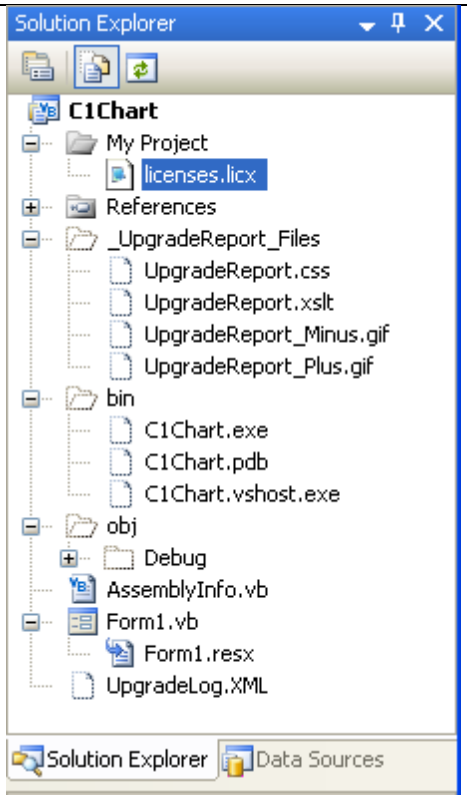
9. 展开**References**节点,右键点击C1.Common并选择**Remove.**然后以同样的操作步骤删除C1.Win.C1Chart引用.
10. 右键点击**References**节点并选择**Add Reference.**
11. 定位并选择**C1.Win.C1Chart.2.dll**文件.然后点击**OK**按钮来添加到工程.

为了更新.lics 文件

1. 在Solution Explorer中,右键点击**licenses.licx**文件,并选择**Delete.**



2. 点击**OK**来永久性地删除**licenses.licx**文件**.**然后工程必须被重新编译来生成一个新的**,**更新过的版本的.licx文件.
3. 点击**Start Debugging**按钮来编译并运行工程.新的.licx文件可能在Solution Explorer中是不可见的.
4. 选择**File, Close**来关闭窗体**,**然后在Solution Explorer中的**Form.vb**或者**Form.cs**文件上双击**,**新的**licenses.licx**文件将会出现在文件列表中.



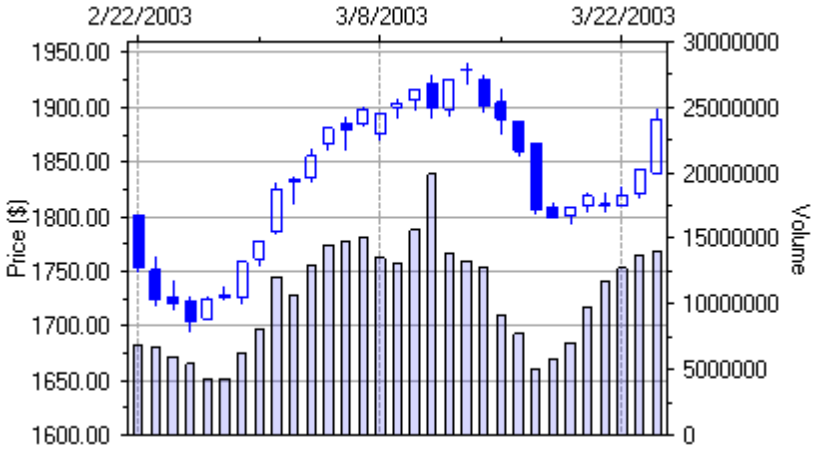
迁移动作已经完成.

2. 关键特性

ComponentOne Chart for WinForms中的**C1Chart**组件含有以下的特性:

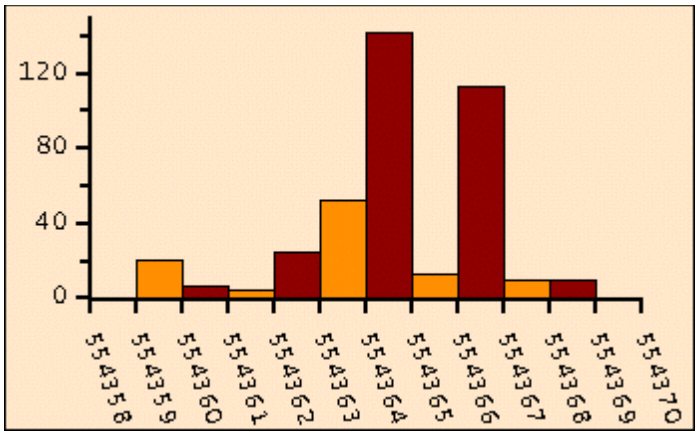
- **可以通过设置一个属性的值,来简单地改变图表的类型**
  将任何的图表重新格式化,例如从线形变成条状图,或者饼状图,可以通过简单地改变一个属性的值完成.从而得到您在任何应用程序中的确切图表形式.
- **SmartDesigner™提供了高度互动的图表编译能力.**
  使用 Chart's ComponentOne SmartDesigner 工具可以大量地节省在处理图表布局上的工作量.不用离开设计器窗体就能完成所有的工作.当您使用鼠标点击时每一个图表元素都会出现内置的工页具栏和编辑器.
  关于**SmartDesigner™**的更多信息,请参见**使用智能设计器来工作**(118页)
- **新接触控件的新手可以使用图表向导中的三个简单步骤来创建图表**
  图表向导引导初学者一步步地完成创建一个控件的完整过程:选择控件类型,修改图表元素,例如表头,页尾和图例,然后是修改图表的数据.
  关于图表向导的更多信息,请参见**使用图表向导**(134页).
- **您再也不用费力为了创建一个图表而在属性窗口中频繁地滚动**
  C1Chart 将所有的元素都分类组织好地放在图表属性设计器中。所以您可以很快地定位一个控件的详细信息.创建或者修改已有的控件:选择简单或者复杂的控件类型,修改数据,轴线,和外观设置.
  关于图表属性设计器的更多信息,请参见**使用图表属性设计器进行工作**(139页).

- **直观地使用可视化效果设计器来可视化地增强图表元素的效果**
  可以通过使用角度,渐变,强度,缩放和形状来改变光源的效果,从而增强图表元素的外观.
  关于图表可视化效果的更多信息,请参见**可视化效果设计器**(231页).关于如何使用图表可视化效果设计器来在图表元素上应用可视化效果的更多信息,请参见**在图表元素上添加可视化效果**(307页).
- **超过20种的内置颜色主题**
  在数据序列上应用与 Microsoft Office 主题类似的颜色主题将变的不可思议的简单.
  关于如何访问颜色主题的更多信息,请参见**为数据序列设置颜色主题**(243页).
- **行业领先的堆叠图表**
  线,区域,条形图,雷达图,绘制图表可以被叠加显示,以在更小的空间内显示更加复杂的数据.
  关于更多叠加控件的信息,请参见**线和XY-绘制图表**(106页).
- **在您的数据分析上添加更加吸引人的视觉效果**
  在您的图表中添加数据高亮,渐近线,警戒区域,可以创建出一个更加有效果,更具可读性的数据图表.
  关于渐近线的更多信息,请参见**使用渐近线进行工作**(174页)
  关于警戒区域的更多信息,请参见**警戒区域**(228页).
- **使用一个属性就可以翻转轴线**
  现在您可以简单地使用一个属性来达到翻转 X 轴线和 Y 轴线的效果.
  关于更多翻转轴线的信息,请参见**翻转和颠倒轴线**(219页).
- **运行时在图表值上操作时的高度交互式行为**
  ComponentOne 图表为选择,缩放(scale),缩放(zoom)操作提供了交互式的内置工具,使用这些工具,您能为您的用户创建高度交互式的图表.
  关于更多终端用户的交互式行为,请参见**旋转,缩放(scale),平移和缩放(zoom)**(263页).
- **C1Chart为图表绘制提供了灵活的图像格式**
  图表可以被保存为任何的图片格式(metafile, BMP, JPG,更多).
  关于保存图表的更多信息，请参见**保存和加载图表,数据,和图片**(255页).
- **在一个单一图表中混用多种图表类型来创建一个吸引人的数据展现**
  可以创建一个区域和散点图混合图表或者条形图和烛状图混合图表.下图展示了在一个含有多种图表类型的单一图表中表示气候信息的效果.



- **当展示在相对狭小的空间内时旋转注解**

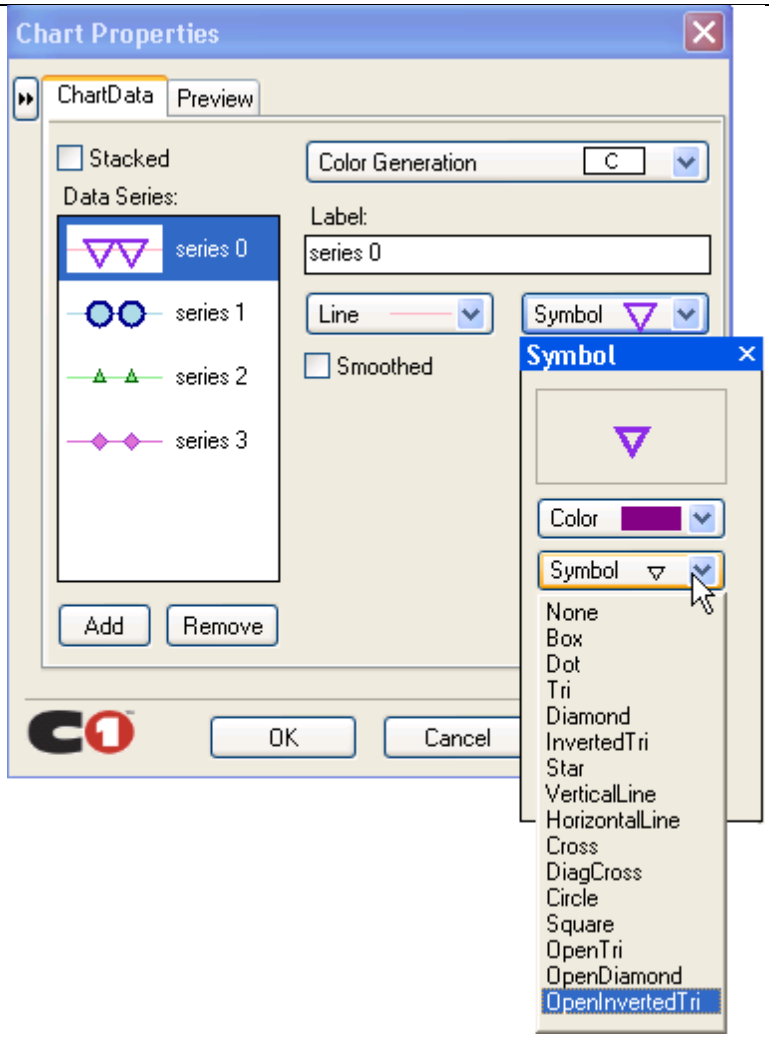
  当展示在相对狭小的空间内时,你可以以任意角度旋转注解来让文字不会溢出.



  关于更多注解旋转的信息,请参见轴线注解旋转(226 页)。

- **添加诸如星状，钻石装，正方形之类的符号来在线，散点图，阶梯图，雷达图等图表中代表不同的数据序列**

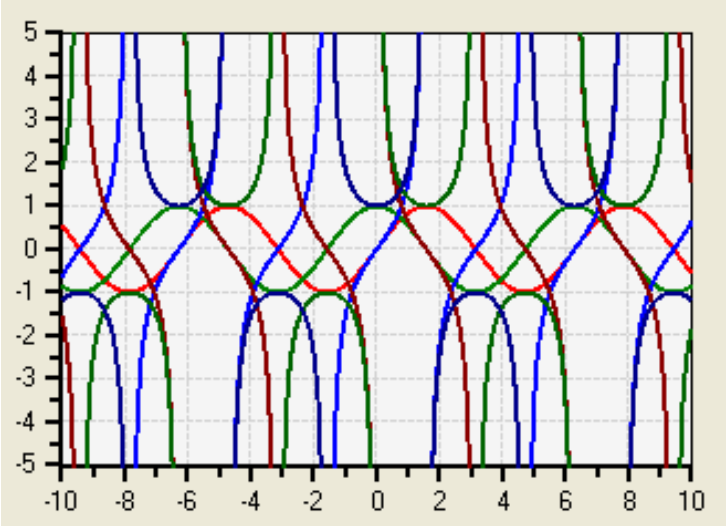
  可以在一个包含很多形状的集合中选择一个符号来代表不同的数据序列，这些符号含有星状，钻石状，正方形，圆圈。可以改变符号的大小和颜色来让它们与各个序列完全匹配。



关于在特殊的数据序列上添加符号的更多信息，请参见**给数据序列添加符号**（301 页）。

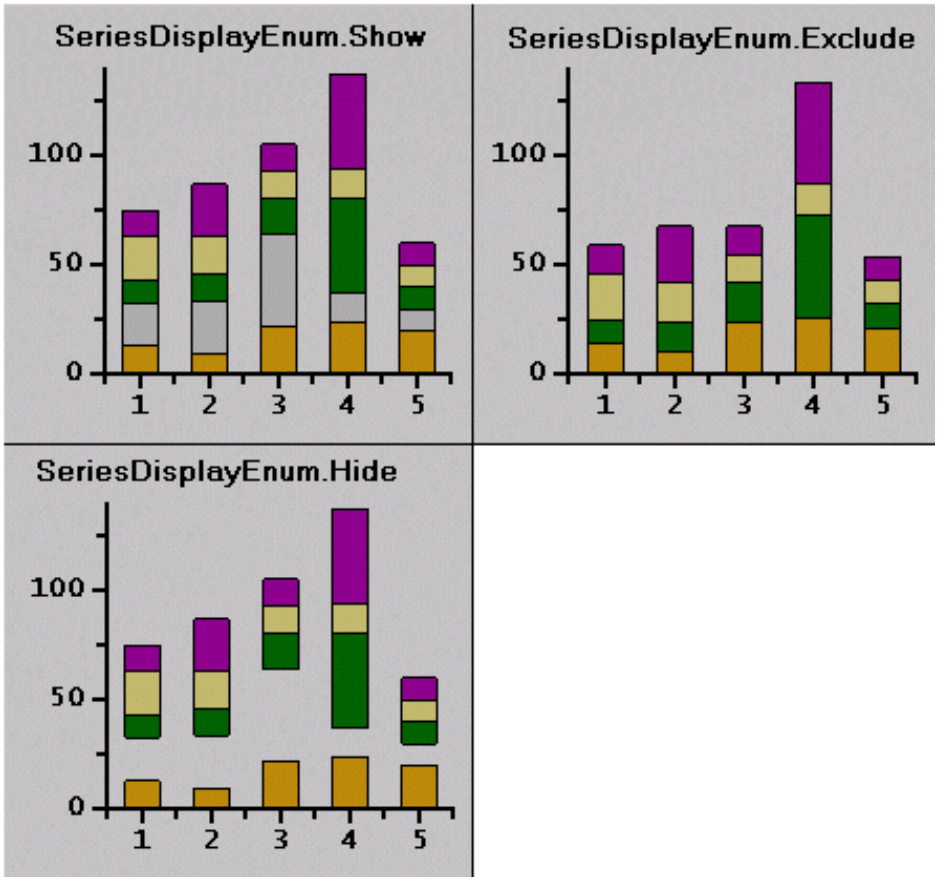
- **绘制高级函数的内置引擎**

  C1Chart 含有一个 FunctionBase Collection 编辑器来帮助您创建和编辑用来绘制显式的和参数化的函数。



关于使用函数来绘制数据的更多信息，请参见**绘制函数**（168页）。

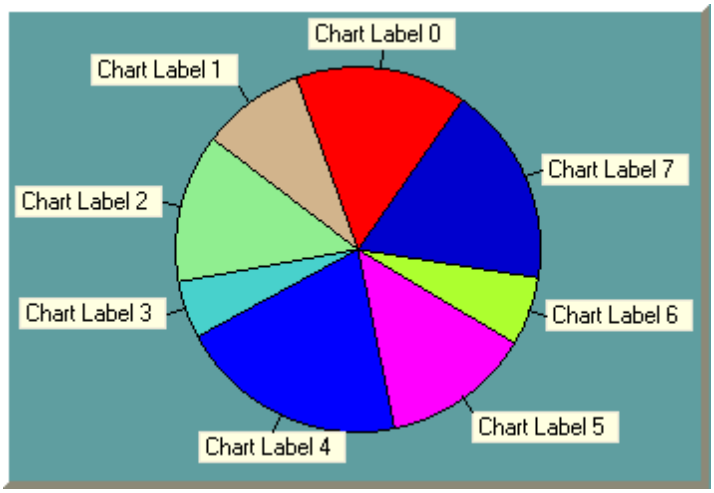
- **隐藏和排除序列来创建诸如浮动柱状图之类的自定义图表**



关于隐藏和排除数据序列的更多信息，请参见**显示，隐藏，和排除数据序列**（164页）。

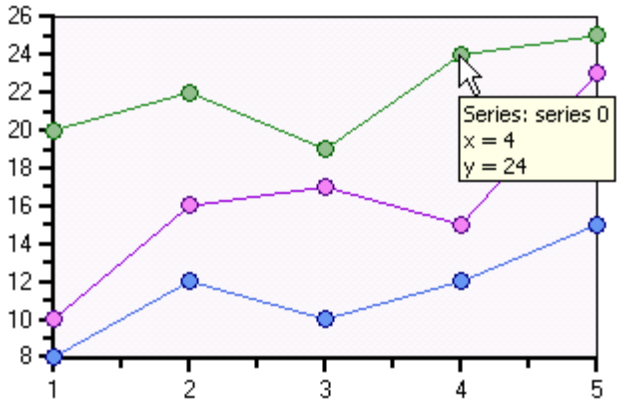
- **自动创建数据标签**

C1Chart 通过 DataLabel 属性来提供了自动创建数据标签的能力。DataLabel 支持一个关键字序号(#TEXT, #XVAL, #YVAL, #YVAL1, #YVAL2, #YVAL3)来统一地完成数据标签任务。



关于添加数据标签的更多信息，请参见**在图表中添加标签**（335页）。

- **灵活和互动的标签**
  图表的标签和注解可以放置在任何位置。多行标签可以无限制地附加到展示在图像中或者用于像素调整的数据上。
  关于使用不同的附件方法的更多信息，请参**附加图表标签**（347页）。
- **图表元素的提示信息**
  使用 ToolTipText 属性来在 C1Chart 元素上突出显示重要的信息



关于在图表元素上添加提示信息的更多信息，请参见**在图表元素上添加提示信息**（304页）。

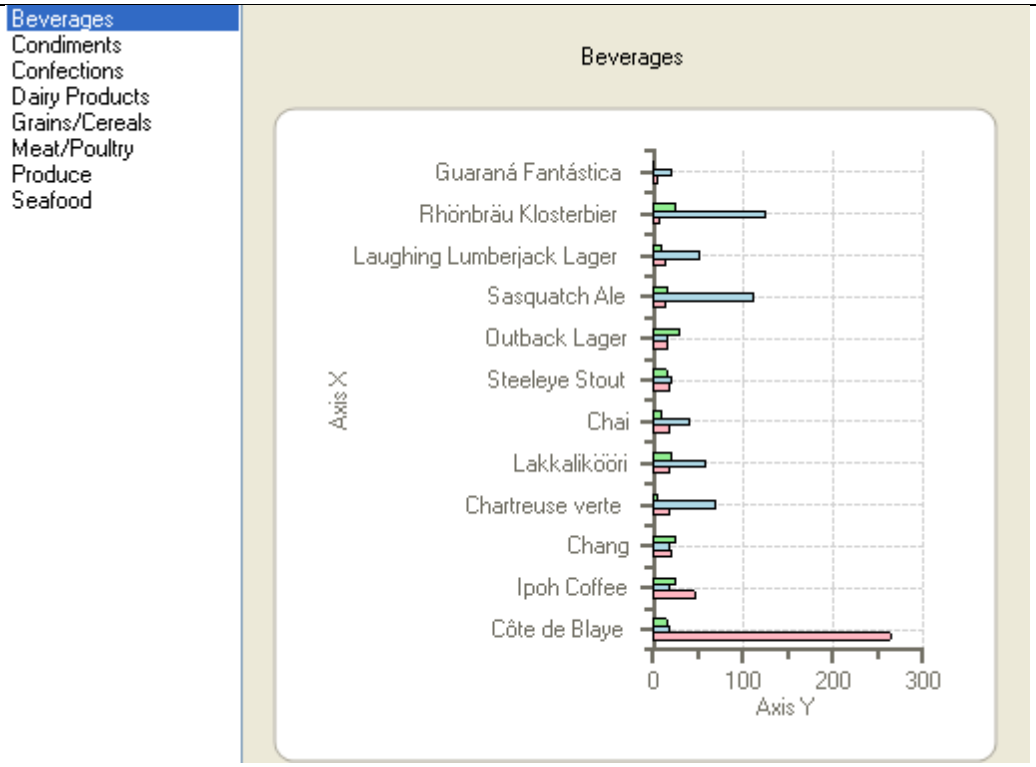
- **自定义画笔**
  使用一个画笔来获得统一的外观，包括影线，渐进，纹理等。
  关于创建自定义的影线和渐进的画笔的更多信息，请参见**为绘制数据自定义画笔**（252页）。
- **高级鼠标跟踪功能**
  提供了一系列和.NET 中的 MouseMove 事件一起使用的转换方法，这让开发人员能够跟踪鼠标位置下的图表区域，序列，和数据点。这让创建具有特殊功能的有意思的程序变得更加容易，例如在图例或图表提示信息上双击，
  关于转换方法的更多信息，请参见**坐标转换方法**（258页）。

3. Chart for WinForms 快速入门

本部分引导您一步步地建立一个报告,它描述了产品的单价,单位库存,和按照类别组织的再订货标准.这个报告使用一个简单的条形图来展示信息,该图中 Y 轴表述了产品的名称,然后 X 轴代表了产品的单价，单位库存,和再订货标准.该图表使用三个序列来绘制产品的单价，单位库存,和再订货标准.一个图例用来说明每个序列的颜色

图表使用一个简单的 Accese 数据库 Nwind.mdb 中的数据.快速入门假设 C1Nwnd.mdb 文件存储在"<PersonalDocumentsFolder>\ComponentOne Samples\Common"路径下,其中的<PersonalDocumentsFolder>路径时用户的 Documents 文件夹,会随着用户和平台的差异而不同.

完成快速入门中的步骤后,会产生一个类似于下图的图表.



3.1 四个步骤中的第一步:为图表创建数据

该步骤创建一个数据源,后续动作可以使用 Chart Properties 设计器来将该数据源绑定到图表中.创建一个.NET 工程,并完成以下的步骤:

添加一个新的数据源

1. 在工程的工具栏中,从**Data**菜单中选择**Add New Data Source. Data Source Configuration Wizard**对话框会出现.
2. 选择**Database,**然后点击**Next**按钮.
3. 点击**New Connection**.
4. 在**Choose Data Source**对话框中,选择Microsoft Access Database File,然后点击**Continue**.
5. 在**Add Connection**对话框中点击**Browse**.
6. 在**Select Microsoft Access Database File**对话框中,选择Nwind.mdb文件(默认放在**C:\Program Files\ComponentOne\Studio for Winforms \Common**路径下),点击**Open**,然后点击**OK**.
7. 点击**Next**来继续操作.一个对话框将会出现,它提示您是否添加数据到您的工程中并且修改链接.因为没有将数据拷贝到工程的必要,所以点击**NO**.
8. 确认**Yes, save the connection as**复选框是选中的**,**然后点击**Next**来继续**.**

a. 链接字符串被当做NwindConnectionString保存

9. 展开**Tables**节点并选中**Categories**和**Products**对象.
10. 点击**Finish**按钮

NwindDataSet.xsd会添加到您的工程中

添加一个OleDbDataAdapter

11. 在工具栏中,双击**OleDbDataAdapter**组件

注意:在 Visual Studio 2005 中,右键点击工具栏.然后选择 **Choose Items**.在对话框中的**.NET Framework Components** 选项卡页中,选择 **OleDbDataAdapter**.

组件托盘上将会出现OleDbDataAdapter,并且**Data Adapter Configuration Wizard**将会出现.

12. 在**Data Adapter Configuration Wizard**中,在下拉列表中选择您将为数据适配使用的连接串(在本例中, **C:\Program Files\ComponentOne\Studio for Winforms Common\Nwind.mdb** \Nwind.mdb).然后点击**Next**按钮.
13. **Use SQL Statements**选项默认是选中的,然后点击**Next**按钮.
14. 在Data Adapter Configuration Wizard的文本输入框中复制和粘贴如下的SQL语句:

```
SELECT CategoryID, ProductName, UnitPrice, UnitsInStock, ReorderLevel FROM Products ORDER BY UnitPrice DESC
```

15. 点击**Next**然后点击**Yes**来在您的查询中增加主键列.
16. 点击**Finish**.

请注意**OleDbConnection1**组件会自动地出现在组件托盘中.

生成一个DataSet

生成一个关于 OleDbDataAdapter1 对象的 DataSet,需要完成以下步骤:

17. 选中**OleDbDataAdapter1**,然后点击它的智能标签,然后点击**Generate DataSet** .然后**Generate DataSet**对话框将会出现.
18. 请确认**Existing**单选按钮是选中的. **Products**表是选中的.然后**Add this dataset to the designer**选项也是选中的,然后点击**OK**.

nwindDataSet1会被添加到组件托盘中.

添加第二个OleDbDataAdapter

19. 在工具栏中,双击**OleDbDataAdapter**组件,来添加另外一个**OleDbDataAdapter**组件到组件托盘中.
20. 选中显示了目录的数据连接,并从下拉列表中选择**C:\Program Files\ComponentOne\Studio for Winforms \Common\Nwind.mdb** .
21. 点击**Next**来继续.
22. 请确认**Use SQL statements**是选中的,然后点击**Next**.
23. **在**Data Adapter Configuration Wizard的文本输入框中,输入以下的SQL语句:

```
SELECT CategoryName, CategoryID FROM Categories
```

24. 点击**Next**,然后点击**Finish**.

为OleDbDataAdapter2生成一个DataSet

生成一个关于 OleDbDataAdapter2 对象的 DataSet,需要完成以下步骤:

25. 选中**OleDbDataAdapter2**,然后点击它的智能标签,然后点击**Generate DataSet**.
26. 在**Generate DataSet**对话框中点击**New**,然后命名它为categoriesDataSet.
27. 请确认Categories表是选中的.然后**Add this dataset to the designer**选项也是选中的,

然后点击**OK**.

categoriesDataSet1会被添加到组件托盘中.

填充DataSets

在Form1_Load事件中,添加以下的代码来给DataSet中填充数据:

- Visual Basic

```
oleDbDataAdapter1.Fill(nwindDataSet1)
oleDbDataAdapter2.Fill(categoriesDataSet1)
```

- C#

```
oleDbDataAdapter1.Fill(nwindDataSet1);
oleDbDataAdapter2.Fill(categoriesDataSet1);
```

添加 DataView 组件

返回到设计视图,然后完成以下步骤:

28. 在工具栏中,双击**DataView**组件来将其添加到组件托盘中.

注意:在 Visual Studio 2005 中,右键点击工具栏.然后选择 **Choose Items**.在对话框中的**.NET Framework Components** 选项卡页中,选择 **DataView**.

29. 在**DataView**属性窗口中以以下方式设置属性:
    - **AllowDelete** = False
    - **AllowEdit** = False
    - **AllowNew** = False
    - **Table** = nwindDataSet1.Products

恭喜您!您已经成功地创建了一个数据源.下一个步骤将会向您展示如何添加一个图表并将其绑定到既存的数据源中,并且还有如何使用 Chart Properties 设计器来轻松地定制您的图表.

3.2 四个步骤中的第二步:绑定数据源到 C1Chart 中

为了将您在前述步骤中创建的数据源绑定到图表中,请完成以下步骤:

1. 在工具栏中双击 **C1Chart** 组件来将其添加到窗体中.将 C1Chart 控件的大小设置为 400X400.
2. 在图表上**右键**点击,然后在菜单中选择 **Chart Properties**.然后 **Chart Properties** 设计器将会出现.
3. 点击 **Gallery,**然后选择 **Simple Types** 选项卡中的 **SelectBar**.在右边的面板中,选择第一行中的第一个条形图图片.然后点击 **Apply** 按钮.
4. 点击 **Data** 然后选择 Series 3.点击 **Remove** 来在条形图表中删除 Series 3.
5. 将 **Series 0** 标记为单价,将 **Series 1** 标记为单位库存,将 **Series 2** 标记为再订货标准.然后点击 **Apply** 按钮.
6. 在树视图中,选择 **Data>Binding**.然后从 **Data source** 的下拉列表中选择 **DataView1**.
7. 在 **Data Series** 下拉列表中点击 **Data** 来选择单价.在数据绑定中,将 X 轴的数据字段设置为 **ProductName.**然后将 Y 轴的数字字段设置为 **UnitPrice.**

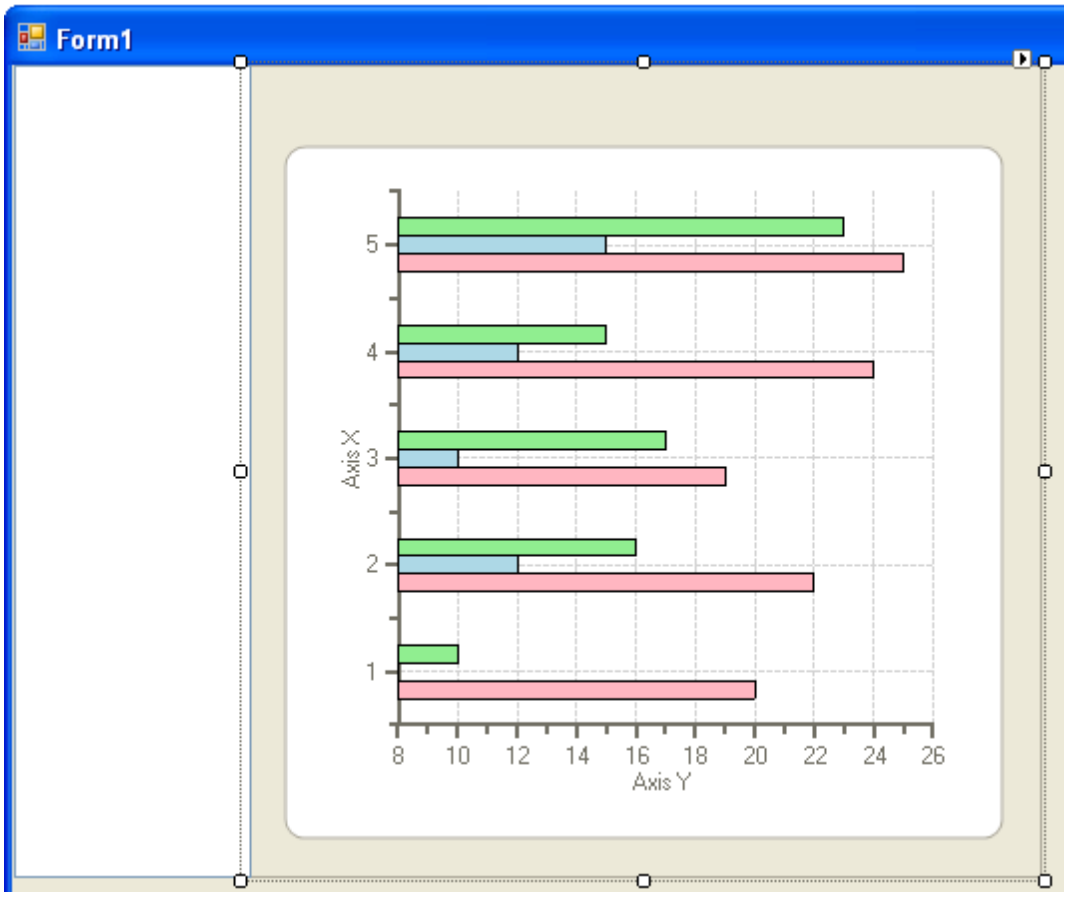
8. 选择单位库存序列.然后将 X 轴的数据字段设置为 **ProductName.**然后将 Y 轴的数字字段设置为 **UnitsInStock.**
9. 选择再订货标准序列.然后将 X 轴的数据字段设置为 **ProductName**.然后将 Y 轴的数字字段设置为 **ReorderLevel.**
10. 这会将 **DataSet** 绑定到图表上.在这之后如果您检查 **ChartGroups>Group0>ChartData>SeriesList**,您将会得到适合 Y 数据字段的数据字段名称.
11. 在树视图中,选择 **Appearance>Header.**
12. 在 **Appearance [Header]**选项卡上,点击 **Visible** 复选框来启用图表表头.
13. 点击 **OK** 来关闭 **Chart Properties.**
14. 运行应用程序

您已经成功地连接图表到数据源并且配置好了图表的设置.为了让我们的数据在图表上更加的可读.在下一个步骤中我们将给数据添加一个列表框来过滤产品种类.

3.3 四个步骤中的第三步:将列表框绑定到 DataSet 中

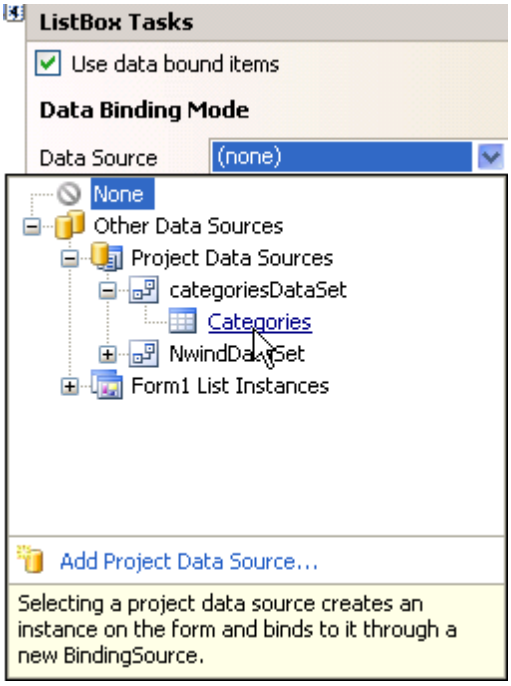
为了将数据进行分类,完成以下步骤:

1. 在工具栏中,双击**ListBox**控件来将其添加到窗体上去.将其停靠在**C1Chart**控件的左边,以达到下图的效果:



2. 选择**ListBox**控件并点击其智能标签来打开菜单.选择**Use databound items,**然后在

Data Source下拉列表中按照**Other Data Sources>Project Data Sources>CategoriesDataSet**的顺序选择**Categories.**



3. 设置**DisplayMember**为**CategoryName**.
4. 在**ListBox**上双击来产生listbox1_SelectedIndexChanged事件的处理.
5. 在L**istbox1_SelectedIndexChanged**事件处理函数中添加以下的代码,来在用户选择一个分类项目是在下拉列表中过滤CategoryID.

- Visual Basic

```
   Private Sub listBox1_SelectedIndexChanged(sender As Object, e As
System.EventArgs) Handles listBox1.SelectedIndexChanged
   If listBox1.SelectedIndex >= 0 Then
   Dim categoryID As String =
Me.categoriesDataSet1.Categories(listBox1.SelectedIndex).CategoryID.ToString
()
   Me.dataView1.RowFilter = "CategoryID = " + categoryID
   Me.c1Chart1.Header.Text = listBox1.Text
   End If
   End Sub
```

- C#

```
    private void listBox1_SelectedIndexChanged(object sender, System.EventArgs
e)
    {
    if (listBox1.SelectedIndex >= 0)
    {
    string categoryID =
this.categoriesDataSet1.Categories[listBox1.SelectedIndex].CategoryID.ToStri
ng();
    this.dataView1.RowFilter = "CategoryID = " + categoryID;
    this.c1Chart1.Header.Text = listBox1.Text;
    }
    }
```

6. 在**Form1_Load**事件中添加如下的代码,以在重新填充后进行新的计算.这样做的话,将会展示第一个分类的产品元素而不是所有未分类的元素.

- Visual Basic

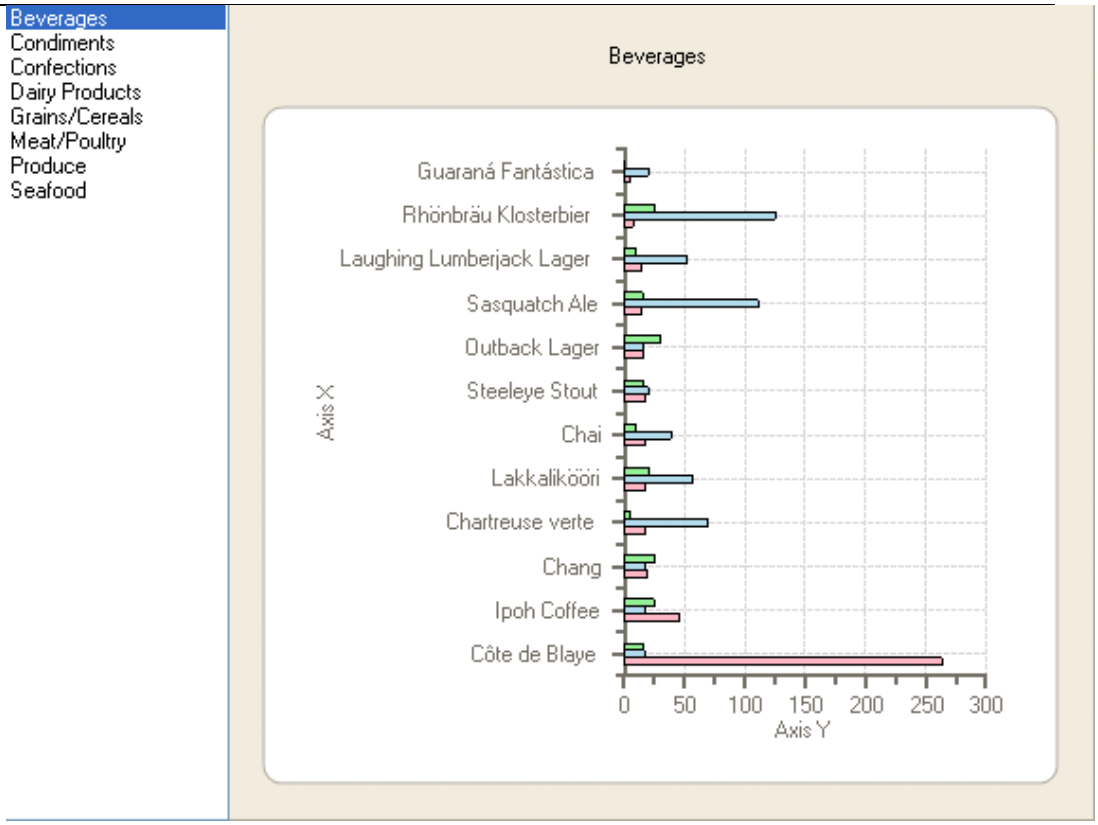
```
    'force the new calculation after the refill
    listBox1_SelectedIndexChanged(Me.listBox1, New EventArgs())
```

- C#

```
    //force the new calcuation after the refill
    listBox1_SelectedIndexChanged(this.listBox1, new EventArgs());
```

7. 运行程序并在下拉列表中选择一个分类来观察进行过数据过滤的图表

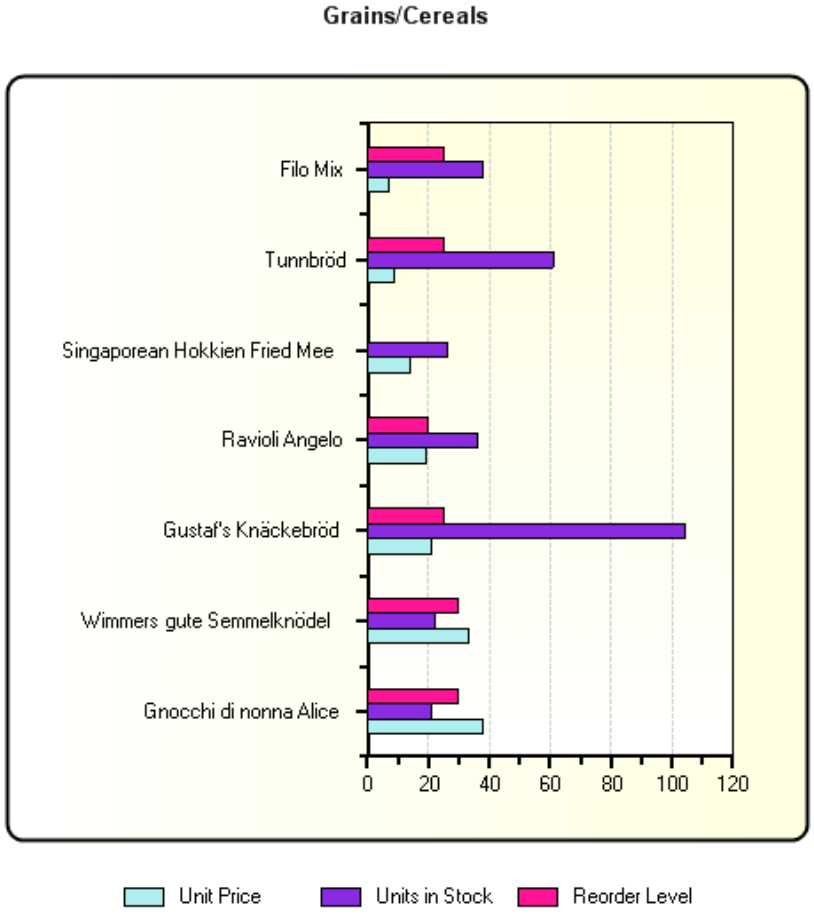
恭喜您!您已经将数据绑定到了图表上.在下一个步骤中,您将修改图表的外观.



3.4 四个步骤中的第四步:自定义图表

在这个部分,您将学习如何使用内置的格式化效果来创建更有视觉效果和看起来更加专业的图表.

当您完成指南的步骤后,您的图表将看起来如下:



C1Chart 将图表的背景色格式化为一个控件的颜色.然后将绘制区域涂成白色.但是您可以选择一个其它的颜色,或者不使用颜色,或者使用两个颜色来创造出很有创意的渐变效果.

改变数据序列的颜色

1. 在**C1Chart**控件上右键点击,并选择**Chart Properties.**
2. 在**Chart Properties**设计器中,选择**Data.**
3. 在**ChartData**选项卡中,在数据序列分组框中选择各个序列,然后在填充下拉列表复选框上点击,来选择一个特定的填充色.
4. 选择**Simple**填充类型,然后将单价的颜色设置为**PaleTurquoise**,库存单位的颜色设置为**BlueViolet**,再进货标准的颜色设置为**DeepPink**.
5. 点击**OK**.

   请注意图表序列上的亮色和暗色的对比会更容易地看出图表序列上的差异.

为了在图表中给数据留有更多的空间,我们可以调整图表图例的位置

修正图例的位置

6. 在**C1Chart**图表上右键点击,并选择**Chart Properties**.
7. 在**Chart Properties**设计器中,选择**Appearance>Legend.**
8. 在**Position**下拉复选框中,选择**South.**
9. 点击**OK**.

这将会释放水平方向上的空间.所以图表将会看起来更加的吸引人和更加的可读.

为了让图表看起来不是那么的拥挤.我们可以删除水平网格线.因为我们仅需要垂直网格线来帮助我们阅读数据.

删除水平网格线

10. 在**C1Chart**图表上右键点击,并选择**Chart Properties.**
11. 在**Chart Properties**设计器中,选择**AxisX>Gridlines**.
12. 在**Major grid group**中,点击Visible复选框来取消选中.AxisX网格线会在图表中消失.
13. 点击**OK**.

为了在图表中更加地突出文字的效果,您可以修改图表的字体样式和颜色.

改变图表表头的字体样式

14. 在**C1Chart**图表上右键点击,并选择**Chart Properties.**
15. 在**Chart Properties**设计器中,选择**Appearance>Header**.
16. 在**ForeColor**下拉列表中点击,然后选择**Black**.
17. 在**Font**按钮上点击来打开Font对话框.改变字体为**Arial**,样式为**Bold,**然后大小为**10**.
18. 点击**OK.**

下一步,我们将改变图表的边框颜色从控件色到黑色来让它跟图表文字的颜色形成鲜明的对比.

改变图表边框颜色

19. 在**C1Chart**图表上右键点击,并选择**Chart Properties**.
20. 在**Chart Properties**设计器中,选择**Appearance>ChartArea**
21. 在**Appearance [ChartArea**]选项卡中,点击**Border**下拉列表复选框,然后选择**Solid**样式, **Black**颜色,和**2**的厚度.
22. 点击**OK.**

为了给图表添加一个样式,我们可以给整个图表添加一些渐变填充效果.

给图表添加渐变填充效果

23. 在**C1Chart图表**上右键点击,并选择**Chart Properties**.
24. 在**Chart Properties**设计器中,选择**Appearance**
25. 在**Appearance**选项卡中,点击Fill下拉列表复选框,然后选择**Gradient**选项.分组框中会出现几种渐变选项.选择第一种渐变并选择**LightYellow**作为**Color1**的颜色, **White**作为**Color2**的颜色.然后点击**OK**.

运行程序来观察图表的新外观.

恭喜您!您已经完成了 C1Chart 的快速入门.在这次的快速入门中您学习到了:

- 静态地绑定数据表到图表中
- 使用 **Chart Properties** 设计器来自定义 C1Chart 的属性.

4. 设计时支持

C1Chart 中的视觉效果设计器来提供了丰富的设计时支持,并且很容易地操作对象模型.

你可以自由地选择使用 Visual Studio 中以下的设计器,菜单,集合编辑器来创建功能强大的图表:

- 智能标签
- 图表向导
- 图表属性设计器
- 视觉效果设计器
- C1Chart 集合编辑器

这个章节讲述了如何使用在 C1Chart 设计时环境中包含的丰富的工具来配置 C1Chart 控件.

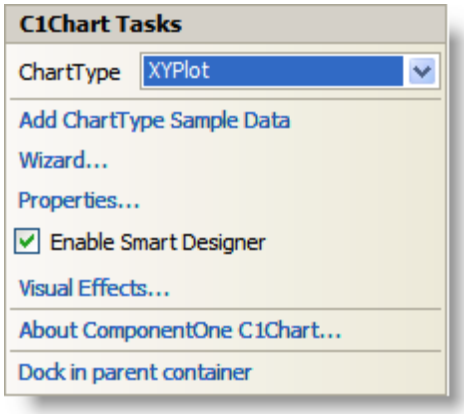
在 Visual Studio 设计时支持之外,C1Chart 提供了特有的设计时支持来帮助您在短时间内创建简单或者复制的图表类型.关于特有的设计时工具的更多信息,请参见用于创建 2D 图表的设计时工具(118 页).

4.1 C1Chart 智能标签

在 Visual Studio 2005 中,C1Chart 组件会含有智能标签.一个智能标签代表一个快捷任务菜单,该菜单提供了该组件最常用的属性.

C1Chart 组件通过使用它的智能标签可以拥有快速访问 **Chart Wizard** 设计器, **Visual Effect** 设计器和常用属性的能力.

为了访问 C1Chart 任务菜单,在 C1Chart 控件的右下方的智能标签(▸)上点击,然后 C1Chart 任务菜单将会打开.



C1Chart 任务菜单含有以下操作选项:

- 添加图表类型的示例数据

  点击 Add CharType Sample Data 会出现一个 Warning 对话框来提示您当前的数据将会丢失.一旦您选择 Yes 来继续操作,当前选中图表类型的的当前或者默认数据将会被修改为示例数据.
- 向导

  在 Wizard 条目上点击会打开图表向导编辑器.关于图表向导编辑器上的元素的更多信息,请参见使用图表向导进行工作(134 页).
- 属性

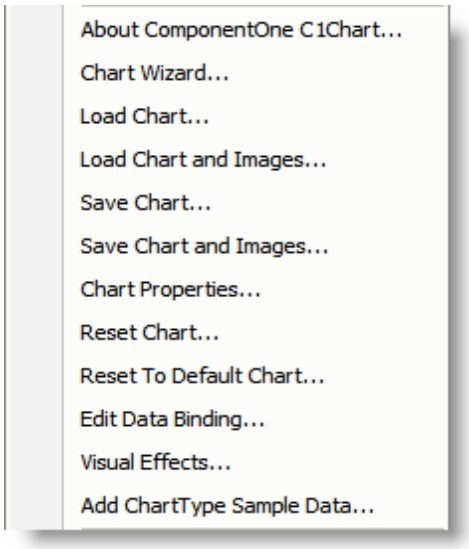
  在 Properties 条目上点击会打开图表属性设计器.关于图表属性设计器上的元素的更多信息,请参见使用图表属性对话框进行工作(139 页).
- 启用智能标签

选择 Enable Smart Designer 复选框会启用 C1Chart 控件的智能设计器.默认值是 True(checked).关于图表智能设计器上的元素的更多信息,请参见使用图表智能设计器进行工作(118 页).

- 视觉效果
  点击 Visual Effects 会打开视觉效果设计器. 关于图表智能设计器上的元素和如何使用它们的的更多信息,请参见视觉效果设计器(231 页).
- 关于 ComponentOne C1Chart
  点击 About 条目会显示 About ComponentOne C1Chart 对话框.使用它能够很容易地找到 C1Chart 的版本号和在线资源.
- 停靠在父容器上
  在父容器上点击 Dock 会将 C1Chart2D 控件停靠在它的父容器上.

4.2 C1Chart 上下文菜单

C1Chart2D 控件提供了一个上下文菜单,它可以在设计时用来使用附件的功能.
为了访问 C1Chart2D 控件的上下文菜单,在 **C1Chart2D** 控件上右键点击.



C1Chart 上下文菜单中含有以下的操作:

- **关于 ComponentOne C1Chart**
  用来显示 About ComponentOne C1Chart 对话框.使用它能够很容易地找到 C1Chart 的版本号和在线资源.
- **图表向导**
  用来打开图表向导.
- **加载图表**
  用来加载保存过的 C1Chart2D 控件的布局.
- **保存图表**
  保存 C1Chart2D 控件的布局到一个 XML 文件中.
- **图表属性**
  打开图表属性设计器

- **重置图表**
  用来重置图表.
- **重置为默认图表**
  将图表还原到原始状态.
- **编辑数据绑定**
  打开 C1Chart 数据绑定编辑器.
- **视觉效果**
  打开视觉效果设计器.

4.3 C1Chart 集合编辑器

C1Chart 提供了以下的集合编辑器,来让您在设计时给图表应用属性.

- 动作集合编辑器
- 警戒区域集合编辑器
- 轴线集合编辑器
- 图表数据序列集合编辑器
- 图表组集合编辑器
- 基于函数的集合编辑器
- 标签集合编辑器
- 点样式集合编辑器
- 缩放菜单项目集合编辑器
- 渐近线集合编辑器
- 值标签集合编辑器
- 视觉效果样式集合编辑器

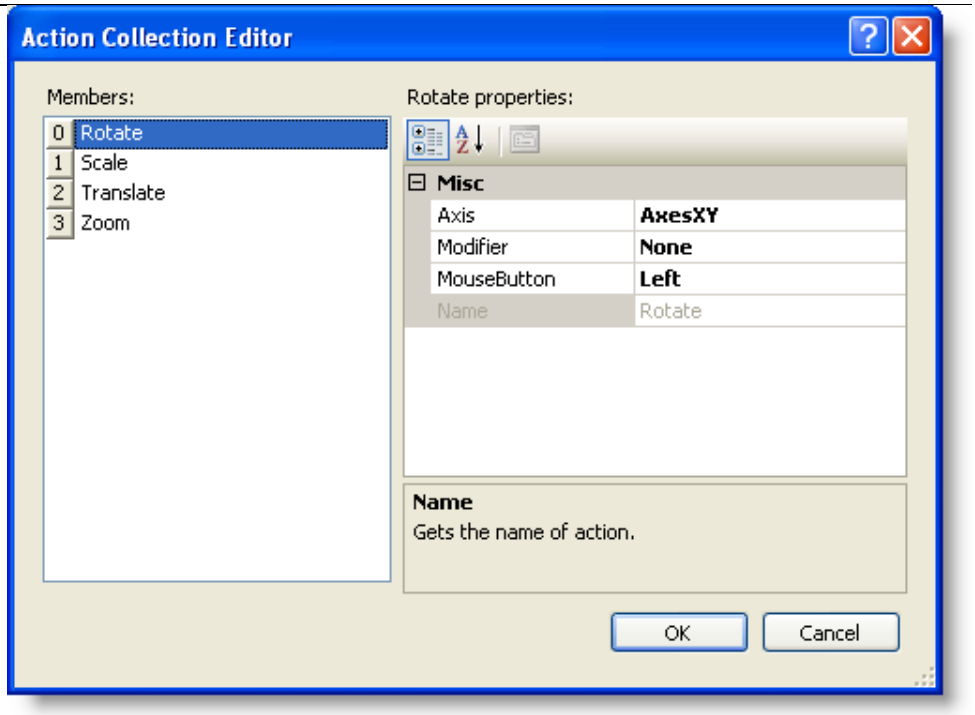
各个编辑器的主要部分都由窗体组成,它让用户可以很方便地编辑C1Chart控件.
下面的主题简要地描述了各个编辑器,并且说明了如何使用它们:

4.3.1 动作集合编辑器

动作集合编辑器用来在设计时进行调定旋转,缩放(scale),平移和缩放(zoom)的交互动作. 关于旋转,缩放(scale),平移和缩放(zoom)的更多信息,**请参见旋转,缩放(scale),平移和缩放(zoom)** (263页).

访问动作集合编辑器

1. 在 C1Chart 控件上右键点击,然后从上下文菜单中选择 Properties
2. 在属性窗口中,展开 Interaction 节点,然后点击紧挨着 Actions 属性的 ellipsis 按钮.Action Collection Editor 对话框将会出现.



在动作集合编辑器中可用的属性

通过 Action Collection Editor,在设计时可以使用以下的属性.或者在运行时可以通过 Interaction 对象来访问这些属性.

成员	描述
Axis	获取或者设置在坐标转换动作中会用到的轴线
Modifier	获取或者设置修改键
MouseButton	获取或者设置开始本动作的鼠标按钮
Name	获取动作的名称

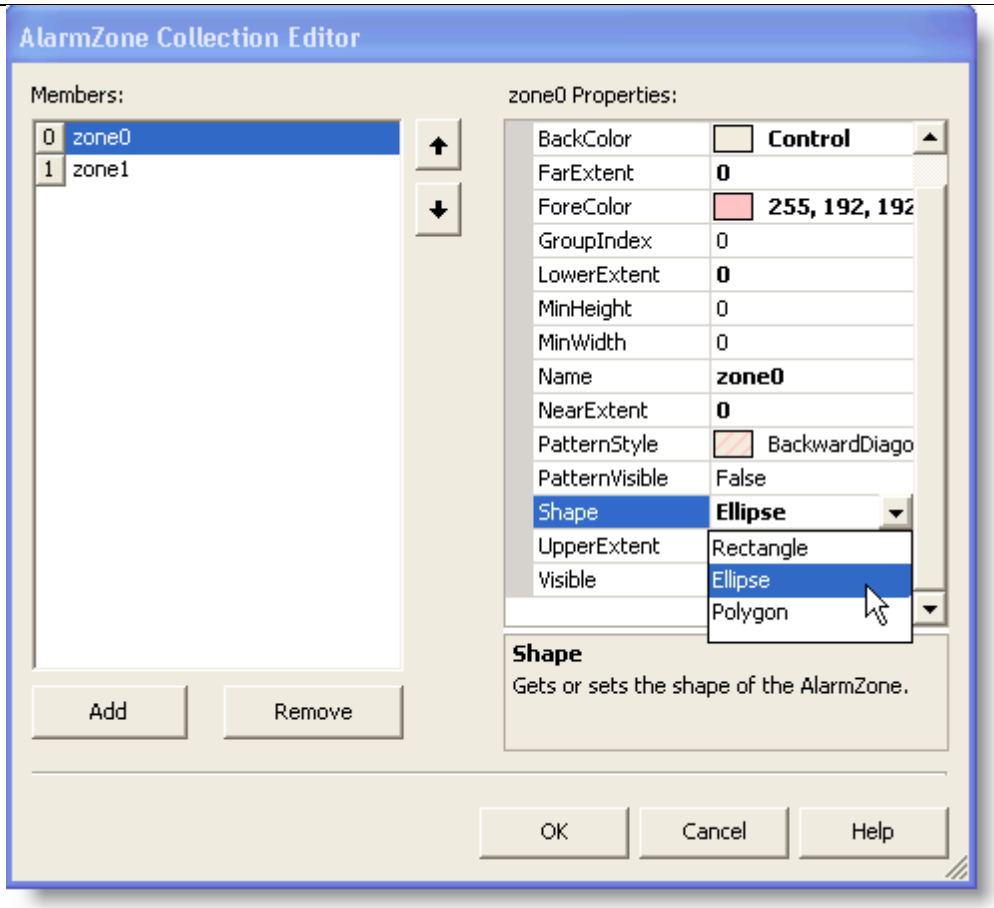
4.3.2 警戒区域集合编辑器

警戒区域集合编辑器用于在运行时给绘制区域添加或修改警戒.关于警戒区域的更多信息,请参见警戒区域(228 页).

访问警戒区域集合编辑器

1. 在 C1Chart 控件上右键点击,然后从上下文菜单中选择 Properties
2. 在属性窗口中,展开 ChartArea 节点,然后展开 PlotArea 节点,然后点击紧挨着 AlarmZones 属性的 ellipes.

警戒区域集合编辑器将会出现.



在警戒区域集合编辑器中可用的属性

通过 **AlarmZone Collection Editor,**在设计时可以使用以下的属性.或者在运行时可以通过 AlarmZone 对象来访问这些属性.

成员	**描述**
BackColor	获取或者设置警戒区域的背景色.从PlotArea中继承.
FarExtent	获取或者设置警戒区域中X轴线数据坐标中的较远的延伸区.
ForeColor	获取或者设置警戒区域的前景色.从PlotArea中继承.
GroupIndex	获取或者设置警戒区域中数据组的索引.
LowerExtent	获取或者设置警戒区域中X轴线数据坐标中的较低的延伸区.
MinHeight	获取或者设置警戒区域的最小的轴线高度.
MinWidth	获取或者设置警戒区域的最小的轴线宽度.
Name	获取或者设置警戒区域的名称.名称可以用来进行索引或者简单定位.
NearExtent	获取或者设置警戒区域中X轴线数据坐标中的较近的延伸区.
PatternStyle	获取或者设置当PatternVisible属性为True时使用的模

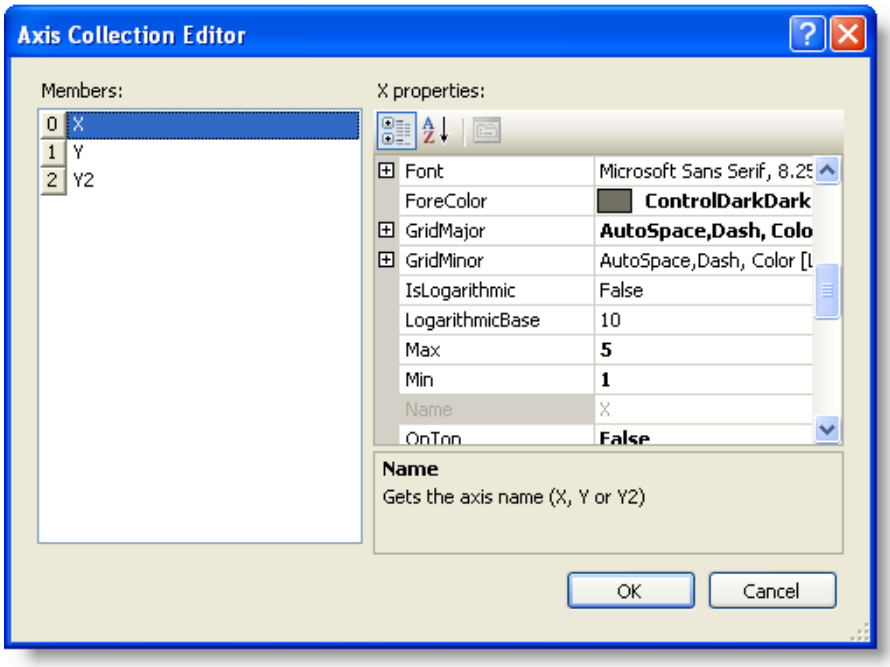
	板样式,PatternStyle用于绘制警戒区域的前景色.
PatternVisible	获取或者设置当PatternVisible属性为True时使用的模板样式,PatternStyle用于绘制警戒区域的背景色.
Shape	获取或者设置警戒区域的形状.
UpperExtent	获取或者设置警戒区域中Y轴线数据坐标中的较上方的延伸区
Visible	获取或者设置警戒区域的可见性.

4.3.3 轴线集合编辑器

轴线集合编辑器用来编辑 C1Chart 中的 X,Y,Y2 轴线的属性.关于轴线属性的更多信息,请参见**轴线**(205 页).

访问轴线集合设计器

1. 在 C1Chart 控件上右键点击,然后从上下文菜单中选择 Properties.
2. 在属性窗口中,展开 **ChartArea** 节点,然后展开 **PlotArea** 节点,然后点击紧挨着 **Axes** 属性的 **ellipes**. 轴线集合编辑器将会出现.



在轴线区域集合编辑器中可用的属性

通过 **Axis Collection Editor**,在设计时可以使用以下的属性.或者在运行时可以通过 Axis 对象来访问这些属性.

轴线集合编辑器的属性:

成员	描述
Alignment	获取或者设置轴线展示时的文字对齐方式,从 ChartArea 中继承.
AnnoFormat	获取或者设置注解格式.
AnnoFormatString	获取或者设置用于手工格式化的注解格式化字符串.

AnnoMethod	获取或者设置注解的方法.
AnnotationRotation	获取或者设置轴线注解的正时针方向的旋转角度
AutoMajor	获取或者设置是否自动计算主要刻度线的值.
AutoMax	获取或者设置是否自动计算轴线的最大值.
AutoMin	获取或者设置是否自动计算轴线的最小值
AutoMinor	获取或者设置是否自动计算次要刻度线的值.
AutoOrigin	获取或者设置是否自动计算轴线的起点.
Compass	获取或者设置轴线的一般位置,X 可能被设置为南/北.Y 可能被设置为东/西.
Font	获取或者设置数据点符号的轮廓宽度.
ForeColor	获取或者设置前景色,继承自 ChartArea.
GridMajor.AutoSpace	获取或者设置自动格网线控件计算.
GridMajor.Color	获取或者设置线的颜色.
GridMajor.Pattern	获取或者设置线的样式.
GridMajor.Spacing	获取或者设置数据坐标单位中的格网线空间.
GridMajor.Thickness	获取或者设置线的厚度.
GridMinor.AutoSpace	获取或者设置自动格网线控件计算
GridMinor.Color	获取或者设置线的颜色.
GridMinor.Pattern	获取或者设置线的样式.
GridMinor.Spacing	获取或者设置数据坐标单位中的格网线空间.
GridMinor.Thickness	获取或者设置线的厚度
GridMinor.Visible	获取或者设置格网线的可见性.
IsLogarithmic	获取或者设置轴线是否是对数的.
LogarithmicBase	获取或者设置对数刻度使用的基准.小于或者等于 1 的值代表自然对数.
Max	获取或者设置轴线的最大值.
Min	获取或者设置轴线的最小值
Name	获取轴线的名称(X,Y,Y2).
NoAnnoOverlap	获取或者设置轴线是否允许溢出.
OnTop	获取或者设置轴线和格外线是否显示在图表图像的上方.
Origin	获取或者设置轴线的起点.
Reversed	获取或者设置轴线是普通的还是翻转过的.(上升的或者下降的)
Rotation	获取或者设置轴线的文本标题的选择定位.
ScrollBar	获取滚动条.
ScrollBar.Alignment	获取或者设置滚动条相对于绘制区域的对齐方式.
ScrollBar.Anchored	指示轴线滚动条是固定在绘制区域的边界还是随着轴

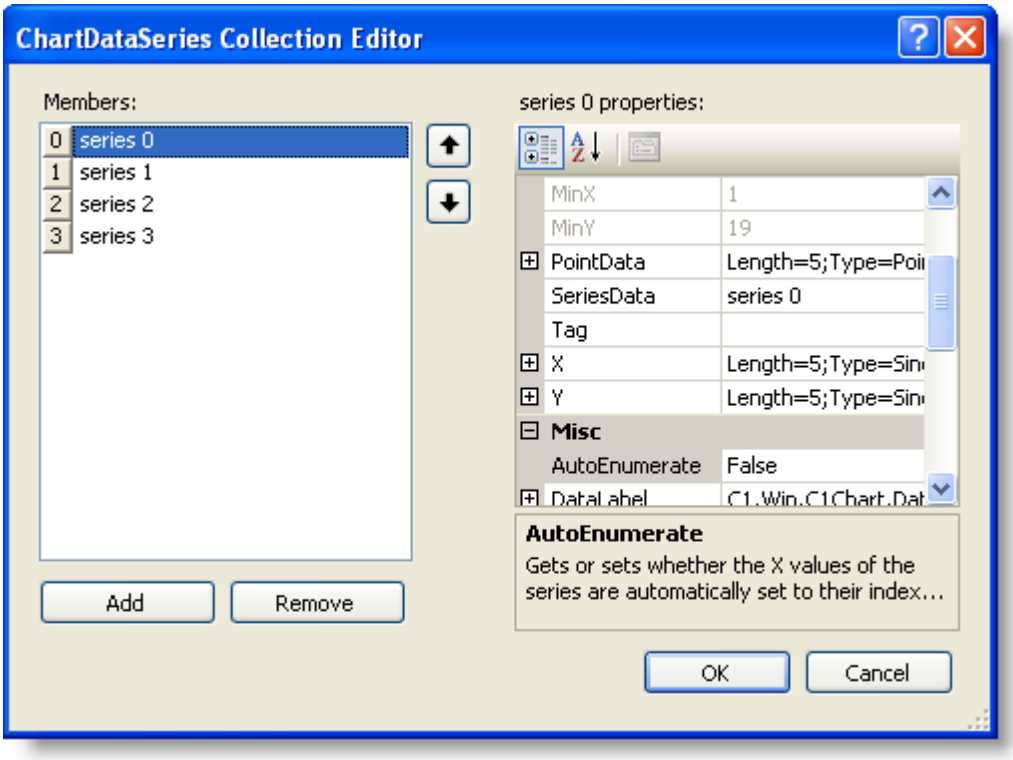
	线起点移动.
ScrollBar.Appearance	获取或者设置滚动条的外观.
ScrollBar.Buttons	获取或者设置滚动条的按钮.
ScrollBar.Max	获取或者设置滚动条位置的最大值.
ScrollBar.Min	获取或者设置滚动条位置的最小值
ScrollBar.Scale	获取或者设置滚动条的缩放.
ScrollBar.ScaleMenu	获取或者设置当用户点击缩放按钮时出现的自定义上下文菜单.
ScrollBar.ScaleMenuItems	获取缩放菜单项目的集合.
ScrollBar.ScrollKeys	获取或者设置滚动条会相应的按键.
ScrollBar.Size	获取或者设置滚动条的大小.
ScrollBar.Step	获取或者设置滚动条位置转换需要的步骤.
ScrollBar.Value	获取或者设置滚动条上滚动方框当前的相当位置.
ScrollBar.Visible	获取或者设置滚动条的可见性.
Text	获取或者设置轴线的文字性标题.
Thickness	获取或者设置轴线以像素为单位的厚度.
TickFactorMajor	获取或者设置主要刻度线的长度的积分因子.
TickFactorMinor	获取或者设置次要刻度线的长度的积分因子.
TickGauge	获取或者设置主要刻度线间由刻度线划分的近似间隔数.
TickLabels	获取或者设置注解标签相对于轴线的位置(目前还没有实现).
TickMajor	获取或者设置主要刻度线的类型
TickMinor	获取或者设置次要刻度线的类型
TooltipText	获取或者设置提示信息的内容.
UnitMajor	获取或者设置主要刻度间的间隔单位.设置该属性将会关闭 AutoMajor.
UnitMinor	获取或者设置次要刻度间的间隔单位.设置该属性将会关闭 AutoMinor.
ValueLabels	获取轴线的值标签集合.
VerticalText	获取或者设置标签文字是否垂直显示.
Visible	获取或者设置轴线的可见性.

4.3.4 图表数据序列集合编辑器

图表数据序列集合编辑器让用户能够添加和删除数据序列,同时还能编辑它们的属性.关于图表数据序列的更多信息,请参见定义图表数据序列对象(159 页).

访问图表数据序列集合编辑器

1. 在 C1Chart 控件上右键点击,然后从上下文菜单中选择 Properties.
2. 在属性窗口中,展开 ChartGroup 节点,然后展开 Group0 或者 Group1 节点.
3. 展开 ChartData 节点,然后点击紧挨着 SeriesList 属性的 ellipes.图表数据序列集合编辑器将会出现.



在图表数据序列集合编辑器中可用的属性

下表展示了 **ChartDataSeries Collection Editor** 在设计时可以使用的属性的名字和描述.或者在运行时可以通过 ChartDataSeries 对象来访问这些属性.

成员	描述
ChartDataSeries.Length	获取序列上数据点的个数.
ChartDataSeries.MaxX	返回数据序列上最大的 X 值.
ChartDataSeries.MaxY	返回数据序列上最大的 Y 值.
ChartDataSeries.MinX	返回数据序列上最小的 X 值.
ChartDataSeries.MinY	返回数据序列上最小的 Y 值.
ChartDataSeries.PointData	获取管理点数据的 ChartDataArray 对象.(同时含有 X 和 Y 的数据).
StatisticalData.PropertyGridEnabled	获取或者设置统计计算是否在属性网格中使用反射来展示.
StatisticalData.DataStatus	获取一个用来指示是否含有用于统计计算的数据的字符串.
ChartDataSeries.Tag	获取或者设置序列的用户数据.
ChartDataArray.DataField	被限制的数据字段.
ChartDataArray.DataType	获取或者设置外部数据类型.

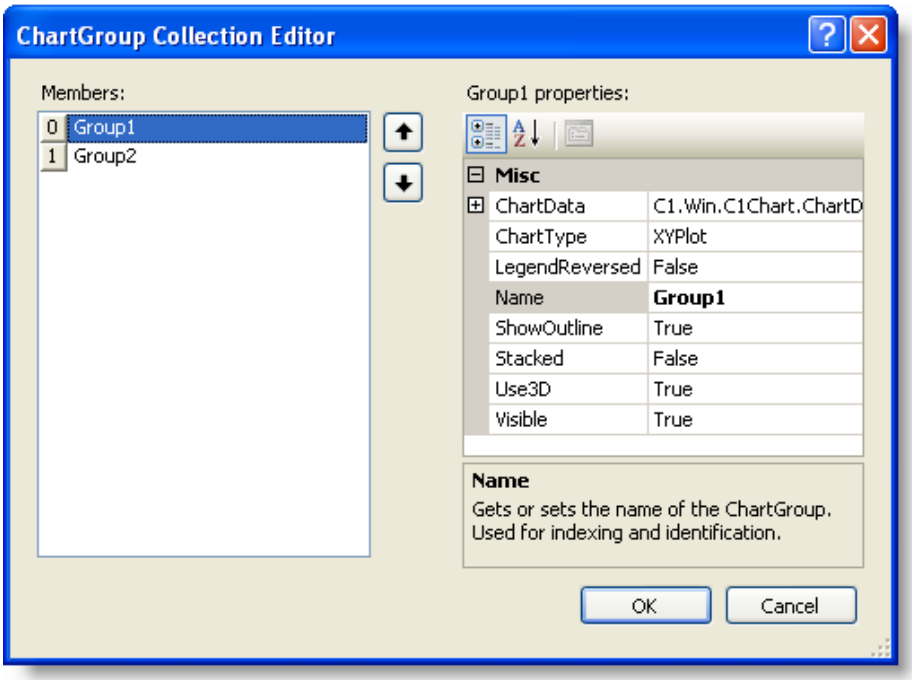
ChartData.Hole	获取数据洞的值.
ChartDataSeries.Length	获取或者设置 ChartDataArray 的元素个数.

4.3.5 图表组集合编辑器

图表组集合编辑器允许用户设置或者修改 Group1 或者 Group2 的属性。关于图表组的更多信息，请参见定义图表组对象（157 页）。

访问图表组集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartGroup** 节点,然后点击紧挨着 **ChartGroupCollection** 属性的 ellipes.图表组集合编辑器将会出现.



在图表组集合编辑器中可用的属性

下表展示了 **ChartGroup Collection Editor** 在设计时可以使用的属性的名字和描述.或者在运行时可以通过 ChartGroup 对象来访问这些属性.

成员	**描述**
ChartData	获取序列上数据点的个数.
ChartData.FunctionsList	返回数据序列上最大的 X 值.
ChartData.HighLight	返回数据序列上最大的 Y 值.
DataHighlight.Activation	返回数据序列点数据数组上最小的 X 值.
DataHighlight.Appearance	返回数据序列点数据数组上最小的 Y 值.
ChartDataSeries.PointData	获取管理点数据的 ChartDataArray 对象.(同时含有 X 和 Y 的数据).
StatisticalData.PropertyGridEnabled	获取或者设置统计计算是否在属性网格中使用反射来展示.

StatisticalData.DataStatus	获取一个用来指示是否含有用于统计计算的数据的字符串.
ChartDataSeries.Tag	获取或者设置序列的用户数据.
ChartDataArray.DataField	被限制的数据字段.
ChartDataArray.DataType	获取或者设置外部数据类型.
ChartData.Hole	获取数据洞的值.
ChartDataSeries.Length	获取或者设置 ChartDataArray 的元素个数.

4.3.6 基于函数的集合编辑器

基于函数的集合编辑器是用来在设计时添加或者修改以绘制数据为目的的函数。关于使用函数来绘制图表的更多信息，请参见绘制函数（168 页）。

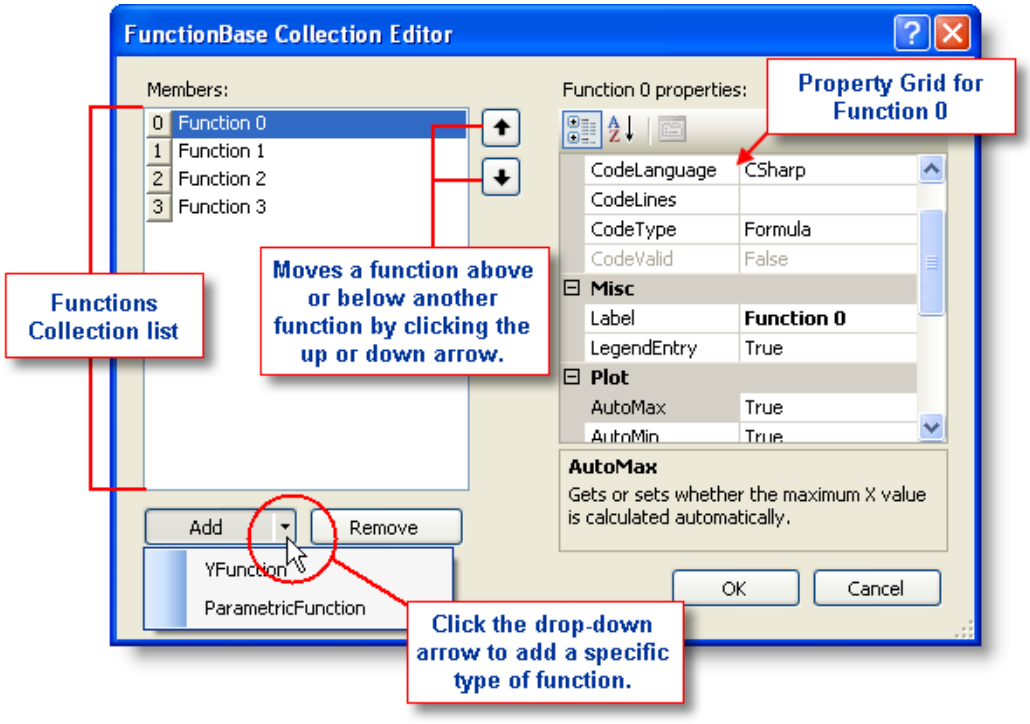
访问基于函数的集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartGroup** 节点,
3. 然后展开 **Group0** 节点，然后再展开 **ChartData** 节点。
4. 选择 **FunctionList** 属性，并点击 **ellipes** 按钮.

基于函数的集合编辑器将会出现.

基于函数的集合编辑器的基础部分

下面的插图描述了 **FunctionBase Collection Editor** 中的元素.



下表展示了 **FunctionBase Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 FunctionBase 对象来访问这些属性.

在基于函数的集合编辑器中可用的属性

Code 属性指定了函数代码的类型和语言,还有代码的多行的展示.Code 属性和它们的描述

如下:

成员	描述
CodeErrors	获取所有的编译错误的文字描述.
CodeLanguage	获取或者设置使用 VB 还是 C#作为编程语言.用户可以通过点击代码语言文本框的下列按钮来指定使用哪种语言.
CodeLines	获取或者设置方法代码的多行展示.通过点击 ellipsis 按钮您可以在集合字符串编辑器中的列表框中输入代码。
CodeType	获取或者设置代码将会被编辑成公式，方法，或者一个完整的编译单元。
CodeValid	获取函数编译是否成功。

基于函数的集合编辑器的其他属性

函数的其它属性如下：

成员	描述
Label	获取或者设置函数的标签。
LegendEntry	获取或者设置函数是否被显示在图例中。

基于函数的集合编辑器的绘制属性

绘制属性用于修改您的函数中使用的图表数据序列绘制的样式或者功能。绘制属性和它们的描述如下：

成员	描述
AutoMax	获取或者设置最大 X 值是否自动计算。
AutoMin	获取或者设置最小 X 值是否自动计算。
LineStyle	获取或者设置用于绘制函数的线的样式。
MaxX	获取或者设置函数绘制时的最大 X 值。
MinY	获取或者设置函数绘制时的最大 X 值。
PlotLinesMethod	获取或者设置函数绘制时使用的方法。
PlotNumPoints	获取或者设置函数绘制时使用的点的数目。
PlotOverData	获取或者设置函数是绘制在数据序列的上面还是后面。
Visible	获取或者设置函数绘制的可见性。

4.3.7 标签集合编辑器

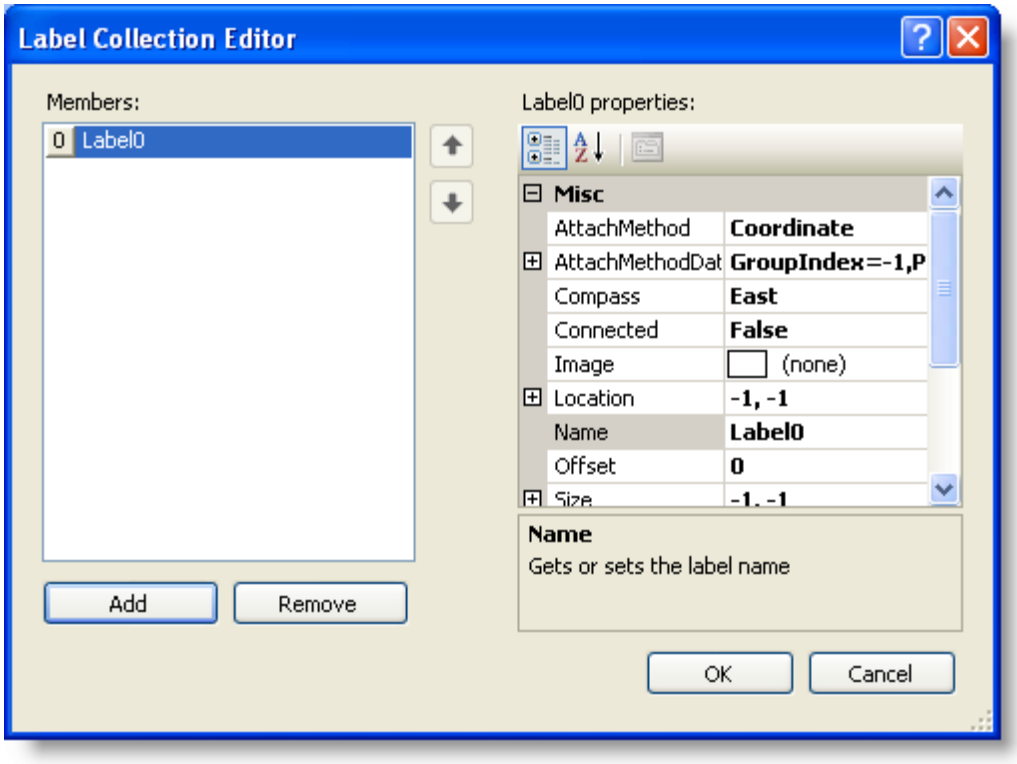
标签集合编辑器用来增加和编辑数据序列上的图表标签。图表标签用来指示在数据序列中的重要的数据点。关于如何使用图表标签的更多信息，请参见图表标签（198 页）。

访问标签集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 Properties.
2. 在属性窗口中,展开 ChartLabels 节点,然后点击紧挨着 LabelsCollection 属性的

ellipes 按钮.

标签集合编辑器将会出现.



在标签集合编辑器中可用的属性

下表展示了 **Label Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 ChartLabels 对象来访问这些属性.

成员	**描述**
AttachMethod	获取或者设置标签的附件方法。
AttachMethodData	获取或者设置填充颜色。
GroupIndex	当标签的 AttachMethod 属性指定 DataIndex 附属时，获取或者设置附加到一个标签上的数据点的组索引。
PointIndex	当标签的 AttachMethod 属性指定 DataIndex 附属时，获取或者设置附加到一个标签上的数据点的点索引。
SeriesIndex	当标签的 AttachMethod 属性指定 Coordinate 或者 DataCoordinate 附属时，获取或者设置附加到一个标签上的数据点的序列索引。
X	当标签的 AttachMethod 属性指定 Coordinate 或者 DataCoordinate 附属时，获取或者设置 X 坐标（数据或者用户工作区）.
Y	当标签的 AttachMethod 属性指定 Coordinate 或者 DataCoordinate 附属时，获取或者设置 Y 坐标（数据或者用户工作区）.
Compass	获取或者设置标签相对于其特定位置的方位。

Connected	获取或者设置一个连接线是否绘制到一个相关联的数据点。
Image	获取或者设置图片。
Location	获取图表在控件用户工作区中标签的位置.
Name	获取标签的名字。
Offset	获取或者设置从一个关联的数据点的以像素为单位的偏移距离。
Label.RotationOverride	这个属性允许指定标签连接点的正时针旋转角度。这个属性重载了 label Style 对象的 RotationEnum，同时不适用于 Radial 或者 RadialText 罗盘值。
Size	获取或者设置在图表控件客户工作区域中的标签大小。
SizeDefault	获取或者设置标签的默认大小。
Style	获取标签的 Style 对象.
AutoWrap	获取或者设置文字是否自动换行。
BackColor	获取或者设置背景色。
BackColor2	获取获取设置渐进或者影线背景色
Border	获取 border 对象。
BorderStyle	获取或者设置边框样式。
Border.Color	获取或者设置边框颜色。
Rounding	获取控制边角的环绕的 Rounding 对象.
Thickness	获取或者设置边框厚度.
Font	获取或者设置 font 对象.
ForeColor	获取或者设置前景色.
GradientStyle	定义背景的渐进填充的颜色。
HatchStyle	定义背景影线填充的样式.
HorizontalAlignment	获取或者设置文字的水平对齐.
ImageAlignment	获取或者设置图片对齐.
Opaque	获取或者设置背景的不透明性。
Rotation	获取或者设置文字定位。
VerticalAlignment	获取或者设置文字的垂直对齐.
Text	获取或者设置标签的文字.
TooltipText	获取或者设置提示信息的文字.
Visible	获取或者设置标签可见性.

4.3.8 点样式集合编辑器

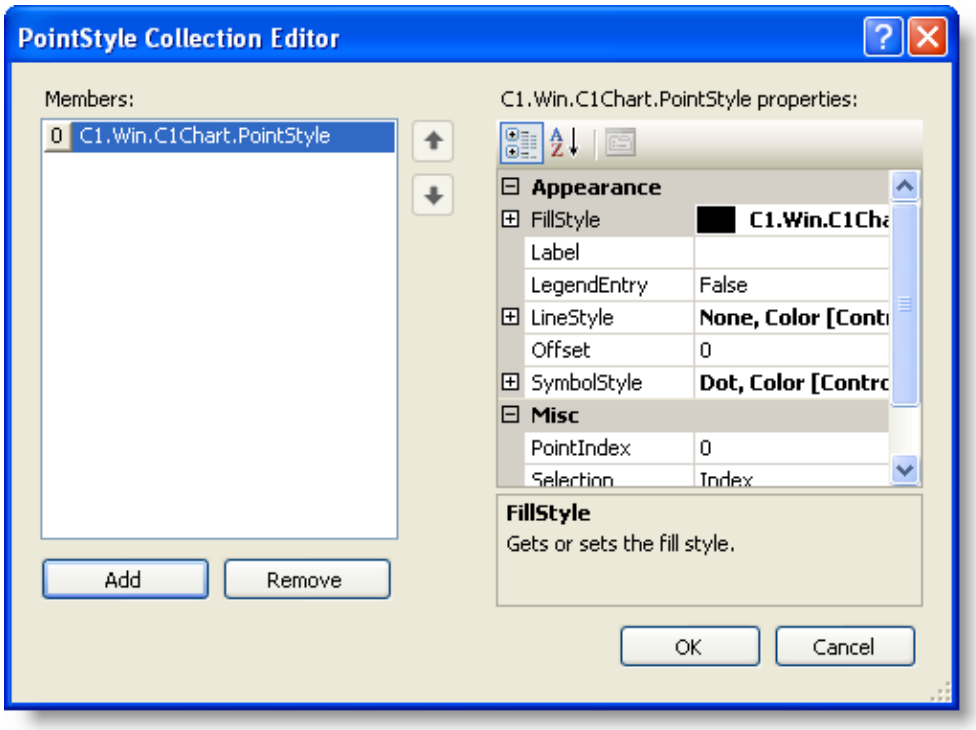
点样式集合编辑器是用来在设计时添加或者修改图表上特定数据点的点样式的.关于如何使用点样式的更多信息,请参见使用点样式来工作(182 页).

访问点样式集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartGroup** 节点,然后展开 **Group0** 或者 **Group1** 节点.

3. 展开 **ChartData** 节点,然后点击紧挨着 **PonitStyleList** 属性的 **ellipes**
4. 点击 Add 按钮来在集合编辑器上添加一个点样式.

点样式的属性会出现在**点样式集合编辑器**中的属性网格中.



在点样式集合编辑器中可用的属性

下表展示了 **PointStyle Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 PointStyle 对象来访问这些属性.

点样式集合编辑器中的外观属性

成员	描述
FillType	获取或者设置填充类型.
Color1	获取或者设置填充颜色.
Alpha	获取或者设置填充 alpha 值(透明性).
Label	获取或者设置点样式标签.
LegendEntry	获取或者设置点样式是否显示在图例中.
Color	获取或者设置线的颜色.
Pattern	获取或者设置线的样式.
Thickness	获取或者设置线的厚度.
Offset	获取或者设置偏移量来适应图表.
Color	获取或者设置数据点符号的颜色.
OutlineColor	获取或者设置数据点符号的轮廓线颜色.
OutlineWidth	获取或者设置数据点符号的轮廓线宽度.
Shape	获取或者设置数据点符号的形状.

Size	获取或者设置数据点符号的大小.

点样式集合编辑器中的其它属性

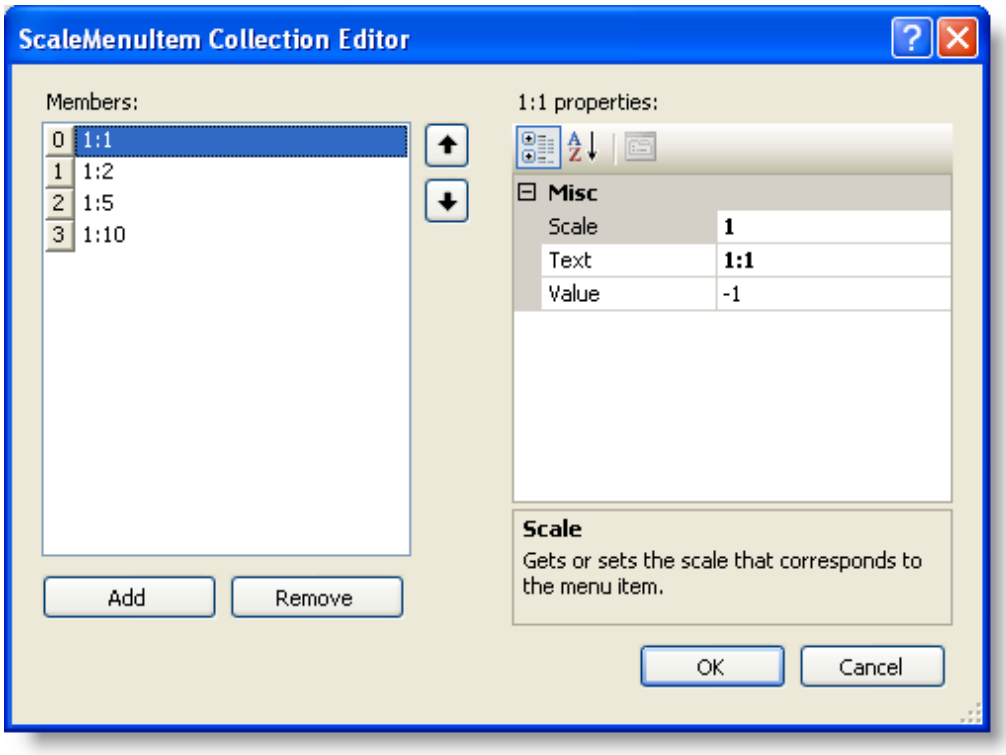
成员	描述
PointIndex	获取或者设置附加在点样式上的数据点的索引.
Selection	获取或者设置点样式的选中方法.
SeriesIndex	获取或者设置附加在点样式上的数据序列的索引.

4.3.9 缩放菜单集合编辑器

缩放菜单集合编辑器是用来编辑缩放菜单集合编辑器上的属性的.

访问缩放菜单集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartArea** 节点,然后展开 AxisX,AxisY 或者 AxisY2 节点.然后展开 **ScrollBar** 节点.
3. 点击紧挨着 **ScaleMenuItems** 属性的 **ellipes** 按钮.**缩放菜单集合编辑器**将会出现.



在缩放菜单集合编辑器中可用的属性

下表展示了 **ScaleMenuItems Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 ScaleMenuItem 对象来访问这些属性.

成员	描述
Scale	获取或者设置菜单条目对应的缩放.
Text	获取或者设置菜单条目的文字.

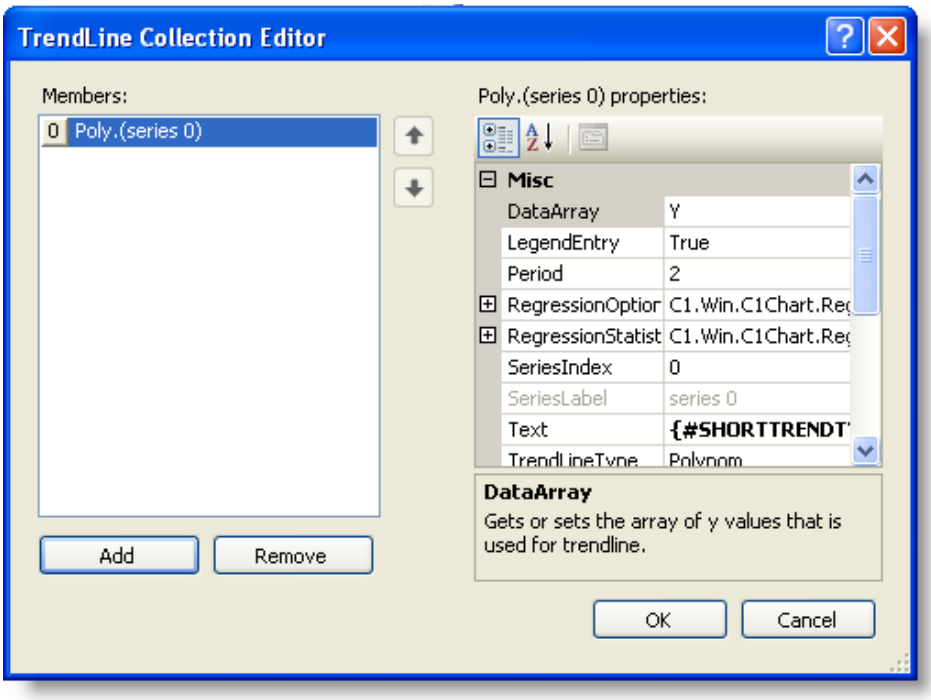
Value	获取或者设置菜单条目对应的值.

4.3.10 渐近线集合编辑器

渐近线集合编辑器是用来在设计时添加或者修改渐近线,它是用来模拟功能发展渐进的数据的.关于渐近线的更多信息,请参见使用渐近线进行工作(174 页).

访问渐近线集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartGroup** 节点,然后展开 Group0 或者 Group1 节点.
3. 展开 **ChartData** 节点.点击紧挨着 **TrendsLine** 属性的 **ellipes** 按钮.渐近线集合编辑器将会出现.



在渐近线集合编辑器中可用的属性

下表展示了 **Trendline Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 TrendLine 对象来访问这些属性.

渐近线集合编辑器的一些属性

成员	描述
DataArray	获取或者设置渐近线使用的 Y 值的数组.
LegendEntry	获取或者设置渐近线是否显示在图例中.
Period	获取或者设置渐近线移动平均值的取值区间.
RegressionOptions	获取回归计算的选项.
NumTerms	获取或者设置回归方程式的条目数量,仅对多项式渐近线和傅里叶渐近线可用.
UseYIntercept	获取或者设置渐近线是否使用 Yintercept 属性.仅对多项式渐近线可用.

YIntercept	获取或者设置当渐近线截取到 x=0 线时的 Y 值.仅对多项式渐近线可用.
Coeffs	获取回归方程式的系数.
DF	获取自由度.
F	获取 F-observed(F-statistic) 值.
Rsq	获取测量的系数(R-squared).
Sey	获取 Y 数值的标准错误.
Sse	获取由于错误导致的校验的总数.
Ssr	获取由于回归导致的校验的总数.
SeriesIndex	获取或者设置用于渐近线绘制的数据序列的索引.
SeriesLabel	获取用于渐近线绘制的数据序列标签.
Text	获取或者设置显示在图例上的文字.
TrendLineType	获取或者设置渐近线的类型.

渐近线集合编辑器的绘制属性

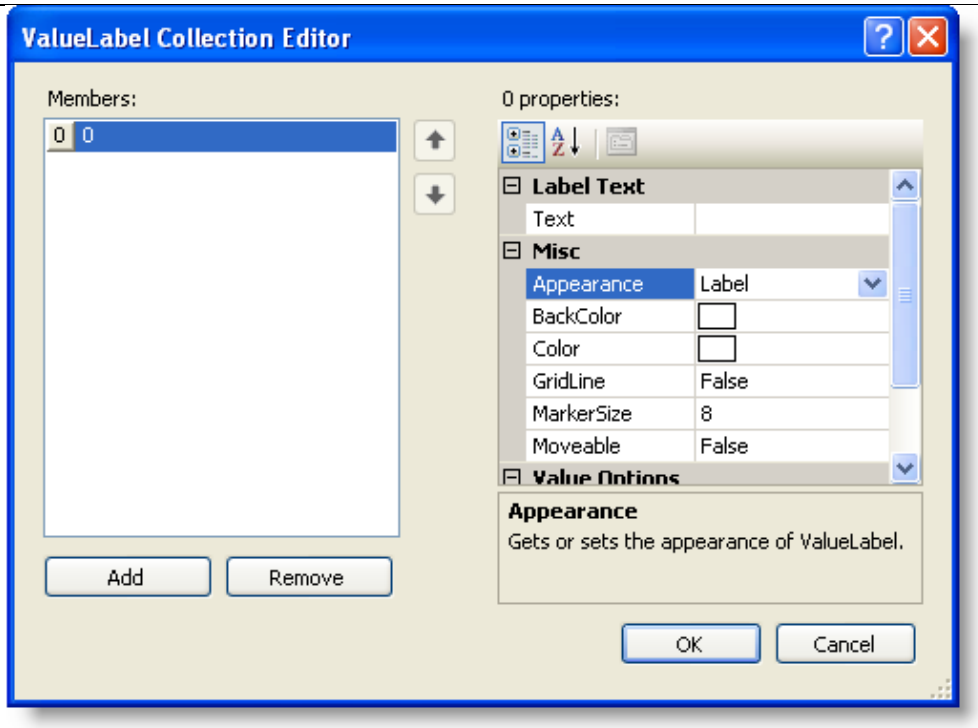
成员	**描述**
ForecastBackward	获取或者设置渐近线超过序列数据最小值的程度.
ForecastForward	获取或者设置渐近线超过序列数据最大值的程度.
LineStyle	获取或者设置用于绘制渐近线的线.
Color	获取或者设置线的颜色.
Pattern	获取或者设置线的样式.
Thickness	获取或者设置线的厚度.
PlotLinesMethod	获取或者设置渐近线绘制的方法.
PlotNumPoints	获取或者设置渐近线绘制的点的数量.
PlotOverData	获取或者设置渐近线是否绘制在数据的前面还是后面.
Visible	获取或者设置渐近线是否可见.

4.3.11 值标签集合编辑器

值标签集合编辑器是用来在设计时添加或者修改 ValueLabels 类中属性的.值标签显示了定义在一个特殊的轴线坐标上的文字.关于值标签的更多信息,请参见值标签注解(222 页).

访问值标签集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **ChartArea** 节点,然后展开 AxisX,AxisY 或者 AxisY2 节点.然后点击紧挨着 **ValueLabels** 属性的 **ellipes** 按钮.值标签集合编辑器将会出现.



在值标签集合编辑器中可用的属性

下表展示了 **Valuelabel Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 ValueLabel 对象来访问这些属性.

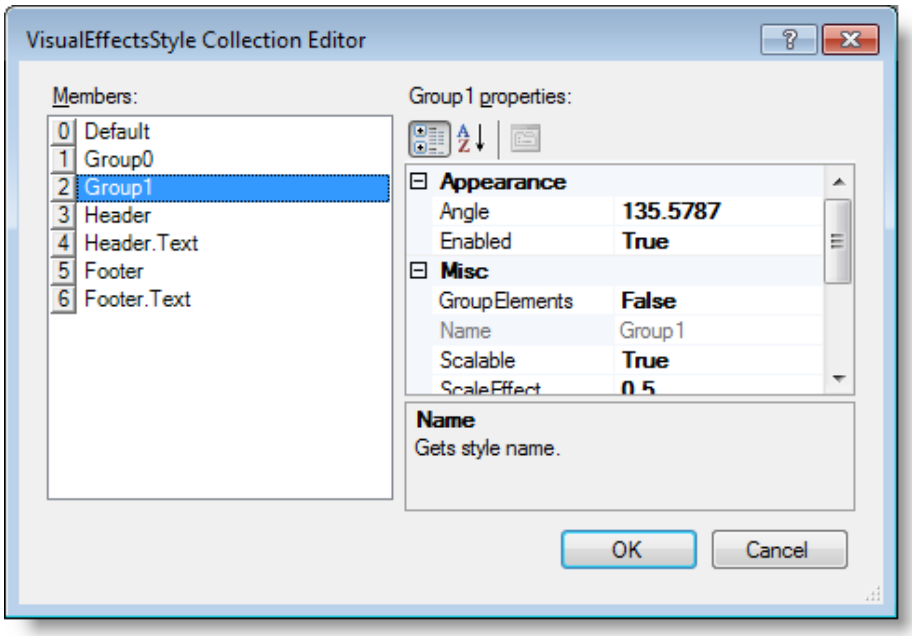
成员	描述
Text	获取或者设置在值标签上显示的文字.
Appearance	获取或者设置值标签的外观.
BackColor	获取或者设置值标签的背景色.
Color	获取或者设置值标签的颜色.
Gridline	获取或者设置一个指定了一个网格线是否显示在值标签中的标志值.
MarkerSize	获取或者设置值标签的标记大小.
Moveable	获取或者设置一个指示了值标签是否能够被用户拖拽的标志值.
DateTimeValue	获取或者设置值标签使用一个日期值替代的轴线值.
NumericValue	获取或者设置值标签使用一个日期值替代的轴线值.

4.3.12 视觉效果集合编辑器

视觉效果集合编辑器是用来编辑视觉效果属性的.另外一种用来编辑视觉效果样式的方法是使用视觉效果设计器.关于视觉效果设计器的更多信息,请参见视觉效果设计器(231 页).

访问视觉效果集合编辑器

1. 在 **C1Chart** 控件上右键点击,然后从上下文菜单中选择 **Properties.**
2. 在属性窗口中,展开 **VisualEffects** 节点,然后点击紧挨着 **Styles** 属性的 **ellipes** 按钮. 视觉效果集合编辑器将会出现.



在视觉效果集合编辑器中可用的属性

下表展示了 **VisualEffectsStyle Collection Editor** 在设计时可以使用的属性.或者在运行时可以通过 VisualEffectsStyle, ColorOptions, EdgeStyle, LightStyle 对象来访问这些属性.

成员	描述
Angle	获取或者设置在绘制时使用的角度.
Enabled	启用效果.
GroupElements	获取或者设置一个用来指定图形元素在绘制时能够被分组的标志值.
Scalable	获取或者设置一个用来指定元素的大小是否取决于控件大小的标志量.
ScaleEffect	获取或者设置当Scalable属性为True时用来度量元素的度量因子.
Colors	获取式样的颜色选项.
Brightness	获取或者设置明亮度校正值.
HueShift	获取或者设置色彩偏移,其取值范围为从0到359.
Saturation	获取或者设置饱和度调整,其取值范围为从-100到100.
Edge	获取样式的边框设置.
Rounding	获取或者设置矩形元素的近似半径.
Width	获取或者设置元素边框的宽度.当Scalable属性设置为true时,它使用相对单位进行测量,否则,使

	用像素.
Light	获取样式的光源设置.
Focus	获取或者设置光源的位置.
Gradient	获取或者设置光渐进的样式.
Intensity	获取或者设置光强度.
Scale	获取或者设置光缩放,当缩放小于一个光源模式时,它会在元素上重复.
Shape	获取或者设置光形状.
Shift	获取或者设置光移位.
Size	获取或者设置光点的大小.

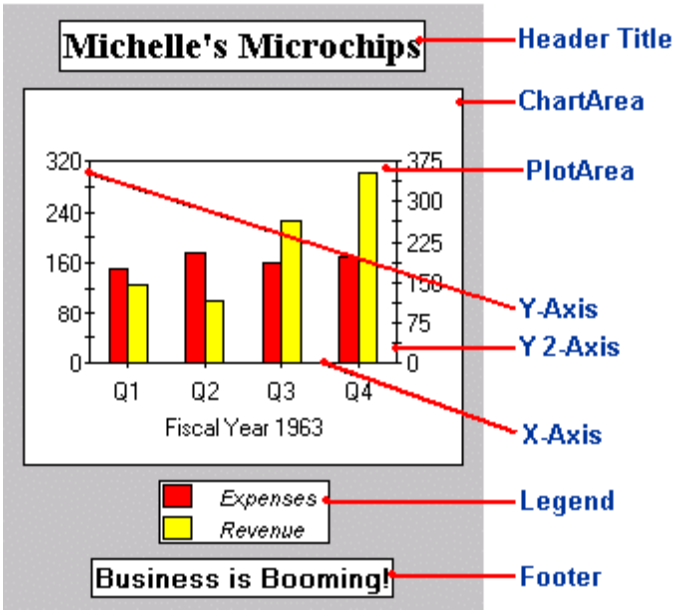
5. 图表基础

C1Chart 控件含有一个丰富的对象模型来让您可以创建大量的图表类型,并控制它们的外观和行为.

本章节通过描述图表中的主要元素和主要属性之间的关联来向您介绍 C1Chart 的对象模型. 在阅读完本章节后,您将对C1Chart对象模型中的主要元素有一个很好的理解,这将帮助您更好地使用 C1Chart 来创建您的控件.后续的章节将会深入到更深的细节中,覆盖编程模型中的剩余部分.

2D 图表技术

下表展示了用于描述图表元素的术语:

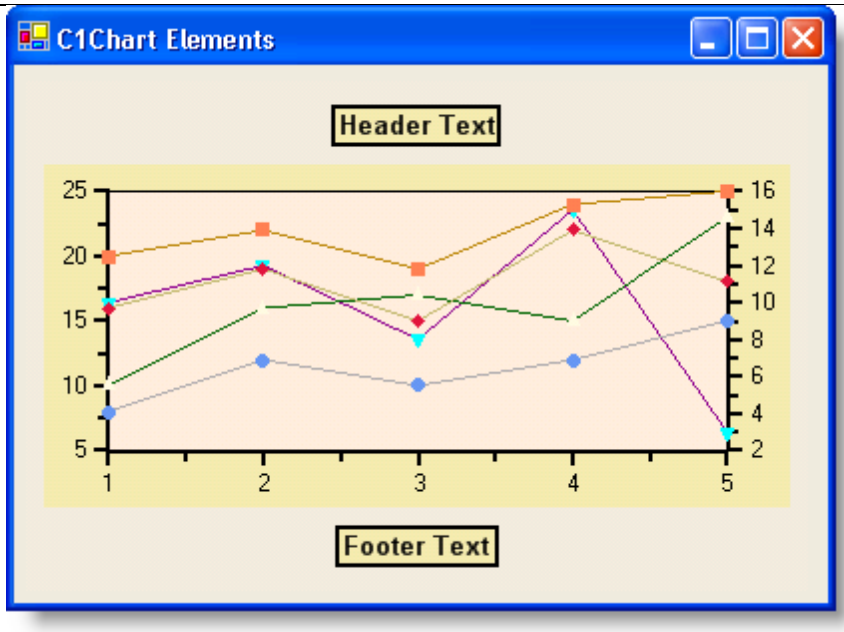


注意:饼状图表含有除了轴线之外的相同元素.

后续的主题将会描述在处理图像时的元素,和每个元素相关的主要属性.

5.1 定义表头和页尾元素

这里展示了表头和页尾元素是如何展示在图表上的.



表头和页尾是用来展示图表的描述性信息的.它们由 Header 和 Footer 属性控制.这两个属性都返回一个 Title 对象,以下是该对象含有一些的主要属性:

属性	描述
Text	在标题(表头和页尾)上显示的文字.
Style	用来设置标题的字体,特性,颜色,边框的属性.
Compass	决定标题的位置.
Visible	决定标题是否可见.

C1Chart 自动处理标题的大小和位置,基于它们的内容和 Compass 属性的设置.标题对象位置可用使用 SizeDefault 和 LocationDefault 属性来进行自定义(如果是负值,会激活自动的位置和大小设置).

创建表头和页尾元素

图表的表头和页尾元素可以通过它们的 Title 对象以编程方式创建,或者可以通过使用图表属性窗口,图表属性编辑器,图表智能设计器来在设计时创建.

最简单的创建方式是使用图表智能设计器,关于使用图表智能设计器创建表头和页尾的更多信息,请参见添加一个图表表头(330 页)或者添加一个图表页尾(329 页).

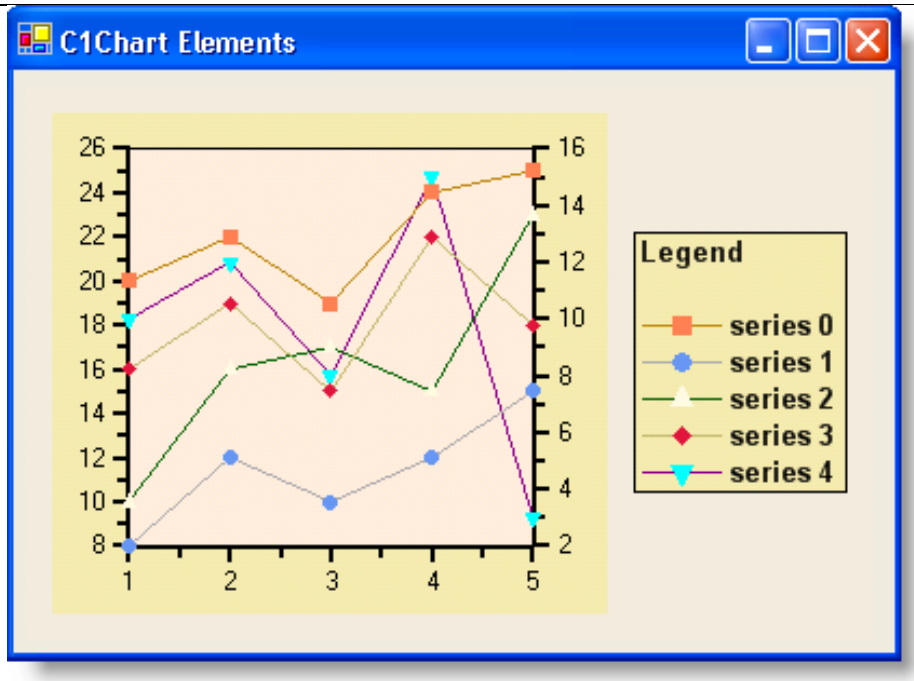
自定义表头和页尾元素

可以通过 Title 的属性来自定义表头和页尾的文字,对齐,位置,边框,颜色,和字体.另外,可以使用视觉效果设计器来给表头和页尾元素添加光源模式,阴影,自定义颜色.

请参见自定义图表元素(231 页)来获取关于增强 2D 图表元素的属性和工具的更多信息.关于定位表头元素的更多信息,请参见显示图表图例和图表表头(297 页).

5.2 定义图例元素

下图显示了图表中的图例元素.



图例元素显示了图表中每一个数据序列的信息.图表图例显示了物理颜色和数据序列的映射.图例由 Legend 属性控制,该属性返回一个 Legend 对象,该对象含有以下的主要属性:

属性	描述
Text	显示在图例标题上的文字.
Style	用于设置图例的字体,定位,颜色,和边框信息.
Compass	决定图例的位置.
Visible	决定图例是否可见.
Orientation	决定图例元素是横向还是竖向显示.
Reversed	控制ChartGroup显示在图例上的顺序,一个ChartGroup元素显示在图例上的顺序是由每一个CahrtGroup对象的LegendReversed属性来决定的.

C1Chart 自动处理图例的大小和位置,基于它们的内容和 Compass 属性的设置.标题对象位置可用使用 SizeDefault 和 LocationDefault 属性来自定义(如果是负值,会激活自动的位置和大小设置).

请注意 Legend 属性并不控制每一个图例元素的文字,它由每一个序列附件的标签决定,例如,标签含有文字, LegendEntry 属性决定序列标签是否显示在图例上.

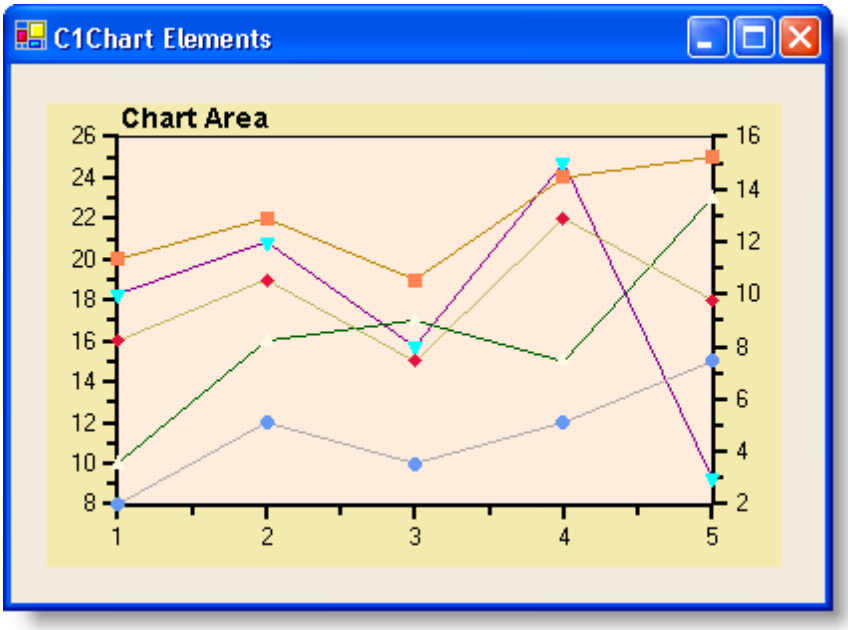
创建图例元素

图表的图例元素可以通过它们的legend对象以编程方式创建,或者可以通过使用图表属性窗口,图表属性编辑器,图表智能设计器来在设计时创建.

最简单的创建方式是使用图表智能设计器,关于使用图表智能设计器创建图例的更多信息,请参见添加一个图表图例(331 页),关于以编程方式定位图表图例的更多信息,请参见显示图表的图例和表头(297 页).

5.3 定义图表区域元素

下图显示了 C1Chart 中的图表区域元素.

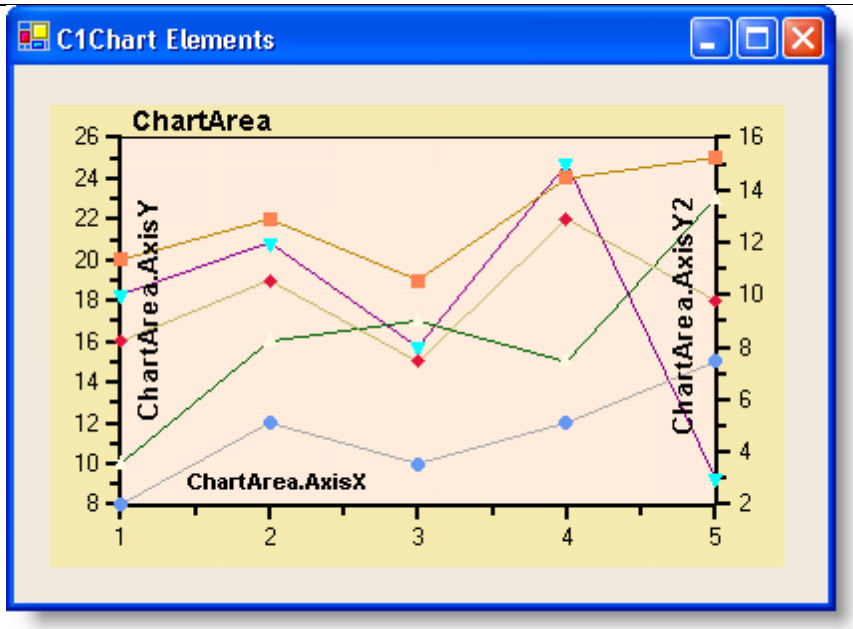


ChartArea 对象代表那些包含数据的图表区域(不包括标题和图例,但是包括轴线),ChartArea 属性返回一个 Area 对象,该对象含有以下主要的属性:

属性	描述
AxisX, AxisY, AxisY2	这些属性中的每一个都会返回一个 Axis 对象,该对象让您能够自定义轴线的外观.Axis 对象会在下一个章节轴线对象(64 页)中进行解释,并且会在轴线(205 页)的主题中做进一步的介绍.
Inverted	允许您以 90 度旋转图表(当需要创建水平柱状图时将很有用),更多的信息,请参见翻转和旋转图表轴线(219 页).
Margins	返回一个 Margin 对象,该对象让您指定图表区域和绘制区域的距离.轴线标签在这个区域被显示.
PlotArea	返回一个 PlotArea 对象,该对象控制轴线内的区域的外观.关于此的更多信息,请参见绘制区域(227 页).
Style	包括设置图表区域的颜色和边框的属性.

5.3.1 轴线对象

下图展示了 C1Chart 的 X 和 Y 轴线:



多数的图表含有两个轴线,X 和 Y.例外的是饼状图(没有轴线),和含有第二个 Y 轴线的图表(Y2,所以含有三个轴线).

这些轴线被 ChartArea 的子属性代表: ChartArea.AxisX, ChartArea.AxisY,和 AxisY2.这些属性中的每一个都返回一个 Axis 对象,该对象含有以下的主要属性:

- 布局和样式属性

下面描述了在C1Chart控件中轴线的布局和式样属性.

属性	描述
Compass	允许您设置轴线的位置.例如,您可能想在数据的上面而不是下面显示X轴线.关于此的更多信息,请参见轴线位置(205页).
Font	设置在轴线上显示值时使用的字体.关于更多信息,请参见轴线外观.
ForeColor	设置用于展示轴线,刻度线,和值元素时使用的的颜色.关于更多信息,请参见轴线外观.
Reversed	允许您翻转轴线的方向.例如,您可以从上到下显示Y的值,关于此的更多信息,请参见翻转和旋转图表轴线(219页).
Text	设置一个紧挨着轴线显示的字符串(典型的应用场景是描述被轴线叙述的变量和单位).关于此的更多信息,请参见轴线标题和旋转(206页).
Rotation	设置文字字符串的方向.

- 注解属性

下面描述了在C1Chart控件中轴线的注解属性.

属性	描述
AnnoFormat	一个预先定义的格式集合,用来格式化紧挨着轴线显示的值.
AnnoFormatString	.NET格式化字符串是用来当AnnoFormat属性被设置为

	"NumericManual" 或者"DateManual"时用来格式化紧挨着轴线显示的值. 当AnnoFormat属性被设置为"NumericManual" 或者"DateManual"并且AnnoFormatString为空时,图表使用一个算法来寻址"最优"的格式化字符串.
AnnoMethod	决定哪些值紧挨着轴线显示.当设置为"Values"时,显示真实的序列值,当设置为"ValueLabels"时显示在ValueLables集合中的元素.
ValueLabels	一个文字/值的集合,当AnnoMethod设置为"ValueLabels"时显示紧挨着轴线.当您想在轴线中间显示字符串而不是数值时非常有用(例如,您可能在绘制产品的价格时,同时希望在X轴线上显示产品名称),关于此的更多信息,请参见值标签注解(222页).
AnnotationRotation	运行您旋转值,以让它们占用轴线上更少的空间.关于此的更多信息,请参见轴线注解旋转(226页).

- 缩放,刻度线和表格线属性

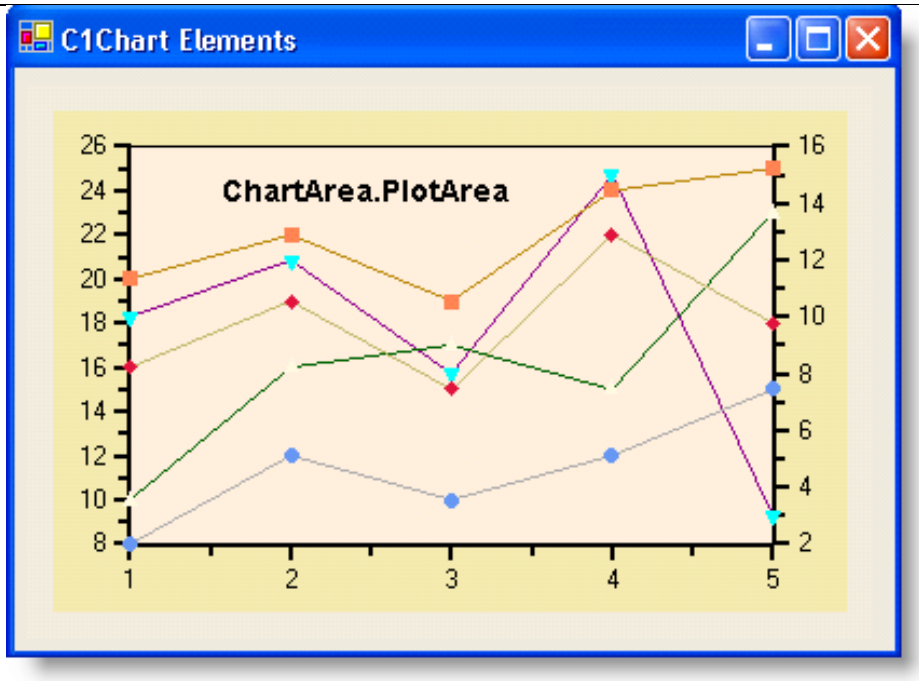
下面描述了在**C1Chart**控件中轴线的缩放,刻度线,网格线样式和功能属性.

属性	**描述**
AutoMin, AutoMax	决定了轴线的最大值和最小值是否应该被自动计算,关于此的更多信息,请参见轴线边界(209页).
Min, Max	设置轴线的最大值和最小值(当AutoMin和AutoMax被设置为False时).关于此的更多信息,请参见轴线边界(209页).
AutoMajor, AutoMinor	决定了主要和次要刻度线之间的间距是否应该被自动计算.
UnitMajor, UnitMinor	设置主要和次要刻度线之间的间距(当AutoMajor和AutoMinor属性被设置为False时).
GridMajor, GridMinor	返回一个ChartGridStyle对象,该对象含有控制在主要和次要刻度线间绘制垂直网格线的属性,关于此的更多信息,请参见轴线网格线(208页).
AutoOrigin	决定X轴是否应该根据Y轴自动计算位置.
Origin	设定X轴相对于Y轴的位置,X轴一般放置在图表的底部,这个属性允许您定位它,以使得它在一个给定的坐标上穿过Y轴.
TickFactorMajor	获取或者设置主要刻度线的整因子.
TickFactorMinor	获取或者设置次要刻度线的整因子.
TickGauge	获取或者设置主要刻度线之间以刻度线描述的大致距离数.

关于Axis对象的更多信息,请参见轴线(205页).

5.3.2 绘制区域对象

下图展示了**C1Chart**中的绘制区域对象



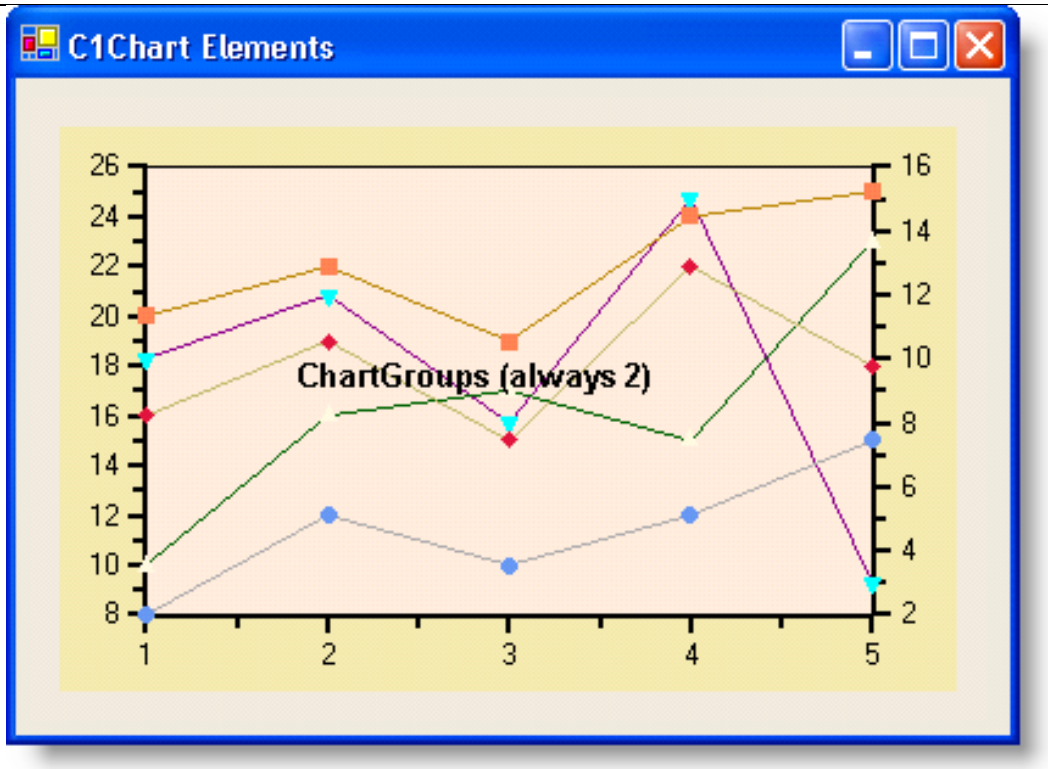
PlotArea 对象代表了用于显示数据序列的图表区域部分.PlotArea 类中最经常被使用的用于自定义该对象的属性如下:

属性	描述
BackColor	设置绘制区域的背景色.
Boxed	决定绘制区域是否被一个实线边框环绕.
ForeColor	设置环绕绘制区域的边框的颜色(当 Boxed 属性设置为 True 时).
View3D	返回一个 View3D 对象,该对象允许您给 2D 图表添加 3D 效果.

关于自定义绘制区域的更多信息,请参见绘制区域(227 页).

5.4 定义图表组对象

下图展示了 C1Chart 中的图表组元素.



这个属性返回一个含有两个图表组对象的集合(通常是 Group0 和 Group1,您不能在这个集合中进行添加或者删除动作).这些对象决定了图表类型并且包含将被用于绘制的数据.

第一个组(Group0)含有针对主 Y 轴线上被绘制的序列,第二个组(Group1)可以为空,或者它可以含有针对次要 Y 轴线上被绘制的序列.因为这里含有两个图表组对象,而且图表类型跟图表组有关系,您可以在同一个图表中混合使用两种图表类型.例如,您可以创建即作为条形图显示的序列,同时还含有作为线性图显示的序列.

ChartGroup 对象含有以下的主要属性:

属性	描述
ChartData	返回一个ChartData对象,该对象含有一个持有该组中所有数据的SeriesList属性.该属性在图表数据序列对象(68页)的主题中有更详细的描述.
ChartType	决定在绘制该组时使用的图表类型(这里含有很多种类的图表,您可以给每一个图表组使用不同的类型).关于可用的2D图表类型的信息,请参见特定的2D类型(80页).
ShowOutline	决定用来显示数据的图形化的元素 (图形图,区域,扇形扇区)是否应该使用由ForeColor属性指定的颜色来重点突出显示
Stacked	决定在序列被绘制时数据是否应该堆叠(通过添加Y值).

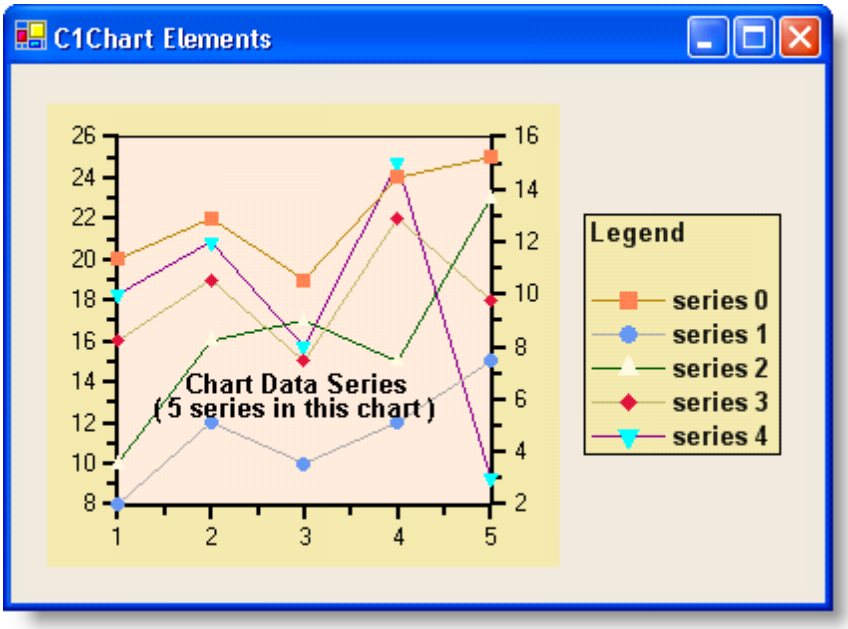
5.4.1 图表数据对象

ChartData 对象含有一些的主要属性:

属性	描述
FunctionsList	FunctionsList属性用来获取附加在当前ChartData对象上的函数集合对象.关于FunctionsList属性的更多信息,请参见定义ChartData对象(158页).
Hole	获取或者设置数据洞的值,关于此的更多信息,请参见指定的数据洞(167页).
PointStylesList	获取附加在当前ChartData对象上的PointStylesCollection对象,关于点样式的更多信息,请参见使用点样式进行工作(182页).
SeriesList	获取附加在当前ChartData对象上的DataSeriesCollection对象,下一个主题,图表数据序列对象(68页),将提供关于ChartData.SeriesList属性的更多信息.
TrendsList	获取附加在当前ChartData对象上的TrendLinesCollection对象,关于渐近线的更多信息,请参见使用渐近线进行工作(174页).

5.4.2 图表数据序列对象

下图展示了在 C1Chart 中显示五个图表数据序列的示例:



从一个开发人员的角度来看,这是在C1Chart中最重要的一个属性.所有其它的属性都可以在设计时使用图表向导,图表属性设计器,或者从预先定义好的图表布局文件中加载.在最常见的情况下,被绘制的数据时通过代码添加的,为了做这个工作,您需要使用 SeriesList 属性.

SeriesList 属性返回一个 ChartDataSeriesCollection 对象,该对象允许您在图表上添加或者删除序列,或者可以获取一个单独的序列. Adding Data Series 这个示例向您展示了以编程的方式添加数据序列的方法.

ChartDataSeries 对象含有以下的主要属性:

属性	描述
Display	决定了序列是否可见,和如何展示缺失值(数据洞).
Label	含有展示在图例上的文字(当 LegendEntry 属性设置为 True 时).
LegendEntry	决定了序列标签是否应该被显示在图例上.
LineStyle	包含了以下的属性:用来决定序列显示时的颜色,厚度,和样式(颜色被用在线条,区域,条形图,和扇形切片上),关于此的更多信息,请参见**序列的线条和符号样式**(240 页).
SymbolStyle	包含了以下的属性:用来决定在序列上标记数据点时使用的形状,大小,和符号的颜色.关于此的更多信息,请参见**序列的线条和符号样式**(240 页).
PointData	返回一个 ChartDataArray 对象,该对象用来获取或者设置序列上每一个数据点的 X,Y 坐标.
X, Y	返回一个 ChartDataArray 对象,该对象用来获取或者设置序列上每一个数据点单独的坐标.一些图表类型含有附加的数据数组.(例如, HiLoOpenClose 还含有 Y1,Y2,和 Y3).

PointData, X, 和 Y 属性允许您获取或者设置用于展示每一个单独序列的数据.

大多数的图表类型需要您为每一个点都提供 X 和 Y 值.例外是饼状图(它不需要 X 值)和一些特殊的图表,例如 **Bubble, HiLo, Gantt,和 HiLoClose**,它们都需要额外的 Y 值.

6. 基本 2D 图表的一般用法

本章节介绍一些基本图表类型的一般用法,这些类型包括 Bar, Pie, 和 X-Y Plot.同时也为每一种图表类型提供了示例代码.示例都是简洁和明了的,同时重点关注的是每一种图表类型的主要方面.本章节中大多数示例程序使用的图表数据都在 Nwind.mdb 文件中,该文件在 Winforms 图表安装过程中被创建.另外在发布包中含有其它很多复杂的示例,这些示例程序包含了在快速入门中没有介绍的高级用法.

6.1 简单条形图表

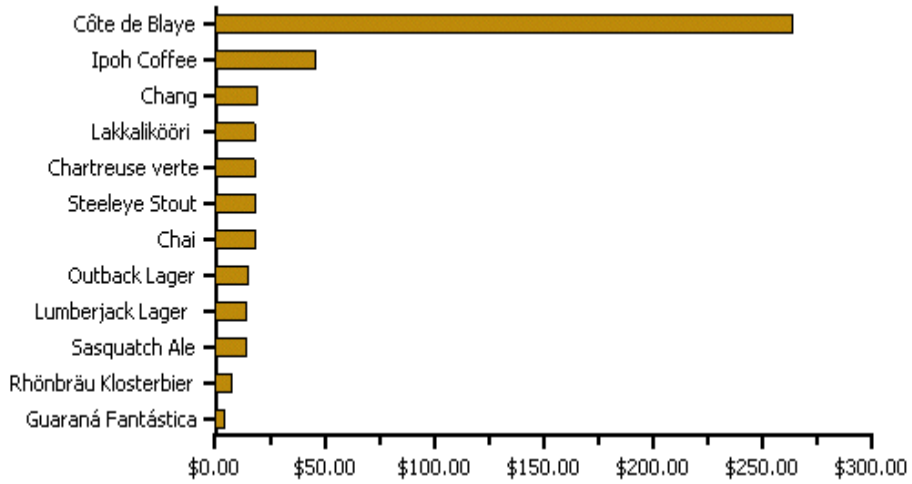
使用图表来展示一系列简单数值的场景是很常见的,例如展示产品价格.这种类型的图表是非常有用的.因为它以一个快速高效的方式展示了各个值之间的相对量.

这种类型图表的主要特征是每一个数据点仅表达了一个价格的信息.当创建这些图表时,将要被展示的值被赋给序列的每一个 Y 值上.Y 值仅提供了点之间的固定间隔,并且经常显示在每一

个数据点附加的标签上.

展示这种信息时推荐使用的图表类型是条形图和饼状图这两种图表类型.

下图展示了一个简单的显示产品价格的条形图表(NorthWind 产品列表中的饮料产品)



请注意条状图是水平显示的,这是为了给显示在轴线上的产品名字留够展示的空间.这个效果通过使用 Inverted 属性完成.

下面的代码创建了上述的条形图.使用的数据的来源是 NorthWind 产品列表中的 Beverages 表中的内容,可以在 C:\Program Files\ComponentOne\Studio for Winforms\Common 文件夹中找到它.在实际的应用程序中,代码可能比示例中的更加简洁,因为一些属性可以在设计时设置.

- Visual Basic

```
    ' get chart data
    Dim data As DataView = _dataSet.Tables("Products").DefaultView
    data.Sort = "UnitPrice"
    data.RowFilter = "CategoryID = 1" ' beverages
    ' configure the chart
    C1Chart1.Reset()
    C1Chart1.ChartArea.Inverted = true
    C1Chart1.ChartGroups(0).ChartType = Chart2DTypeEnum.Bar
    ' create single series for product price
    Dim dscoll As ChartDataSeriesCollection =
C1Chart1.ChartGroups(0).ChartData.SeriesList
    dscoll.Clear()
    Dim series As ChartDataSeries = dscoll.AddNewSeries()
    series.Label = "Product Prices"
    ' populate the series
    series.PointData.Length = data.Count
    Dim i As Integer
    For i = 0 To data.Count - 1
    series.X(i) = I
    series.Y(i) = data(i)("UnitPrice")
    Next I
    ' attach product names to x-axis
    Dim ax As Axis = C1Chart1.ChartArea.AxisX
    ax.AnnoMethod = AnnotationMethodEnum.ValueLabels
    For i = 0 To data.Count - 1
    ax.ValueLabels.Add(i, CType(data(i)("ProductName"), String))
    Next I
    ' configure y-axis
    Dim ay As Axis = C1Chart1.ChartArea.AxisY
    ay.AnnoFormat = FormatEnum.NumericCurrency
```

- C#

```
    // get chart data
    DataView data = _dataSet.Tables["Products"].DefaultView;
    data.Sort = "UnitPrice";
    data.RowFilter = "CategoryID = 1"; // beverages
    // configure the chart
    c1Chart1.Reset();
    c1Chart1.ChartArea.Inverted = true;
    c1Chart1.ChartGroups[0].ChartType = Chart2DTypeEnum.Bar;
    // create single series for product price
    ChartDataSeriesCollection dscoll =
c1Chart1.ChartGroups[0].ChartData.SeriesList;
    dscoll.Clear();
    ChartDataSeries series = dscoll.AddNewSeries();
    series.Label = "Product Prices";
    // populate the series
    series.PointData.Length = data.Count;
    for (int i = 0; i < data.Count; i++)
    {
    series.X[i] = i;
    series.Y[i] = data[i]["UnitPrice"];
    }
    // attach product names to x-axis
    Axis ax = c1Chart1.ChartArea.AxisX;
    ax.AnnoMethod = AnnotationMethodEnum.ValueLabels;
    for (int i = 0; i < data.Count; i++)
    ax.ValueLabels.Add(i, (string)data[i]["ProductName"]);
    // configure y-axis
    Axis ay = c1Chart1.ChartArea.AxisY;
    ay.AnnoFormat = FormatEnum.NumericCurrency;
```

在一个图表中展示多于一个的数据序列的场景是很常见的,使用 C1Chart 可以很容易地做到这一点.而且代码可能很类似于上述的代码,除了您可能需要在 ChartDataSeriesCollection 中添加一个附件的列表之外.

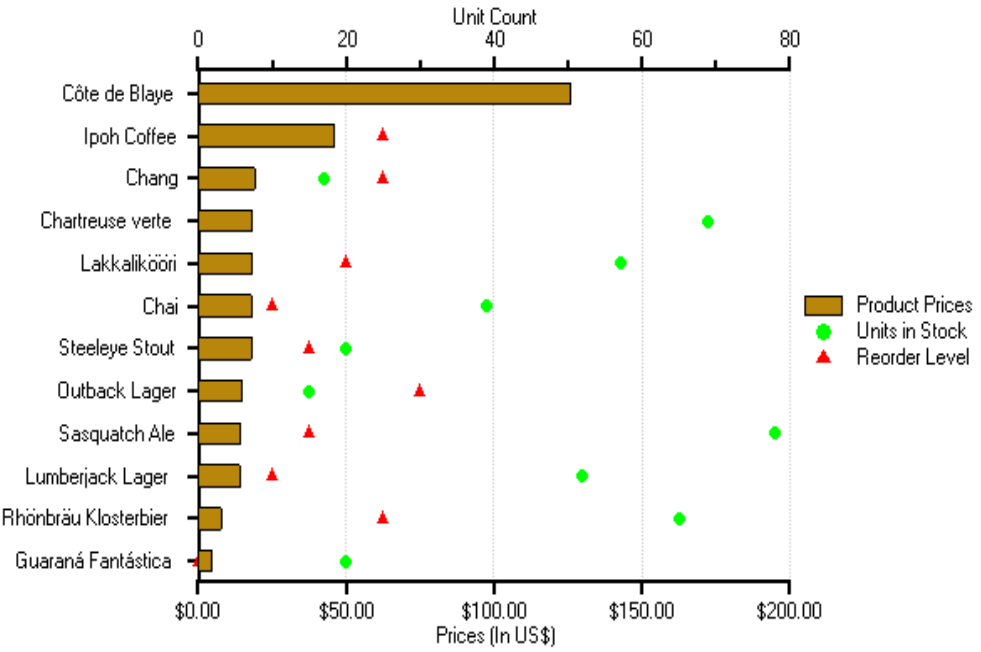
6.2 拥有两个 Y 轴线的条形图表

有一种不太常见的场景,在一个单独的图表中使用不同的单位和缩放来显示序列.例如,您可能需要在一个单独的图表中同时展示产品价格和库存量.为了做到这一点,您需要使用第二个 Y 轴线.主要的轴线用来显式价格(现金单位),次要的轴线用来显示数量.

为了展示用于第二个 Y 轴线显示的数据,添加一个新的序列到第二个图表组中(请注意每一个 C1Chart 含有两个图表组),您可以在第二个图表组中使用一个不同的数据类型,来让图表变得更

加的清晰.

例如,下面的图表含有三个序列,第一个展示了产品单位价格(作为条状图,展示在主要 Y 轴线上),其余的部分展示了仓库中的数量和重新订货标准(作为符号,展示在次要 Y 轴线上).



创建第二个图表的代码的开始部分与创建第一个图表的开始部分的步骤是一样的.两个额外的序列, **Units in Stock** 和 **Reorder Level**,是使用以下的代码创建的.

- Visual Basic

```
' label Y-axis and show legend
C1Chart1.Legend.Visible = True
C1Chart1.ChartArea.AxisY.Text = "Prices (in US$)"
C1Chart1.ChartArea.AxisY2.Text = "Unit Count"
' create two series for units in stock and reorder level
' (these are plotted against the secondary Y axis)
dscoll = C1Chart1.ChartGroups(1).ChartData.SeriesList
dscoll.Clear()
' units in stock
series = dscoll.AddNewSeries()
series.Label = "Units In Stock"
series.SymbolStyle.Color = Color.Green
series.LineStyle.Pattern = LinePatternEnum.None
series.PointData.Length = data.Count
For i = 0 To data.Count - 1
series.X(i) = I
series.Y(i) = data(i)("UnitsInStock")
Next I
' reorder level
series = dscoll.AddNewSeries()
series.Label = "Reorder Level"
series.SymbolStyle.Color = Color.Red
series.LineStyle.Pattern = LinePatternEnum.None
series.PointData.Length = data.Count
For i = 0 To data.Count - 1
series.X(i) = I
series.Y(i) = data(i)("ReorderLevel")
Next I
' show gridlines for secondary Y-axis
C1Chart1.ChartArea.AxisY2.GridMajor.Visible = True
```

- C#

```
// label Y-axis and show legend
_c1c.Legend.Visible = true;
_c1c.ChartArea.AxisY.Text = "Prices (in US$)";
_c1c.ChartArea.AxisY2.Text = "Unit Count";
// create two series for units in stock and reorder level
// (these are plotted against the secondary Y axis)
dscoll = _c1c.ChartGroups[1].ChartData.SeriesList;
dscoll.Clear();
// units in stock
series = dscoll.AddNewSeries();
series.Label = "Units In Stock";
series.SymbolStyle.Color = Color.Green;
series.LineStyle.Pattern = LinePatternEnum.None;
series.PointData.Length = data.Count;
for (int i = 0; i < data.Count; i++)
{
series.X[i] = i;
series.Y[i] = data[i]["UnitsInStock"];
}
// reorder level
series = dscoll.AddNewSeries();
series.Label = "Reorder Level";
series.SymbolStyle.Color = Color.Red;
series.LineStyle.Pattern = LinePatternEnum.None;
series.PointData.Length = data.Count;
for (int i = 0; i < data.Count; i++)
{
series.X[i] = i;
series.Y[i] = data[i]["ReorderLevel"];
}
// show gridlines for secondary Y-axis
_c1c.ChartArea.AxisY2.GridMajor.Visible = true;
```

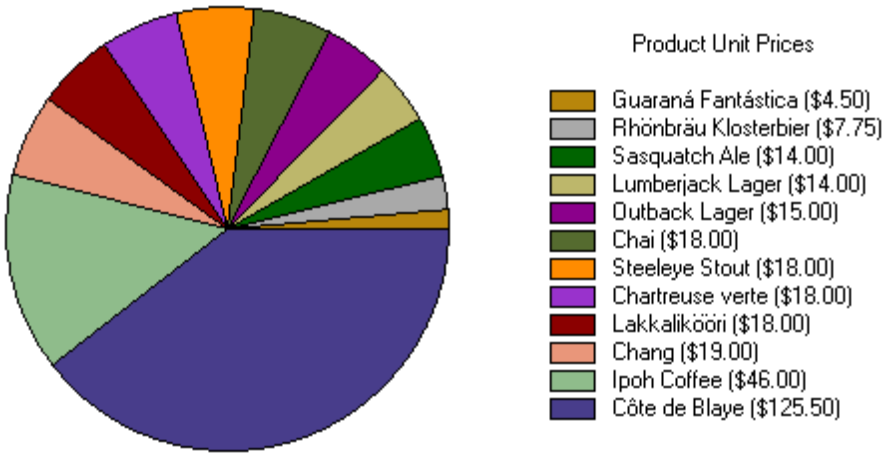
6.3 饼状图表

饼状图表一般用来展示简单值.它们在视觉上很有吸引力,并且经常被附加上 3D 效果,例如阴影和旋转.**C1Chart** 允许您给图表添加 3D 效果.但是您必须小心使用,因为它们可能会使数据失真.

饼状图表跟其它的 **C1Chart** 图表类型有一个最显著的区别:在饼状图表中,一个饼状图的切

片代表一个数据序列.所以,您将不会遇见只含有一个序列的饼状图(这样的话饼状图只会含有一个切片).在最常见的场景下,饼状图含有多个序列(一个切片对应一个序列),且每一个序列含有一个单独的数据点.**C1Chart** 在图表中通过多个切片来展示有多个数据点的序列.

当您思考用于标注每一个序列的标签和如果想要在图表图例中展示它们时需要注意这一个特征.以下的图表展示了在一个饼状图表中展示与之前相同的销售数据.



用来创建这个图表的代码跟创建条状图表的代码有明显的不同.它为每一个值创建了一个序列.每一个序列只含有一个单独的数据点.下面的代码展示了创建上述图表的步骤:

- Visual Basic

```
    ' get chart data
    Dim data As DataView = _dataSet.Tables["Products"].DefaultView
    data.Sort = "UnitPrice"
    data.RowFilter = "CategoryID = 1" ' beverages
    ' configure chart
    C1Chart1.Reset()
    C1Chart1.BackColor = Color.White
    C1Chart1.ChartArea.Style.Font = new Font("Tahoma", 8)
    C1Chart1.ChartGroups(0).ChartType = Chart2DTypeEnum.Pie
    ' get series collection (pies have one series per slice)
    Dim dscoll As ChartDataSeriesCollection =
C1Chart1.ChartGroups(0).ChartData.SeriesList
    dscoll.Clear()
    ' populate the series
    Dim i As Integer
    For i = 0 To data.Count - 1
    Dim series As ChartDataSeries = dscoll.AddNewSeries()
    series.PointData.Length = 1
    series.Y(0) = data(i)("UnitPrice")
    series.Label = String.Format("{0} ({1:c})", _
    data(i)("ProductName"), data(i)("UnitPrice"))
    Next I
    ' show pie legend
    C1Chart1.Legend.Visible = True
    C1Chart1.Legend.Text = "Product Unit Prices"
```

- C#

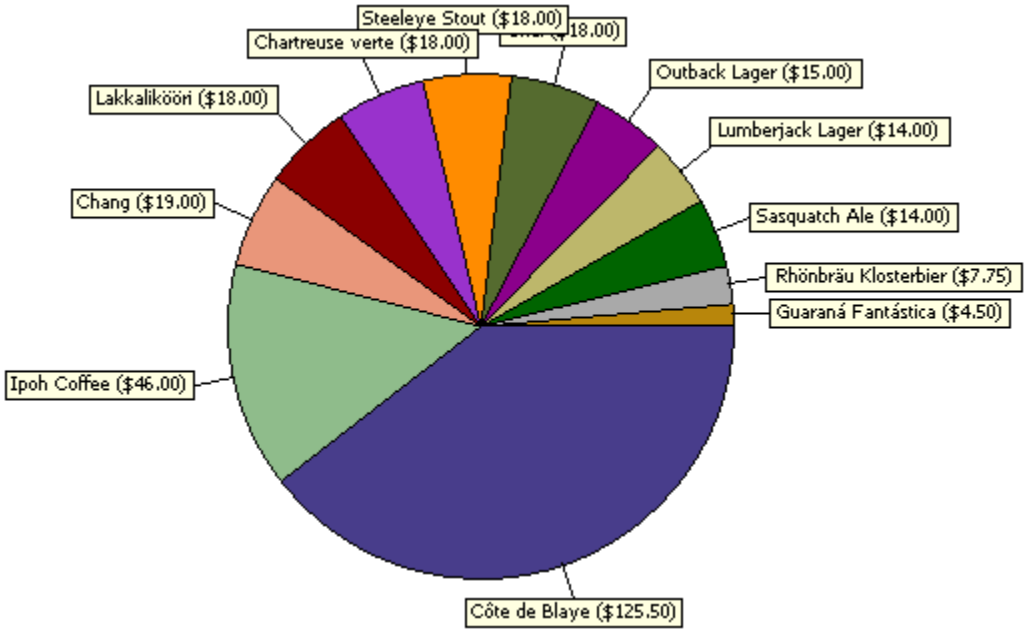
```
    // get chart data
    DataView data = _dataSet.Tables["Products"].DefaultView;
    data.Sort = "UnitPrice";
    data.RowFilter = "CategoryID = 1"; // beverages
    // configure chart
    c1Chart1.Reset();
    c1Chart1.BackColor = Color.White;
    c1Chart1.ChartArea.Style.Font = new Font("Tahoma", 8);
    c1Chart1.ChartGroups[0].ChartType = Chart2DTypeEnum.Pie;
    // get series collection (pies have one series per slice)
    ChartDataSeriesCollection dscoll =
c1Chart1.ChartGroups[0].ChartData.SeriesList;
    dscoll.Clear();
    // populate the series
    for (int i = 0; i < data.Count; i++)
    {
    ChartDataSeries series = dscoll.AddNewSeries();
    series.PointData.Length = 1;
    series.Y[0] = data[i]["UnitPrice"];
    series.Label = string.Format("{0} ({1:c})",
    data[i]["ProductName"], data[i]["UnitPrice"]);
    }
    // show pie legend
    c1Chart1.Legend.Visible = true;
    c1Chart1.Legend.Text = "Product Unit Prices";
```

6.4 带有图表标签的饼状图表

在一个饼状图表中显示数据序列的通常做法是使用一个图例,例如上一个例子.这种方法很管用,因为信息被显示在一个单独的区域中,无论您有多少个的标签,它们被隔离在在图例的旁边.

但是,在一些场景下,您可能希望将标签紧挨着数据点显示,以便让它们的关系更加的明显.您还可能希望在数据点上添加含有信息的标签或者其它的图表元素,并且在一个特定的位置或者紧挨着某个特定的图表元素来放置标签.对于这种类型的任务,C1Chart 提供了 ChartLabels 属性,它让您创建 Label 对象并且将它们附加在任何的图表元素上.

例如,下面的图例展示了与前面的示例相同的饼状图表,这次给每一个切片上附加了标签而不是使用图例来表示它们.



这个图表也表明了一个常见的问题:当您附加了过多的图表标签时,他们可能会溢出,并且很难准确地定位它们,因为它们实在太多了(在本示例中,C1Chart 自动定位了标签).

创建这个图表的代码的开始部分和用于创建第一个饼状图表的步骤是一样的.标签是使用以下的代码来创建的.

- Visual Basic

```
' hide legend, configure labels
C1Chart1.Legend.Visible = false
Dim s As Style = C1Chart1.ChartLabels.DefaultLabelStyle
s.Font = new Font("Tahoma", 7)
s.BackColor = SystemColors.Info
s.Opaque = true
s.Border.BorderStyle = BorderStyleEnum.Solid
' attach labels to each slice
Dim i As Integer
For i = 0 To data.Count - 1
Dim lbl As C1.Win.C1Chart.Label = _
C1Chart1.ChartLabels.LabelsCollection.AddNewLabel()
lbl.Text = string.Format("{0} ({1:c})", _
data(i)("ProductName"), data(i)("UnitPrice"))
lbl.Compass = LabelCompassEnum.Radial
lbl.Offset = 20
lbl.Connected = True
lbl.Visible = True
lbl.AttachMethod = AttachMethodEnum.DataIndex
Dim am As AttachMethodData = lbl.AttachMethodData
am.GroupIndex = 0
am.SeriesIndex = I
am.PointIndex = 0
Next i
```

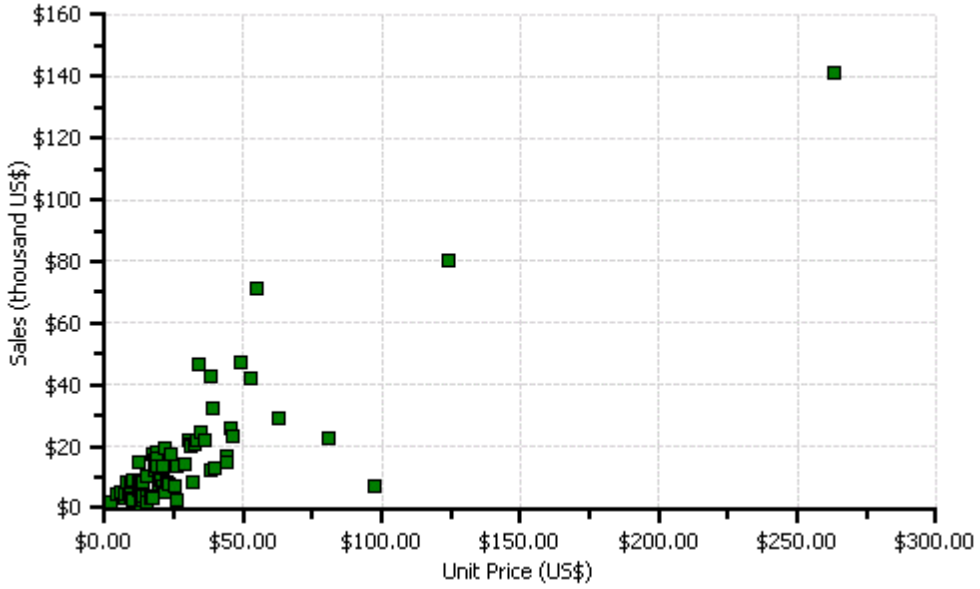
- C#

```
// hide legend, configure labels
c1Chart1.Legend.Visible = false;
Style s = c1Chart1.ChartLabels.DefaultLabelStyle;
s.Font = new Font("Tahoma", 7);
s.BackColor = SystemColors.Info;
s.Opaque = true;
s.Border.BorderStyle = BorderStyleEnum.Solid;
// attach labels to each slice
for (int i = 0; i < data.Count; i++)
{
C1.Win.C1Chart.Label lbl =
c1Chart1.ChartLabels.LabelsCollection.AddNewLabel();
lbl.Text = string.Format("{0} ({1:c})",
data[i]["ProductName"], data[i]["UnitPrice"]);
lbl.Compass = LabelCompassEnum.Radial;
lbl.Offset = 20;
lbl.Connected = true;
lbl.Visible = true;
lbl.AttachMethod = AttachMethodEnum.DataIndex;
AttachMethodData am = lbl.AttachMethodData;
am.GroupIndex = 0;
am.SeriesIndex = i;
am.PointIndex = 0;
}
```

6.5 XY-绘制(散点图)

条状图和饼状图使用一个单独的值来代表每一个数据点(价格,单位库存,或者重订货标准).相反地, XYPlot 使用两个值来代表一个数据点.这个特性在描述数据在两个维度上的关系时非常的有用,通常它被用于对数据进行统计分析.

例如,下图展示了一个 XYPlot 图表,它描述了产品单价和销量的关系.



这种类型的图表通常用来支持量化变量间的关系的统计技术(典型的是线性回归分析).例如,上面的图表表明了一个单价在$250 以上的点跟其它点的距离相当的远.在一个线性回归中,这个点被称为"异常值",并会对分析的结果有更多更深远的影响,相对于其它的点.

XYPlot 图表本身也是非常有用的,它展示了关系的定性方面.例如,销量是否跟单价有关系?在这个示例中,数据看上去确实有一个关系.可能客户喜欢在 NorthWind 中购买昂贵的产品,但是在它们当地的超市中喜欢购买便宜的产品.

下面的代码用来创建上面的示例图表.它跟创建条状图表的代码非常的类似.除了它需要设置 X 和 Y 值到数据上,而不是为 X 值使用一个简单的序列来描述.

- Visual Basic

```
' get chart data
Dim data As DataView = _dataSet.Tables("Sales").DefaultView
data.Sort = "UnitPrice"
' configure chart
C1Chart1.Reset()
C1Chart1.ChartGroups(0).ChartType = Chart2DTypeEnum.XYPlot
' create single series for product price vs sales
Dim dscoll As ChartDataSeriesCollection = _
C1Chart1.ChartGroups(0).ChartData.SeriesList;
dscoll.Clear();
Dim series As ChartDataSeries = dscoll.AddNewSeries()
' show symbols only (no lines)
series.SymbolStyle.Color = Color.Green
series.SymbolStyle.OutlineColor = Color.Black
series.LineStyle.Pattern = LinePatternEnum.None
' populate the series
series.PointData.Length = data.Count
Dim i As Integer
For i = 0 To data.Count - 1
series.X(i) = data(i)("UnitPrice")
series.Y(i) = data(i)("ProductSales")
Next I
' attach product names to x-axis
Dim ax As Axis = C1Chart1.ChartArea.AxisX
ax.Text = "Unit Price (US$)"
ax.AnnoFormat = FormatEnum.NumericCurrency
ax.GridMajor.Visible = True
' configure y-axis
Dim ay As Axis = C1Chart1.ChartArea.AxisY
ay.Text = "Sales (thousand US$)"
ay.AnnoFormat = FormatEnum.NumericManual
ay.AnnoFormatString = "$#,##0,"
ay.GridMajor.Visible = True
```

- C#

```
// get chart data
DataView data = _dataSet.Tables["Sales"].DefaultView;
data.Sort = "UnitPrice";
// configure chart
c1Chart1.Reset();
c1Chart1.ChartGroups[0].ChartType = Chart2DTypeEnum.XYPlot;
// create single series for product price vs sales
ChartDataSeriesCollection
dscoll = c1Chart1.ChartGroups[0].ChartData.SeriesList;
dscoll.Clear();
ChartDataSeries series = dscoll.AddNewSeries();
// show symbols only (no lines)
series.SymbolStyle.Color = Color.Green;
series.SymbolStyle.OutlineColor = Color.Black;
series.LineStyle.Pattern = LinePatternEnum.None;
// populate the series
series.PointData.Length = data.Count;
for (int i = 0; i < data.Count; i++)
{
series.X[i] = data[i]["UnitPrice"];
series.Y[i] = data[i]["ProductSales"];
}
// attach product names to x-axis
Axis ax = c1Chart1.ChartArea.AxisX;
ax.Text = "Unit Price (US$)";
ax.AnnoFormat = FormatEnum.NumericCurrency;
ax.GridMajor.Visible = true;
// configure y-axis
Axis ay = c1Chart1.ChartArea.AxisY;
ay.Text = "Sales (thousand US$)";
ay.AnnoFormat = FormatEnum.NumericManual;
ay.AnnoFormatString = "$#,##0,";
ay.GridMajor.Visible = true;
```

7. 专用的 2D 图表

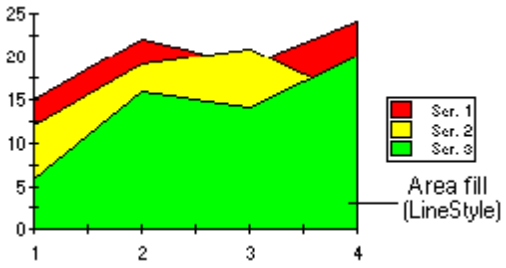
2D 图表可以以几种基本类型中的一种类型或者子类型来展示数据.子类型是对基本图表类型的扩展.例如 Bubble 图表.更加特殊化的图表类型,例如 Gantt 图表,也可以被创建.另外,很多的图表都含有根据属性来转变成为其他图表类型的能力,前提是两种图表类型所需要的数据是兼容

的.例如,相同的数据可以在 XY-Plot 中展示,之后再做为一个条状图表来展示.

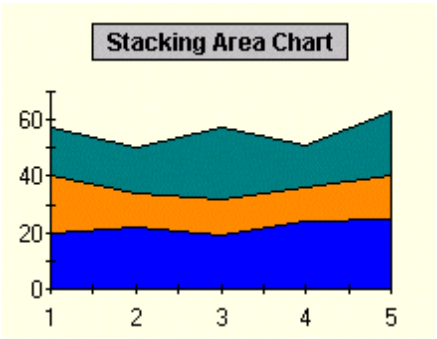
本章节介绍了 C1Chart 中可用的 2D 图表类型.展示了如何选择专有的图表类型,然后展示了如何在一些图表类型中添加 3D 效果.

7.1 区域图表

一个区域图表以数据连接点的形式来绘制每一个序列,然后在点下面填充颜色.每一个序列被绘制在前面的序列的顶部.序列可以单独绘制或者叠加绘制.使用 LineStyle 属性,每一个序列的填充属性都可以被自定义.关于此的更多信息,请参见序列的线条和符号样式(240 页).



使用 ChartGroup 对象的 Stacked 属性来创建一个叠加的区域图表.叠加图表在前一个序列值的顶部叠加显示每一个序列的值.



在设计时设置图表类型为区域图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 **ChartType** 属性为 **Area.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 **Chart Properties**.在 **Gallery** 中选择图表类型为 **Area.**
- 另外一种可行的方法是在属性面板中选择图表属性,在 **Gallery** 中,选择图表类型为 **Area.**

7.1.1 区域图表编程时的讨论

下表描述了区域图表使用的数据数组.每一个数据序列代表将要被创建的 X 和 Y 数组值.给其它数组添加值将不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有 Y 轴线的位置.

Y1	对区域图表没有影响.
Y2	对区域图表没有影响.
Y3	对区域图表没有影响.

7.1.2 区域图表的 3D 效果

C1Chart 的 3D 效果可以用在区域图表或者叠加区域图表上,来创建每一个序列在高度上的假象.通过使用深度,高地,旋转,和阴影效果属性.您可以增强您的区域图表效果.使它们看起来更加的有视觉深度.

为了访问区域图表的 3D 视图.调整 View3D 对象的属性.View3D 对象是 PlotArea 对象的一个成员,然后 PlotArea 又是 ChartArea 对象的一个成员.通过调整 View3D 对象的属性,深度,高地,旋转,阴影,您可以自定义 3D 视图.

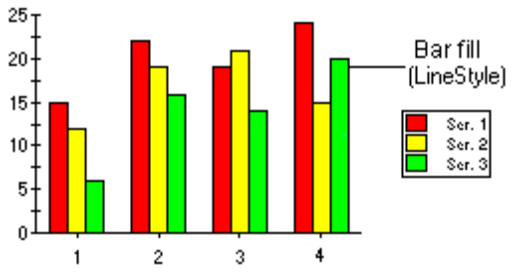
请注意 Depth 属性是所有 3D 图表类型逻辑的关键. Elevation 和 Rotation 属性修改用户查看图表的方式.所以正是 Depth 属性实际上支配了一个图表是否是 3D 的.通过为 Depth 属性使用一个非 0 值,并且设置 Elevation 和 Rotation 属性的值为 0.您可以创建一个 3D 图表,虽然什么事情起来都没有改变.实际上您只是在看图表的正前方表面,就像一个标准的区域图表的视觉效果一样.

同时请注意,对一些数据进行 3D 视图的展示的效果可能是很令人满意的,但是同时其它的数据需要使用 2D 的面板.对于这些场景,调整附加在每一个 ChartGroup 上的 Use3D 属性.

7.2 条状图表

一个条状图表是一个倒置列图表,它的分类轴线是垂直轴线.一个条状/列图表在一个集群中以一个条状显示每一个序列.集群的数目是数据中点的数量.每一个集群显示在每一个序列中的第 n 个数据点.使用 LineStyle 属性,每一个序列的填充属性可以被自定义.关于此的更多信息,请参见**序列的线条和符号样式**(240 页).

下图展示了一个条状图表:



设计时设置图表类型为条状图表

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 **ChartType** 属性为 **bar**.
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 **Chart Properties**.在 **Gallery** 中选择图表类型为 **bar**.
- 另外一种可行的方法是在属性面板中选择图表属性,在 **Gallery** 中,选择图表类型为 **Bar**.

7.2.1 条状图表编程时的讨论

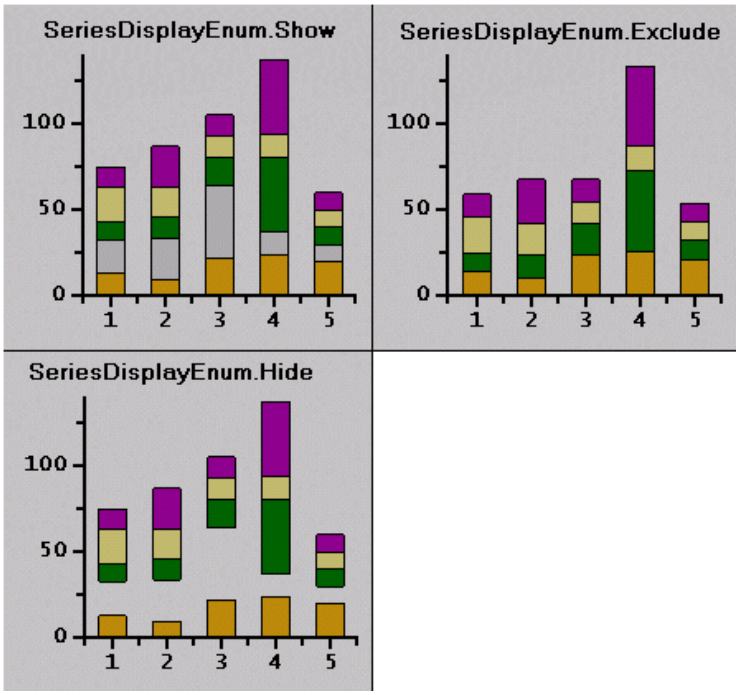
下表描述了条状图表使用的数据数组.每一个数据序列代表将要被创建的 X 和 Y 数组值.给其它数组添加值将不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有 Y 轴线的位置.
Y1	对条状图表没有影响.
Y2	对条状图表没有影响.
Y3	对条状图表没有影响.

7.2.2 浮动的条状图表

每一个叠加条状图表的序列可以被隐藏(序列将被作为一个空白集)或者移除(当序列不被考虑成是图表数据的一部分时).使用 ChartDataSeries 对象的 Display 属性来决定每一个数据序列是否显示,隐藏,还是移除.

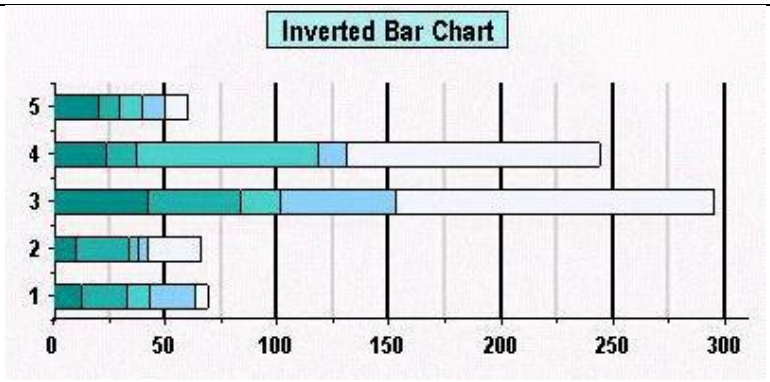
在下面的例子中,第二个数据序列将首先被显示,然后被隐藏,最后被移除.



在设计时可以通过 SeriesList Collection Editor 来访问 Display 属性.该设计器可以通过打开 **ChartGroupsCollection Editor** 后访问.展开 **ChartData** 节点,然后点击紧挨着 **SeriesList** 属性的 **ellipsis** 按钮.

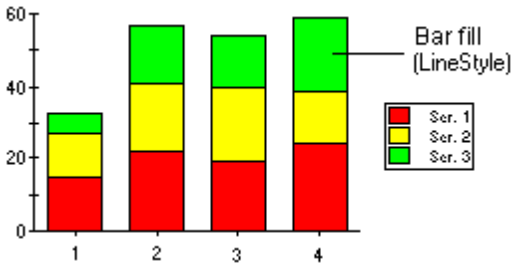
7.2.3 翻转的条状图表

一个条状图表是一个旋转的列图表,它的 X 和 Y 轴线被翻转了.关于此的更多信息,请参见翻转和旋转图表轴线(219 页).



7.2.4 叠加条状图表

一个叠加条状图表将每一个序列作为叠加条状集群的一个部分来显示.集群的数目是数据中点的数目.每一个条状显示每一个数据序列中的第 n 个数据点. Cylinder, Pyramid,和 Cone Bar 图表可以被叠加,这是通过设置 Stacked 属性为 True 来完成的.使用 LineStyle 属性,每一个序列的填充属性都可以被自定义,关于此的更多信息,请参见序列的线条和符号样式(240 页).



示例代码

下面的代码创建了一个叠加的条状图表,首先在您的 Visual Studio 工程中添加 C1Chart 控件,然后添加如下的代码,程序运行后,就可以看到一个叠加条状图表的示例.

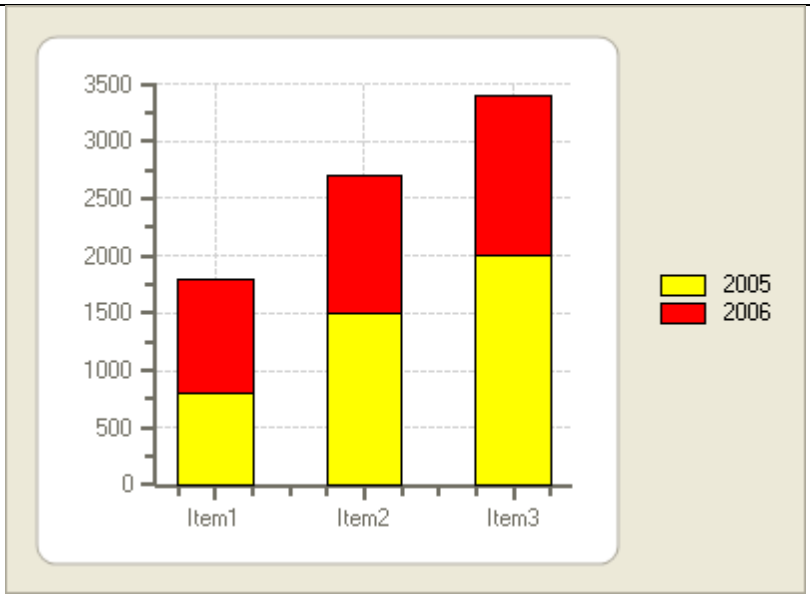
- Visual Basic

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    ' Clear previous data
    C1Chart1.ChartGroups(0).ChartData.SeriesList.Clear()
    ' Data
    Dim items As String() = New String() {"Item1", "Item2", "Item3"}
    Dim sales2005 As Integer() = New Integer() {800, 1500, 2000}
    Dim sales2006 As Integer() = New Integer() {1000, 1200, 1400}
    ' Create first series
    Dim ds2005 As C1.Win.C1Chart.ChartDataSeries =
C1Chart1.ChartGroups(0).ChartData.SeriesList.AddNewSeries()
    ds2005.Label = "2005"
    ds2005.LineStyle.Color = Color.Yellow
    ds2005.X.CopyDataIn(items)
    ds2005.Y.CopyDataIn(sales2005)
    ' Create second series
    Dim ds2006 As C1.Win.C1Chart.ChartDataSeries =
C1Chart1.ChartGroups(0).ChartData.SeriesList.AddNewSeries()
    ds2006.Label = "2006"
    ds2006.LineStyle.Color = Color.Red
    ds2006.AutoEnumerate = True
    ds2006.Y.CopyDataIn(sales2006)
    ' Set chart type
    C1Chart1.ChartGroups(0).ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Bar
    C1Chart1.ChartGroups(0).Stacked = True
    ' Set y-axis minimum
    C1Chart1.ChartArea.AxisY.Min = 0
    C1Chart1.Legend.Visible = True
    ' Remove the Axes caption
    C1Chart1.ChartArea.AxisX.Text = ""
    C1Chart1.ChartArea.AxisY.Text = ""
    End Sub
```

- C#

```
private void Form1_Load(object sender, EventArgs e)
{
// Clear previous data
c1Chart1.ChartGroups[0].ChartData.SeriesList.Clear();
// Data
string[] items = new string[] { "Item1", "Item2", "Item3"};
int[] sales2005 = new int[] { 800, 1500, 2000};
int[] sales2006 = new int[] { 1000, 1200, 1400};
// Create first series
C1.Win.C1Chart.ChartDataSeries ds2005 =
c1Chart1.ChartGroups[0].ChartData.SeriesList.AddNewSeries();
ds2005.Label = "2005";
ds2005.LineStyle.Color = Color.Yellow;
ds2005.X.CopyDataIn( items);
ds2005.Y.CopyDataIn( sales2005);
// Create second series
C1.Win.C1Chart.ChartDataSeries ds2006 =
c1Chart1.ChartGroups[0].ChartData.SeriesList.AddNewSeries();
ds2006.Label = "2006";
ds2006.LineStyle.Color = Color.Red;
ds2006.AutoEnumerate = true;
ds2006.Y.CopyDataIn( sales2006);
// Set chart type
c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Bar;
c1Chart1.ChartGroups[0].Stacked = true;
// Set y-axis minimum
c1Chart1.ChartArea.AxisY.Min = 0;
c1Chart1.Legend.Visible = true;
// Remove the Axes caption
c1Chart1.ChartArea.AxisX.Text = "";
c1Chart1.ChartArea.AxisY.Text = "";
}
```

上面的示例代码会创建如下的图表:



7.2.5 特殊的条形图表属性

一个条形图表将序列作为一个集群中的条形来绘制.您可以在一个单独的行上以 2D 条形图表显示每一个序列,或者在多于一个的行上使用 3D 条状图来显示每一个序列.3D 条状图表提供了一个很令人感兴趣的视图,您可以在 3D 条形图或者列图表的前面查看图表而不是常见的那一面的视图.条形图表和叠加条形图表中集群的大小和空间可以被自定义.另外,您可以改变条形图表的外观到以下的任何一种形状:圆筒,圆锥,或者金字塔形状.

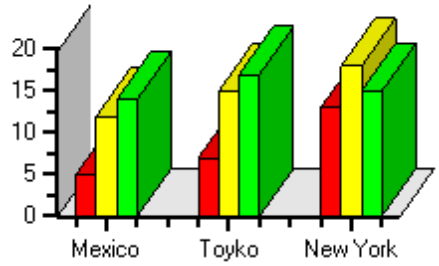
外观

虽然条形图表一般情况下使用矩形条状图(默认方式)来展示,但是您同时可以使用圆筒,圆锥,或者金字塔形状来表述图表,从而达到一个不同的效果.为了改变条状图表的形状,从默认形状到圆筒,圆锥,金字塔形.使用 Appearance 属性.

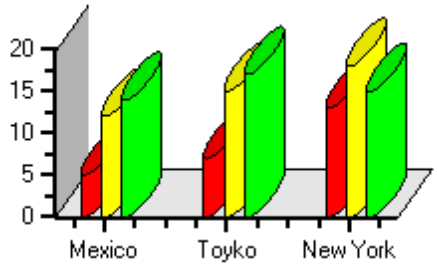
注意:下面的圆筒和圆锥条状图表看其来比圆环更加的椭圆,是因为图表的深度,为了让它们看起来更加的圆,您可以减少 Depth 属性.关于 3D 条状图表效果的更多信息,请参见条状图表 3D 效果(89 页).

下图展示了 Appearance 属性值的效果:

Appearance = Default

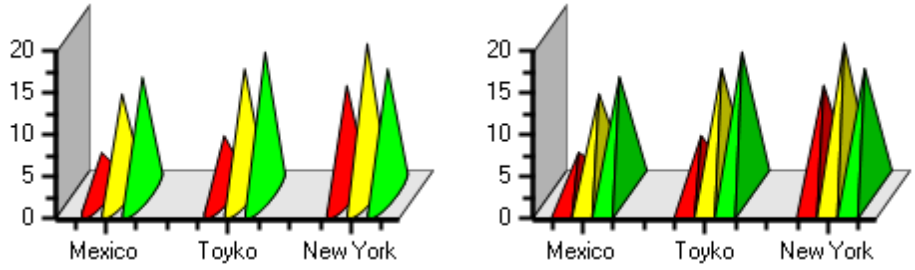


Appearance = Cylinder



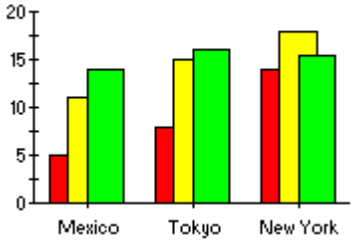
Appearance = Cone

Appearance = Pyramid



集群叠加

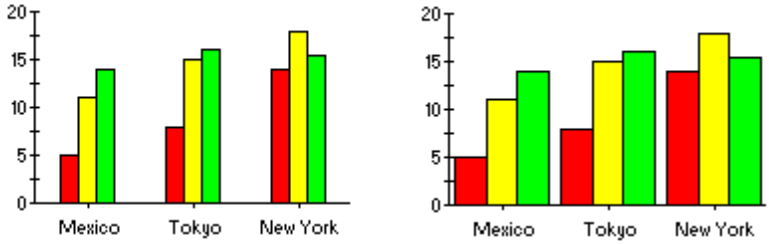
使用 ClusterOverlap 属性来设置在一个集群中条形图叠加的数目.这个值代表了条形叠加的百分比,合法的取值范围为从 0 到 100.下图展示了一个 ClusterOverlap 属性为 50 百分比的条形图表.



集群宽度

使用 ClusterWidth 属性来设置每一个条形集群使用的空间.这个值代表可用空间的百分比,合法的取值范围为从 0 到 100.

ClusterWidth = 50% **ClusterWidth = 90%**



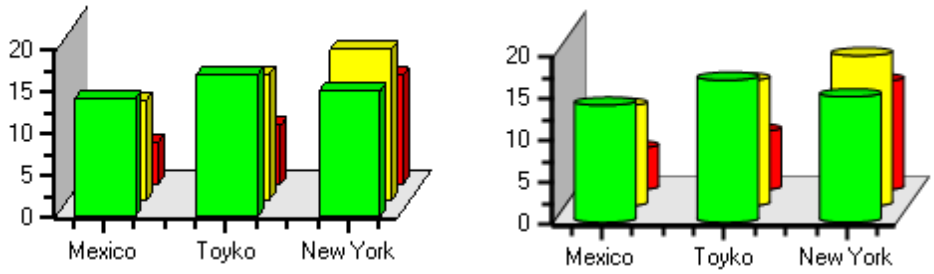
Bar 类的属性 ClusterWidth 和 ClusterOverlap 可以在设计时中通过 ChartGroupsCollection Editor 的 bar 节点来访问.

多行

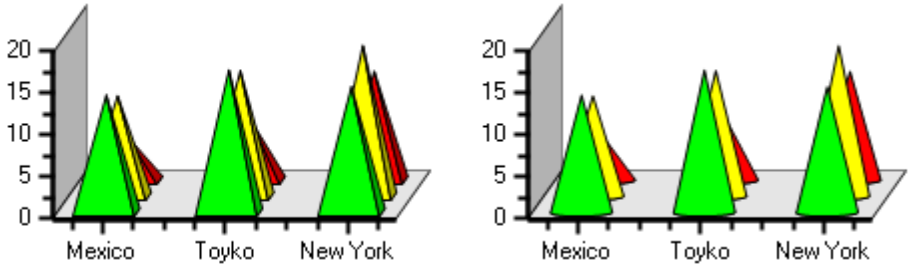
当 Use3D 属性被设置为 True 时,会启用 3D 图表效果.您可以在 3D 图表中使用 MultiRow 属性来在集群中为每一个条形或者列显示一个新行.

下面展示了每一种条形图类型中 MultiRow 属性的效果.

MultiRow = True **MultiRow = True**

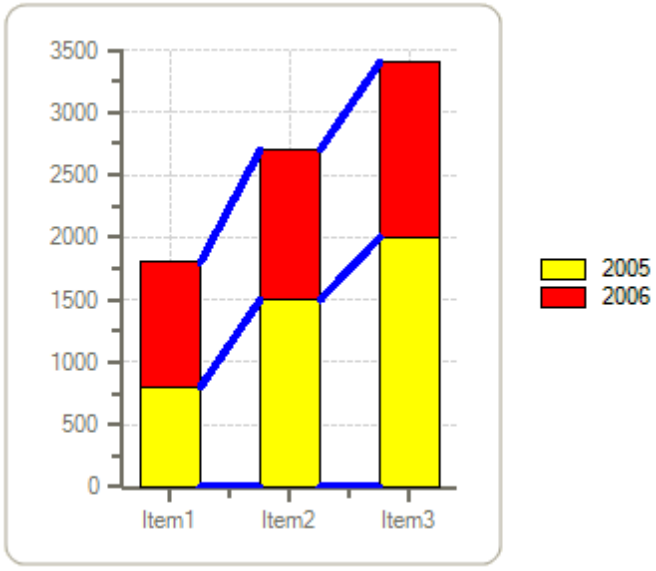


MultiRow = True **MultiRow = True**



叠加 2D 条形图表和列图表的线条

需要设置 BarLines 属性为 True,当您想要在叠加 2D 条状图和列图表的点和点之间的数据序列矩形中显示线条的话.当您设置 BarLines 属性为 True 时,您可以使用 BarLineColor 和 BarLineThickness 属性来为条形图线条指定线条的颜色和线条厚度.如下图所示:



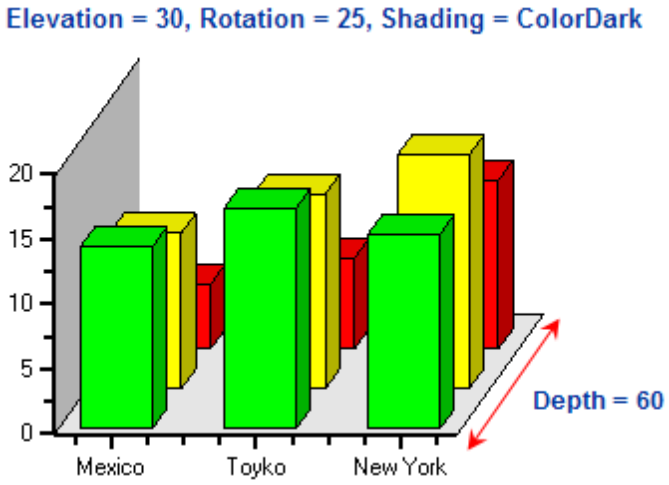
通过在叠加 2D 条形图和列图表中使用条形线条,可以更加容易地参看到量化的信息.因为它使用条形线条来对比在 X-轴线(列图表)或者 Y 轴线(条形图)中的值.

7.2.6 条形图表 3D 效果

您可以在条形图和叠加条形图中使用 C1Chart 的 3D 效果来增加您的图表的外观.通过使用 depth, elevation, rotation, 和 shading 属性,您可以在给每一个数据序列创建深度效果.

当 Depth 属性被用来修改条形图表的深度时,它并不影响列的形状.但是,它影响圆筒,圆锥,和金字塔的形状.金字塔的宽度会在 X 轴线上的条形数目变化时自动改变.但是条形图的深度会受到 Depth 属性的影响.

下图展示了使用 3D 效果来让图表变得更加的在视觉上有吸引力:



为了访问条形图表的 3D 视图.调整 View3D 对象的属性.View3D 对象是 PlotArea 对象的一个成员,然后 PlotArea 又是 ChartArea 对象的一个成员.通过调整 View3D 对象的属性,深度,高地,旋转,阴影,您可以自定义 3D 视图.

请注意 Depth 属性是所有 3D 图表类型逻辑的关键. Elevation 和 Rotation 属性修改用户查看图表的方式.所以正是 Depth 属性实际上支配了一个图表是否是 3D 的.通过为 Depth 属性使用一个非 0 值,并且设置 Elevation 和 Rotation 属性的值为 0.您可以创建一个 3D 图表,虽然什么事情起来都没有改变.实际上您只是在看图表的正前方表面,就像一个标准的区域图表的视觉效果一样.

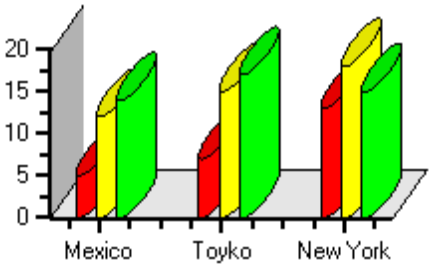
同时请注意,对一些数据进行 3D 视图的展示的效果可能是很令人满意的,但是同时对于其它的数据也需要使用 2D 的面板.对于这些场景,调整附加在每一个 ChartGroup 上的 Use3D 属性.

7.2.7 条形图表的变种

条形图表有很多的变种,您可以使用圆筒,圆锥,和金字塔图表来显示数据序列.这些图表跟 3D 条形图和列图表的功能一样:对比数据序列.唯一个差别在外观上,它们使用圆筒,圆锥或者金字塔来表示代表条形,而不是使用矩形.下面的主题提供了关于圆筒,圆锥和金字塔的更多信息.

7.2.7.1 圆筒图表

一个圆筒图表是条形和列图表的变形.它使用圆筒来代表条形或者列.圆筒图表在两个端点都创建相同的长圆形箱.像所有的条形和列图表一样,圆筒图表非常适合用来对比单独的项目或者组项目.

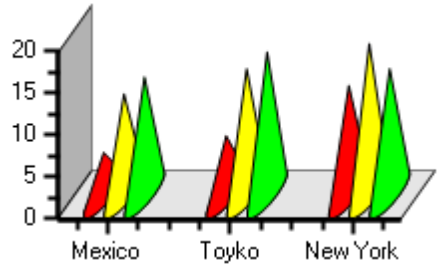


改变图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 ChartType 属性为 Bar,然后设置外观属性为 **Cylinder**.
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties.在 Gallery 中选择 **Cylinder**.
- 另外一种可行的方法是在 **C1Chart** 工具栏中选择 **Cylinder.**

7.2.7.2 圆锥图表

一个圆锥图表是 3D 条形和列图表的变形.它使用圆锥来代表条形或者列.圆锥图表本质上是一个旋转的三角形.它有一个扁圆形的底部和一个在更高点的弯曲侧.

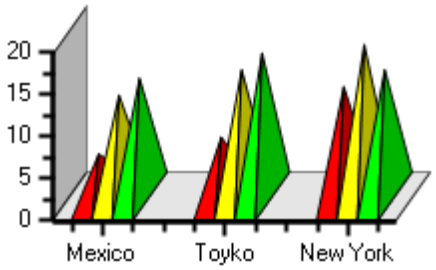


在设计时其设置条形图表类型为圆锥

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 ChartType 属性为 **Cone.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties.在 Gallery 中选择图表类型为 **Cone.**
- 另外一种可行的方法是在 **C1Chart** 工具栏中选择 **Cone.**

7.2.7.3 金字塔图表

一个金字塔图表是 3D 条形和列图表的变形.它使用金字塔来代表条形或者列.金字塔图表非常类型与圆锥图表,处理他们的底部,金字塔图表经常用做地理学上的用途.



在设计时其设置图表类型为金字塔

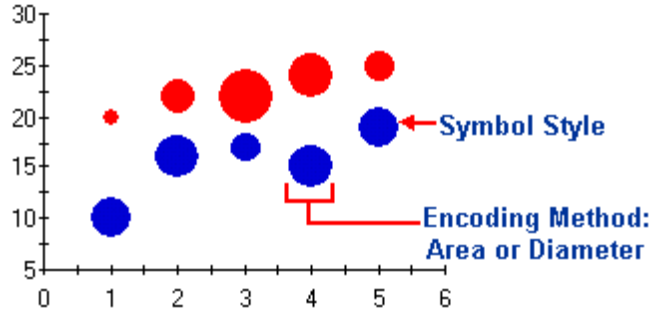
- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 ChartType 属性为 Bar,然后设置外观属性为 Pyramid.
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties.

在 Gallery 中选择图表类型为 **Pyramid.**

- 另外一种可行的方法是在 **C1Chart** 工具栏中选择 **Pyramid.**

7.3 气泡图

一个气泡图由两个独立的值组成,这两个值提供了点的 Y 值和点的大小.气泡图通常通过改变大小来在每一个点上显示一个附加的数据值.Y 数组元素决定了笛卡尔位置(类似于在一个 XY-Plot 图表中),Y1 元素的值决定了每一个点上的气泡的大小.点的大小可以通过区域或者直径进行编码.使用 LineStyle 和 SymbolStyle 属性,符号样式和颜色,连接线的外观,都可以被自定义.关于此的更多信息,请参见序列的线条和符号样式(240 页).



在设计时其设置图表类型为气泡图

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 ChartType 属性为 **Bubble.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties.在 Gallery 中选择图表类型为 XY-Plot,并且图表值类型为 **Bubble..**
- 另外一种可行的方法是在属性面板中选择图表属性,在 **Gallery** 中,选择图表类型为 **XY-Plot,**并且图表值类型为 **Bubble.**

7.3.1 气泡图表编程时的讨论

下表描述了气泡图表使用的数据数组.每一个数据序列代表将要被创建的 X,Y 和 Y1 数组值.给其它数组添加值将不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有气泡相对于 X 轴线的位置.
Y	持有气泡相对于 Y 轴线的位置.
Y1	持有气泡的相对位置.
Y2	对条状图表没有影响.
Y3	对条状图表没有影响.

7.3.2 特殊的气泡图表属性

一个气泡图由两个序列组成,为了能够使用不同的点大小来绘制图表.气泡大小的编码方法,同时它们的最大和最小大小都可以指定.

编码方法

使用 EncodingMethod 属性来设置气泡的大小是否取决于直径和区域.当指定了气泡的大小,直径和区域都会被作为绘制区域整体的直径和整体区域的百分比来测量.反转这些值(使最小值大于最大值)会为小数值绘制大的气泡,而大数值绘制小的气泡.

EncodingMethod 属性是 Bubble 对象的一个属性,它可以通过 **ChartGroupsCollection Editor 的 Bubble** 节点来访问.

最大和最小大小

气泡的最大和最小大小可以通过 MaximumSize 和 MinimumSize 属性来分别指定.Bubble 对象的一些属性也可以通过 **ChartGroups Collection Editor 中的 Bubble** 节点来访问.

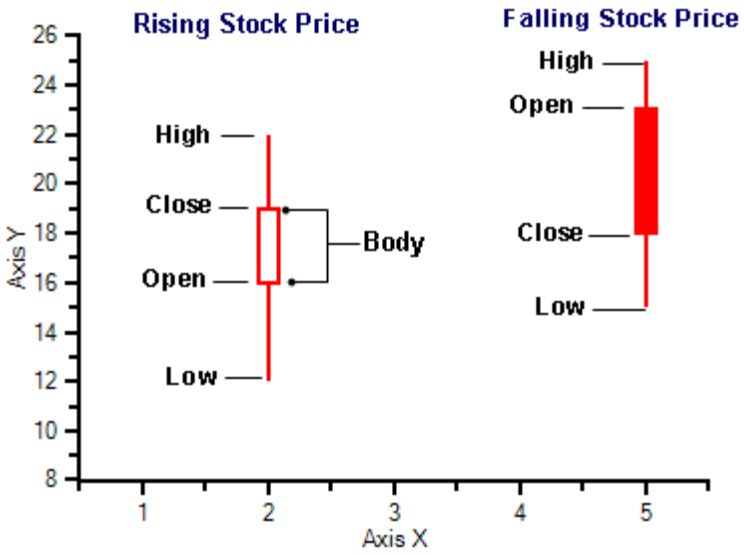
7.4 烛图

烛图,Hilo, Hilo 开闭是所有的股票图表,它们用在金融行业的应用中,来显示一个给定股票在开,闭,高,低时的价格.一个烛图是一个特殊的 Hilo 开闭图表,它用来显式开和闭,高和低的关系.类似于 Hilo 开闭图表,烛图使用相同的价格数据(高,低,开,闭值),除了它们包含了一个较厚的类似于蜡烛的主体,并且使用烛体的颜色来提取开和闭值之间关系的附加信息.

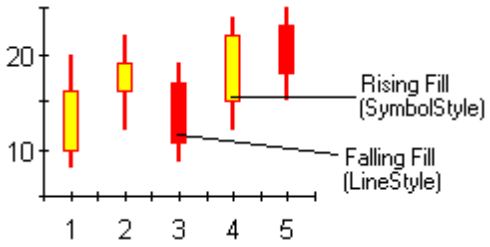
例如,长透明的蜡烛表示了购买压力,长的有填充色的蜡烛代表了出售压力.

烛图由以下的元素组成:烛体,灯芯,尾巴.烛体或者主体(开闭值之间的实心条状)代表从开到闭之间的股票价格的变化.在烛体上下的细线,灯芯,尾巴描述了高/低的范围.一个空的烛体或者透明烛体指示了一个上升的股票价格(闭时的价格高于开时).在一个空的烛体中,烛体的底部代表了开时的价格,烛体的顶部代表了闭时的价格.一个填充过的烛体代表了一个下降股票价格(开时的价格高于闭时的价格).在一个填充过的烛体中,烛体的顶部代表开时的价格,烛体的底部代表了闭时的价格.

C1Chart 使用 Y 值作为烛体高值,Y1 作为烛体低值,Y2 作为烛体开时的值,Y3 作为烛体闭时的值.C1Chart 使用线的颜色来自动填充下降烛体.



使用 LineStyle 和 SymbolStyle 属性和 HiLoData 类,每一个序列的填充和线条属性可以被自定义.关于此的更多信息,请参见特定的烛图属性(94 页).为了从开始到结束完整地了解使用设计器或者以编程方式创建一个烛图的过程,请参见烛图指南(281 页).



在设计时其设置图表类型为烛图

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **Candle.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Candle.**
- 另外一种可行的方法是在属性面板中选择图表属性，在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Candle.**

7.4.1 烛图编程时的讨论

下表描述了烛图使用的数据数组.每一个数据序列代表将要被创建的 Y,Y1,Y2 和 Y3 数组值.

属性	描述
X	持有 X 轴线的位置.
Y	持有烛图的高值.
Y1	持有烛图的低值.
Y2	持有烛图的开值.
Y3	持有烛图的闭值.

7.4.2 特殊的烛图属性

当 ChartType 属性被设置为 Candle 时,您可以使用以下的属性来指定一个上升或者下降的股票价格.

- FillFalling
- FillTransparent

展示下降价格

设置 FillFalling 属性为 True. ChartDataSeries.LineStyle 属性的值决定了填充时使用的颜色.

展示上升价格

设置 FillTransparent 为 True.这将会使得开烛图的烛体透明或者为空.如果您想为上升值指定一个特殊的颜色,那么将 FillTransparent 属性设置为 False,然后为 SymbolStyle 属性设置一个颜色.请注意推荐使用另外一种颜色,这样会让线条样式有一个颜色,而符号样式有另一种颜色.

关于此的更多信息,请参见为上升烛图创建一个**填充颜色**(352 页).

7.5 甘特图

一个甘特图用来描述大量任务的日程表,并且指出工程完成过程中的关键活动.

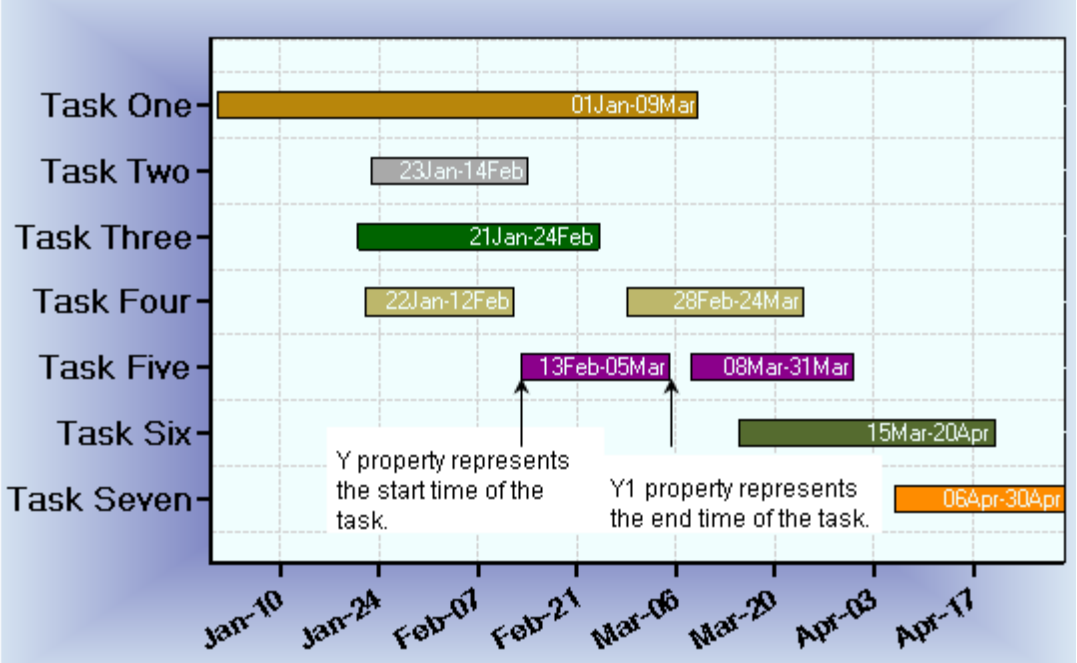
甘特图跟 **Bar** 和 **Hilo** 图表有以下的相似之处:

- 类似于 Bar 图表,甘特图使用条形,但是它通常作为翻转和颠倒条形图来显示.

- 类似于 Hilo 图表,每一个序列中的 Y 和 Y1 数组中元素代表了高值和低值.甘特图使用 Y 和 Y1 元素来代表一个任务的开始和完成时间.
  一个甘特图使用以下的方式来显示一个日程表:
- **活动/任务**
  活动/任务被显示在图表的左边,一个日程表被显示在图表的顶部或者底部.
- **任务有效期**
  每一个工程的任务都被作为一个条形来显示,条形的开始处指定了活动或者任务的开始时间,条形的结束处指定了活动或者任务的结束或者完成时间.
- **紧急活动或者成就**
  典型地,在甘特图中,特殊的图表或者颜色被用来代步在每一个任务的活动期间内的关键活动或者成就.

在一个甘特图中,您可以通过给图表数据序列添加一个新的序列来创建一个新的任务.在每一个序列中,都有一些属性可以用来在任务中给您的数据创建信息.例如,您可以使用 Y 属性来给您的任务的开始时间填充数据,并且可以使用 Y1 属性来记录您的任务的结束时间的数据.您可以从 ChartDataArray 对象的 DataType 属性中用在您的甘特图中的数据类型.

下面的甘特图展示了每一个任务的开始和结束时间.甘特图可以在一个任务中展示一个或者多个工程.在一些情况下,一个任务被指定了多个的子任务.每一个条状代表一个任务或者一个子任务.在一个轴线上包含一些条状可以跨越 X 和 Y 值的一个较大的范围.甘特图的数据在水平显示时会显得更加的吸引人,相比于垂直显示.所以,图表被倒置了,X 轴线是垂直的,而 Y 轴线是水平的.C1Chart 提供了 ChartArea 对象的 Inverted 属性和 Axis 的 Reversed 属性.请注意每一个水平条状被一个单色填充.通常地,这代表一个完成的任务.同时每一个水平图表含有一个指示了开始和技术时间的标签.这个标签可以在运行时或者设计时通过使用 ChartDataSeries 对象的 Lable 属性来添加到水平条形上.



在设计时其设置图表类型为甘特图

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **Gantt.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 **Gantt**.
- 另外一种可行的方法是在属性面板中选择图表属性，在 **Gallery** 中选择图表类型为 **Candle.**

7.5.1 甘特图编程时的考虑

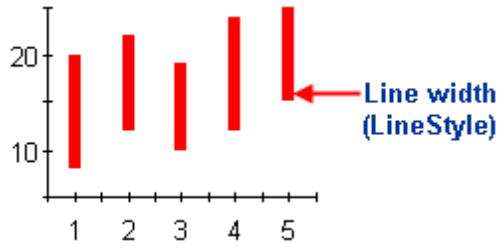
下表列出了在甘特图中的数据序列中的每一个元素和它们的效果.每一个数据序列都需要使用一个 X 数组和两个含有 Y 和 Y1 的 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换甘特图到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有工程的开始时间的开始值.
Y1	持有工程的结束时间的结束值.
Y2	对甘特图无效.
Y3	对甘特图无效.

7.6 HiLo 图表

一个 HiLo 图表含有两个独立的值来支持在一个序列中的高值和低值数据.HiLo 图表主要用在金融应用中,来显示一个给定股票的高价格和低价格.每一个序列中的 Y 和 Y1 数组元素代表高值,和低值.

使用 LineStyle,每一个序列的填充和线属性都可以被自定义,关于此的更多信息,请参见序列的线和符号样式(240).



在设计时其设置图表类型为 HiLo 图

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **HiLo.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Hi-low.**
- 另外一种可行的方法是在属性面板中选择图表属性，在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Hi-low.**

7.6.1 HiLo 图编程时的考虑

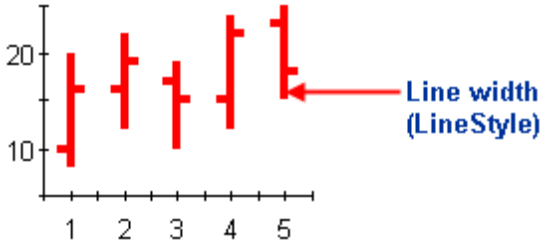
下表列出了在 HiLo 图中的数据序列中的每一个元素和它们的效果.每一个数据序列都需要使用一个 X 数组和两个含有 Y 和 Y1 的 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换 HiLo 图到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有图表的高值.
Y1	持有图表的低值.
Y2	对 HiLo 图无效.
Y3	对 HiLo 图无效.

7.7 HiLo 开闭图表

HiLo 开闭图表非常类似于 HiLo 图表,除了它组合四个独立的值来支持在一个序列中的一个点上的高,低,开,闭数据.除了显示一个股票的高值和低值之外,Y2 和 Y3 数组元素特别用来显示股票开盘和闭市时的价格.

使用 LineStyle,每一个序列的填充和线属性都可以被自定义,关于此的更多信息,请参见序列的线和符号样式(240).



在设计时其设置图表类型为 HiLo 开闭图

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **HiLoOpenClose.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Hi-low-open-close.**
- 另外一种可行的方法是在属性面板中选择图表属性, 在 Gallery 中选择图表类型为 Stock,并且图表子类型为 **Hi-low-open-close.**

7.7.1 HiLo 开闭图编程时的考虑

下表列出了 HiLo 开闭图使用的数据数组元素.每一个数据序列都需要一个含有 Y,Y1,Y2,Y3 值的 Y 数组值.

属性	描述
X	持有 X 轴线的位置.
Y	持有图表的高值.

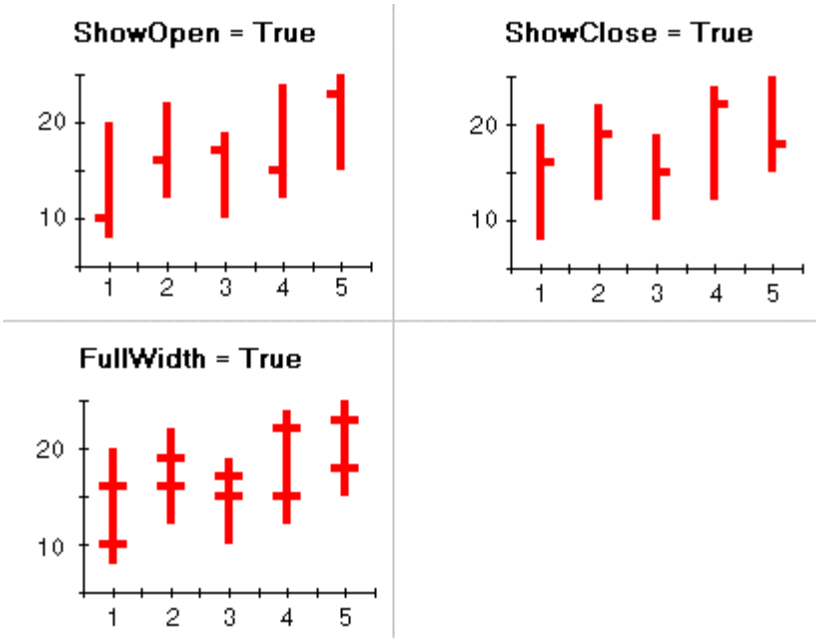
Y1	持有图表的低值.
Y2	持有图表的开值.
Y3	持有图表的闭值.

7.7.2 特殊的 HiLo 开闭图的属性

开发人员可以指定开闭刻度是如何显示在图表上的.

为了显示开闭记号

- **使用 ShowOpen, ShowClose 和 FullWidth 属性来指定开闭刻度是如何被显示的.**
- **启用 ShowOpen 来显示开刻度,启用 ShowClose 来显示闭刻度.**
- **启用 FullWidth 来在垂直范围的两端都显示开和闭刻度.**



格式化线条和刻度

- **查看 HLCandle2005 示例**

7.8 直方图

一个直方图持有一个未加工的数据值的集合,然后绘制频率分布.它经常被用在分组数据,这些数据由测量未加工的数据的集合而产生,并且绘制落在给定区间内的数据值的个数.请注意,未加工的值不会为直方图生成数据,但是它们被用来生成一个频率.它在显示时类似于条形图.非常重要的一点是直方图使用量化的变量.而同时条形图也经常使用量化的变量.

一个直方图能够过非常简洁地指出一个量化的变量在数据分布上的显著特征.量化变量的重要特征如下:

- 它展现典型的平均值.
- 数据产生一个通用的形状.数据值可以对称地分布在中间的两边.或者它们可以被斜交.
- 如果存在距离数据组比较远的值,那么将它们显示为离群值.
- 数据值可以远离或者靠近典型值.

- 分布可以造成单个的峰值或者多个峰值和谷地.

注意:间隔值代表数据元素的频率.有很多显示频率的技术.频率通常被显示为多边形,列,或者条形图.

在设计时其设置图表类型为直方图

- 在属性窗口中展开 ChartGroups 节点,在编辑器右边的面板上,设置 ChartType 属性为 **Histogram.**
- 另外一种改变图表类型的方法是展开.Net 属性窗口中的 Chart Properties,然后通过点击 ellipsis 按钮来打开 **ChartGroups Collection Editor**. 在编辑器的右边面板上.设置图表类型为 **Histogram.**

7.8.1 直方图编程时的考虑

不像其它的笛卡尔图表,一个直方图的输入不用指定 X 和 Y 坐标.相反地,一个单值的未加工数据的数组在每一个图表数据序列的 Y 图表数据数组中指定.根据 IntervalCreationMethod 的选择,直方图输入可以使用图表数据序列中的 X 图表数据数组来指定区间范围.

下表列出了直方图中使用的数据数组,并描述了每一个元素的效果.

属性	描述
X	和 XdataBoundaries 一起作为 IntervalCreationMethod 来持有直方图的间隔范围.
Y	持有未加工的数据值.
Y1	对直方图无效.
Y2	对直方图无效.
Y3	对直方图无效.

7.8.2 直方图的类型

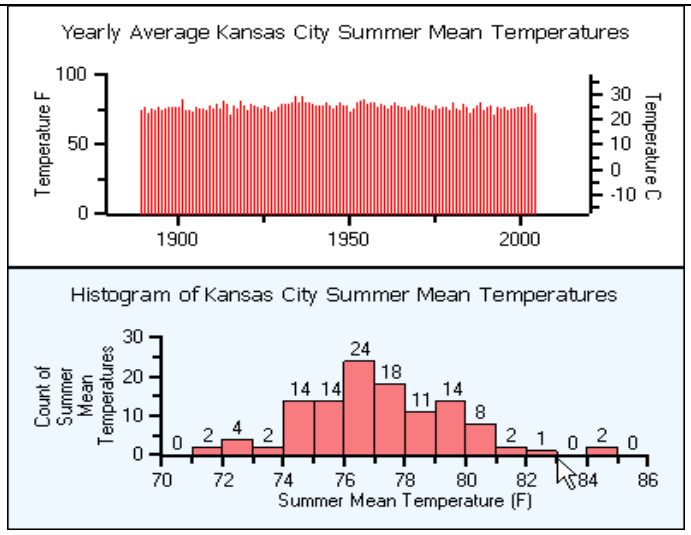
直方图含有集中图表类型,C1Chart 提供了以下类型的直方图:

- 直方图
- 频率图
- 步进频率图

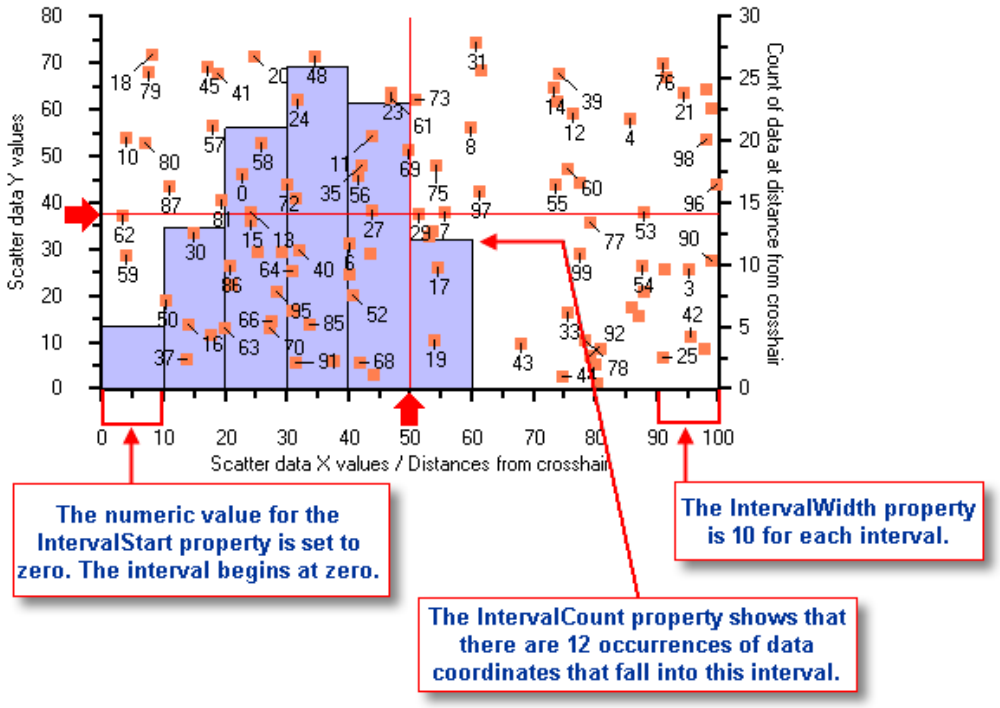
7.8.2.1 直方图

一个直方图的外观非常类似于一个条形图.它们在外观上的一个微小的差异在于,条形图中的矩形经常有一个空间上的间隔,但是直方图上的矩形不会被分割.直方图的列不是分离的,因为列边界代表了真实的 X 轴线的值.虽然直方图和条状图的外观非常相像,但是它们的功能是不同的.一个条状图是由数据点创建的,但是直方图从数据的频率分布中创建.

下面的图表展示了一个直方图和一个条形图的区别.两个图表都使用相同的 Y 数据.条形图(上面的图)显示了每一个年的平均气温.直方图(下面的图)使用相同的输入温度数据,并自动地列出了落在每一个间隔中的温度的数量,然后绘制结果直方图.为了一致性,每一个间隔的数量图表标签并添加在每一个间隔的顶部.



下图展示了一个直方图的详细和属性.它是一个展示一个图表组中含有一百万个随机数据坐标的散点图表,然后在第二个图表组中,展示了一个基于每一个散布的数据点到标记的交叉之间的距离的直方图.



在上面的直方图中, SemiAutomatic 方法被用在 IntervalCreationMethod 属性中来指定超过和低于的限制,还有间隔的数量.当您需要设置不同的间隔边界时 IntervalCreationMethod 属性是很有用的.您可以选择在 IntervalCreationMethod 属性中选择以下三个方法中的一个:

- **SemiAutomatic**

当使用 **SemiAutomatic** 方法时,间隔的超过和低于限制和间隔的数量一起被指定.间隔边界被统一地计算.当您选择 SemiAutomatic 方法时 IntervalStart, IntervalWidth,和 IntervalNumber 属性是可用的. IntervalStart 属性用来获取或者设置第一个间隔开始处的数量值.在上面的直方图中, IntervalStart 属性被设置为 zero.所以,间隔在值 zero 处开始.您可以使用 IntervalWidth 属性来

指定间隔的宽度为一个指定的宽度大小.在上面的示例中, IntervalWidth 属性为每一个间隔设置了 10 的大小.IntervalWidth 属性将会返回在一个直方图中间隔的数量.例如,在上面的直方图中,IntervalWidth 属性设置为 10,所以,最多可以含有 10 个间隔,剩余的四个间隔不会显示在上面的图表中,即使它们含有 zero 数据元素.

- **Automatic**

当使用 Automatic 方法时,图表使用 maximum 和 minimum 数据值来计算间隔的超过和低于限制,并且限制间隔位于数据平均值中的 3 个标准偏差中.间隔的数目是可选的.间隔边界被统一地计算.

- **XDataBoundaries**

当使用XdataBoundaries方法时,数据序列的X值会被用来显式地设置每一个间隔边界.X值被排序过了,所以重复值会被过滤掉.每一个结果的上升值被用来决定下一个间隔的边界.所以,第一个和第二个结果的X值定义了第一个间隔,然后每一个连续的X值指定了下一个间隔的结束.请注意指定N个间隔需要N+1个唯一的X值.

为了在设计时访问 IntervalCreationMethod 属性,展开 ChartData 节点,并且点击紧挨着 SeriesList 节点的 ellipsis 按钮**, ChartDataSeries Collection Editor** 将会出现.定位在 **Histogram** 节点上并展开它.选择 IntervalCreationMethod 属性并选择从 IntervalCreationMethod 属性的下拉列表框中选择三个方法中的一个. IntervalCreationMethod 属性可以通过以下的代码来访问:

- Visual Basic

```
cds.Histogram.IntervalCreationMethod = IntervalMethodEnum.SemiAutomatic
```

- C#

```
cds.Histogram.IntervalCreationMethod = IntervalMethodEnum.SemiAutomatic;
```

请注意,上面的"cds"变量,是一个 ChartDataSeries 对象的变量名称.在上面的示例代码中,使用了 SemiAutomatic 方法.您还可以通过指定 IntervalMethodEnum 枚举中的其它值来指定使用 Automatic 或者 XdataBoundaries 方法.

您可以通过使用 ChartHistogram 对象的 DisplayType 属性来一个一个直方图,频率图,步进频率图的形式来显示间隔和数量.如下述代码所示:

- Visual Basic

```
'Displays the chart histogram as a Histogram
ch.DisplayType = DisplayTypeEnum.Histogram
'Displays the chart histogram as a frequency graph
ch.DisplayType = DisplayTypeEnum.FrequencyGraph
'Displays the chart histogram as a stepped frequency graph
ch.DisplayType = DisplayTypeEnum.SteppedFrequencyGraph
```

- C#

```
    //Displays the chart histogram as a Histogram
    ch.DisplayType = DisplayTypeEnum.Histogram;
    //Displays the chart histogram as a frequency graph
    ch.DisplayType = DisplayTypeEnum.FrequencyGraph;
    //Displays the chart histogram as a stepped frequency graph
    ch.DisplayType = DisplayTypeEnum.SteppedFrequencyGraph;
```

注意:上面的"ch"变量,是一个 **ChartHistogram** 对象的变量名称

有些情况下一些数据元素出现在离其它数据元素较远的地方.在这些场景下,距离值被引用到一个离群值上.在最常见的情况下,离群值不会落在一个间隔中,因为它离其它的数据距离较远.所以,数据值将会丢失.

为了防止离群值被排除, C1Chart 的 ChartHistogram 对象提供了以下两个属性:

- **AboveIntervalCount**

  AboveIntervalCount属性返回比最后一个间隔的结束值还大的值的数量.

- **BelowIntervalCount**

  BelowIntervalCount属性返回比第一个间隔的开始值还要小的值的数量.

在设计器中您可以在 **ChartDataSeries Collection Editor** 中的 Histogram 节点中找到 AboveIntervalCount 和 BelowIntervalCount 属性.下面的代码展示了如何使用 AboveIntervalCount 来查看是否有一些离群值.

- Visual Basic

```
    Dim overflow As Integer = _ CInt(cds.Histogram.AboveIntervalCount)
    Dim msg As String = ""
    ' this tests to see if there are any outlier values that fall after the last
interval.
    If overflow > 0 Then
    msg = "Number > " + carea.AxisX.Max.ToString() + " = " + overflow.ToString()
    End If
    c1Chart1.ChartLabels("overflow").Text = msg
```

- C#

```
    int overflow = (int)cds.Histogram.AboveIntervalCount;
    string msg = "";
    // this tests to see if there are any outlier values that fall after the last
interval.
    if(overflow > 0)
    {
    msg = "Number > " + carea.AxisX.Max.ToString() + " = " + overflow.ToString();
    }
    c1Chart1.ChartLabels["overflow"].Text = msg;
```

在直方图中,列或者间隔的区域是与其所代表的值是成比例的.当所有的间隔的间隔宽度都

相同时,每一个间隔高度代表了在相同单位宽度下的相同频率.在一些情况下列的宽度在大小上差异很大.当这种情况发生时,列的高度必须被调整,以保持区域是成比例的.例如,如果您含有几个不统一的列,您可以使用 C1Chart 的 Histogram 对象的 Normalized 属性来使得列或者间隔与其代表的值是成比例的.

直方图对象含有三个属性: NormalDisplay, NormalizationInterval,和 Normalized. Normalized 属性可以被用来将不均匀的间隔变成标准化的间隔,以让每一个间隔高度代表在单位宽度下的相同频率.一个不均匀的间隔是一个跟其它间隔不统一的间隔.当您设置 Normalized 属性为 True 时,您可以通过 NormalizationInterval 属性来灵活地设置宽度大小.当您想把 IntervalWidth 从 10 设置为 20 时,将会造成每一个间隔的大小成倍增加.并且同时会造成每一个频率数目(y 值)加倍.如果您想自定义您的 IntervalWidth 属性从 2 到 10,您可以将间隔的宽度缩小为原先的 1/5.这样同时会造成每一个频率数目(Y 值)变成原先的 1/5 大小.

7.8.2.2 频率图

一个频率图是一个没有列的直方图.每一个列的中间点使用一个直线或者曲线连接.所以,列被消除了.在 C1Chart 中,连接一个列的中间点和另一个列的中间点的线可以使线条或者曲线.在您对比多个数据序列时频率图非常有用,因为它让数据变得更加的可读.

下面的频率图显示了所有数据坐标中从每一个散列的数据点到标记的交叉点之间的距离.散列点上附加的数字是点索引值.

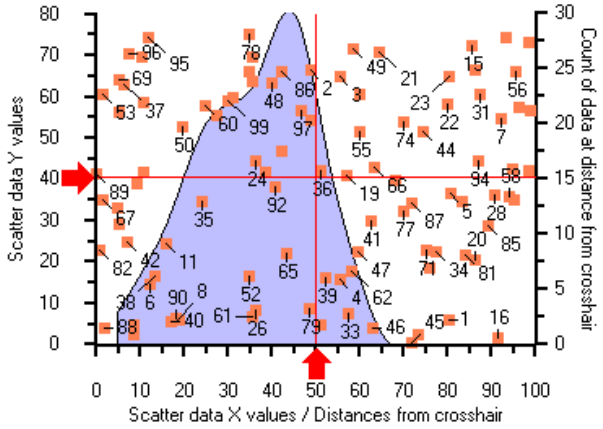
您可以通过设置 FitType 属性为 Line 来指定在频率图中使用线条连接.同样地,曲线可以使用 Spline 类型的 FitType 值来指定.一个频率图的特点是它显示了分布的重要特性,并且没有分散在角落上.下面的代码演示了将频率图的 FitType 属性设置为 Spline.

- Visual Basic

```
cds.FitType = FitTypeEnum.Spline
```

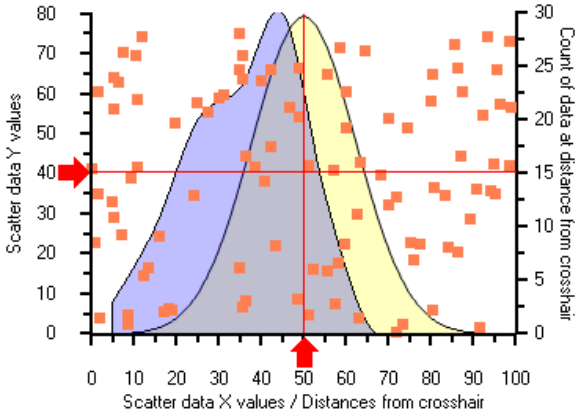
- C#

```
cds.FitType = FitTypeEnum.Spline;
```



下面的频率图跟上面的频率图是一致的,除了它含有一个正交的曲线来使用正交距离分布来对比距离分布.通过显示的正交曲线,我们对原生数据是否是正交分布的一个合理近似做出判断.

原生数据标签被隐藏了,以显示一个对两个曲线都清晰的图像.



图表中的 Histogram 对象提供了一个 NormalDisplay 属性来获取正交(高斯)曲线来显示.您可以在设计时,通过展开 C1Chart 属性表格中 histogram 节点,并且设置 NormalDisplay 属性为 **C1.Win.C1Chart.NormalCurve** 来做到这一点.另外一种可选的方式是通过以下的代码来访问 NormalDisplay 属性:

- Visual Basic

```
Dim nc As NormalCurve = cg.Histogram.NormalDisplay
```

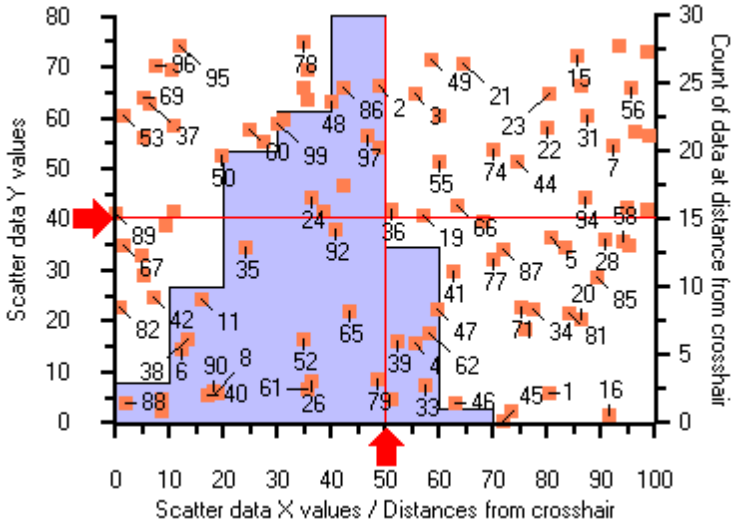
- C#

```
NormalCurve nc = cg.Histogram.NormalDisplay;
```

注意:nc,这个文字,是 NormalCurve 对象的名字.

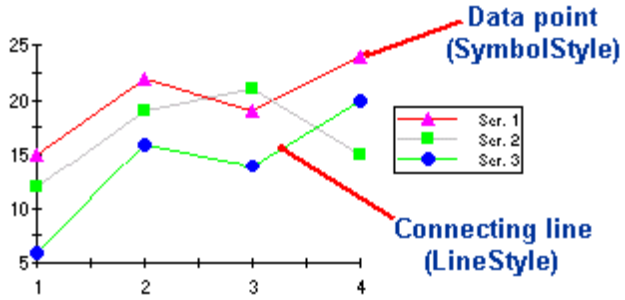
7.8.2.3 步进频率图

一个步进频率图在功能上类似于一个直方图.这两个图表唯一的区别在外观上.在步进频率图中列之间的线条被消除了,但是在直方图中列之间的线还保持着.

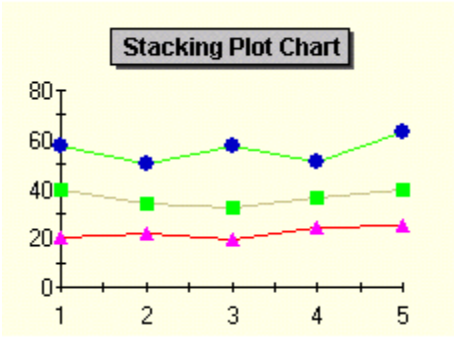


7.9 线和 XY-Plot 图表

线和 XY-Plot 图表以数据的连接点方式来绘制每一个序列.通过自定义每一个序列的线和符号样式,连接线可以被消除,从而突出数据值本身,或者点可以被消除来突出点之间的关系.序列可以独立或者叠加绘制.每一个序列的线和符号属性同样可以被自定义.关于更多的信息,请参见序列的线和符号样式(240 页).



使用 ChartGroup 对象的 Stacked 属性来创建一个叠加的绘制图表.叠加图表通过在前一个序列的值的顶部来叠加显示每一个序列的值来显示数据.



在设计时其设置图表类型为线或 XY-Plot 图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 **ChartType** 属性为**. XY-Plot** 线图表选项在这种方式下是不可用的.
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 **Line** 或者 **XY-Plot.**
- 另外一种可行的方法是在属性面板的底部中选择图表属性, 在 **Gallery** 中选择图表类型为 **Line** 或者 **XY-Plot**

7.9.1 绘制图表编程时的考虑

下表列出了在线和 XY-Plot 图中的数据序列中的每一个元素.每一个数据序列都需要使用一个 X 数组和 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有 Y 轴线的位置.
Y1	对线和 XY-Plot 图表无效.

Y2	对线和 XY-Plot 图表无效.
Y3	对线和 XY-Plot 图表无效.

7.9.2 XY-Plot 图表的 3D 效果

C1Chart 的 3D 效果可以用在 XY-Plot 或者叠加 XY-Plot 上来创建每一个序列在高度上的假象.有些时候,这些图表被称为 Ribbon 图表.通过使用深度,海拔,旋转,和阴影属性,您可以增强您的区域图表的效果,通过创建视觉深度来突出它们.

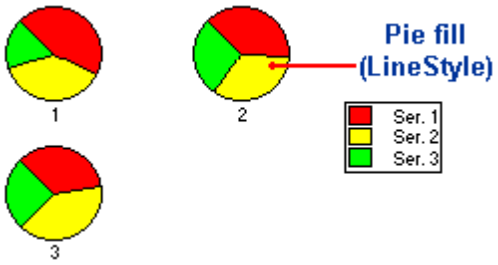
为了访问一个 XY-Plot 图表的 3D 视图,调整 View3D 对象的一些属性 View3D 对象是 PlotArea 对象的一个成员,然后 PlotArea 又是 ChartArea 对象的一个成员.通过调整 View3D 对象的属性,深度,高地,旋转,阴影,您可以自定义 3D 视图.

请注意 Depth 属性是所有 3D 图表类型逻辑的关键. Elevation 和 Rotation 属性修改用户查看图表的方式.所以正是 Depth 属性实际上支配了一个图表是否是 3D 的.通过为 Depth 属性使用一个非 0 值,并且设置 Elevation 和 Rotation 属性的值为 0.您可以创建一个 3D 图表,虽然什么事情起来都没有改变.实际上您只是在看图表的正前方表面,就像一个标准的区域图表的视觉效果一样.

同时请注意,对一些数据进行 3D 视图的展示的效果可能是很令人满意的,但是同时其它的数据需要使用 2D 的面板.对于这些场景,调整附加在每一个 ChartGroup 上的 Use3D 属性.

7.10 饼状和圆环图表

一个饼状图表以一个饼形中的一个切片的方式来绘制每一个序列.切片的数目就是数据中点的数目.每一个切片显示每一个序列中的第 N 个数据点.使用 LineStyle 属性,每一个序列的填充属性可以被自定义.关于此的更多信息,请参见序列的线和符号样式(240 页).



在设计时其设置图表类型为饼状图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 ChartType 属性为 **Pie.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 **Pie.**
- 另外一种可行的方法是在属性面板的底部中选择图表属性，在 Gallery 中选择图表类型为 **Pie**

7.10.1 饼状图表编程时的考虑

下表列出了在饼状图表中的数据序列中的每一个元素.每一个数据序列都需要使用一个 X 数组和 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换图表

到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

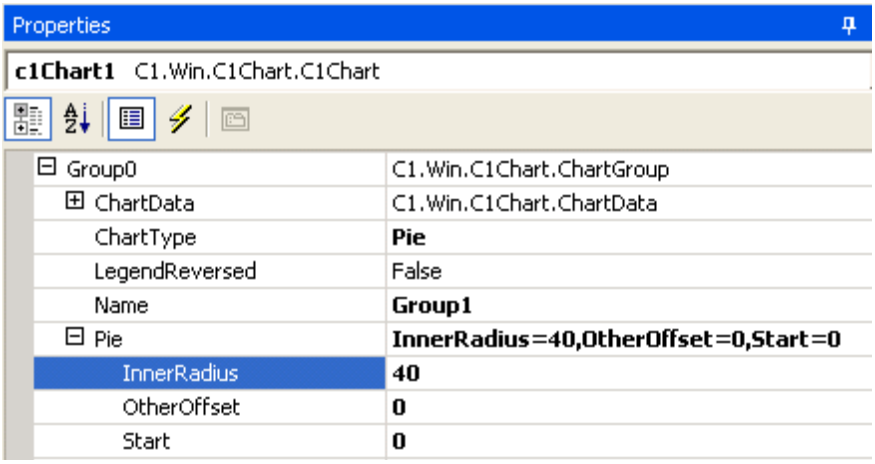
属性	描述
X	持有饼状图表中的以 0 开始的切片索引.
Y	持有由 X 值指定的切片的值.
Y1	对饼状图表无效.
Y2	对饼状图表无效.
Y3	对饼状图表无效.

注意:在多个序列中输入数据会导致在一个图表组中显示多个图表切片.

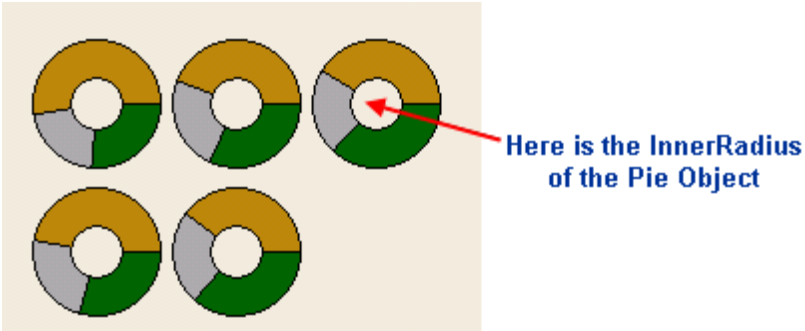
7.10.2 圆环图表

一个 ComponentOne 圆环图表带有非零半径的饼状图表,其在功能上跟一个饼状图表是一致的.但是它可以被用来增加视觉效果,特别是用来显式 3D 效果时.同所有的饼状图表一样,每一个圆环以整体的一个片段的方式显式每一个序列上的数据点.如果指定了多个数据点,那么在图表中将会出现多个圆环.

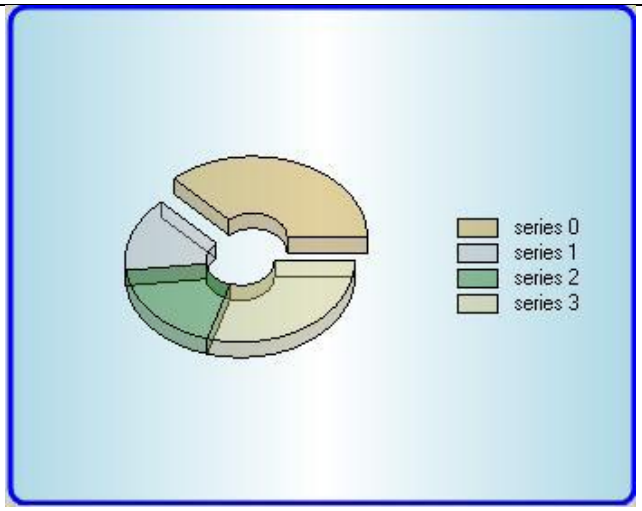
可以通过设置一个饼状图表的 InnerRadius 属性值为非零来创建一个圆环图. InnerRadius 属性的值代表相对于整个饼状半径的百分比.可以在每一个图表组的 Pie 对象中访问 InnerRadius 属性.下面显示了 InnerRadius 属性.在这个示例中 InnerRadius 属性被设置为 40%.



下面的图像展示了将 InnerRadius 属性设置为 40%是的饼状图表的效果.



下面是另外的一个圆环图表的示例.在这个示例中 ChartDataSeries 对象的 OffSet 属性设置为 30.



7.10.3 特殊的饼状图表属性

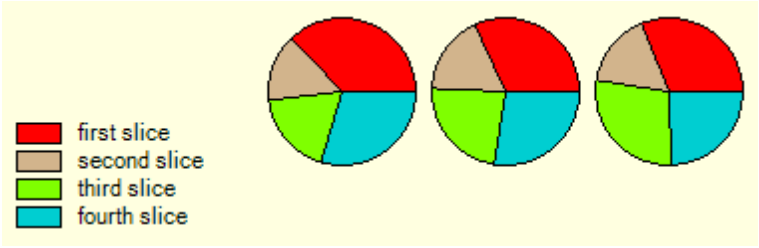
饼状图表与其它图表类型有很大的差异,因为它没有两维网格或者轴线的概念.改变饼状的直径或者分离图表的属性,可以通过 Pie 类的属性来完成.

顺时针或者逆时针方向

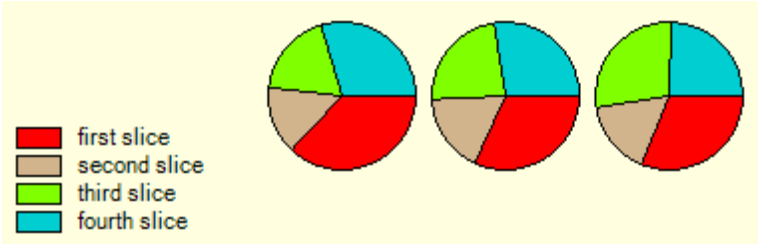
设置 Clockwise 属性为 True 会以顺时针方向在一个饼状中绘制每一个序列,设置为 False 会在饼状中以逆时针方向绘制每一个序列

下图展示了两个饼状图表,一个以顺时针方向绘制,另一个以逆时针方向绘制.

顺时针方向(默认)

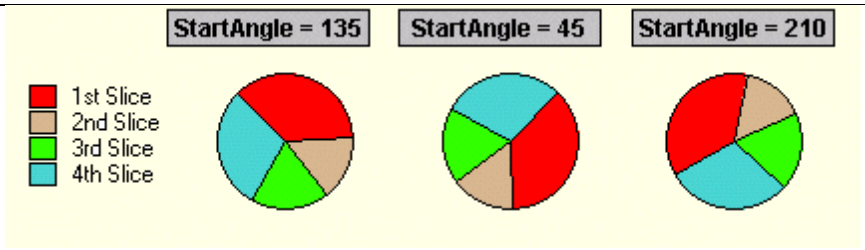


逆时针方向



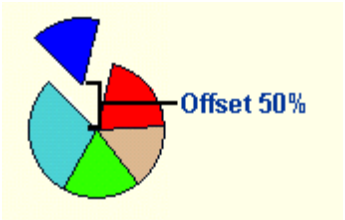
开始角度

使用 Start 属性来指定第一个序列开始时切片的角度.默认的角度是 0 度.角度代表第一个切片的最左上边缘和饼状的右边水平半径之间的弧度,以反时针方向测量(见上图).Pie 类中的 Start 属性可以在设计时中在 ChartGroupsCollection Editor 中的 Pie 节点中访问.



分离饼状图

饼状图表中的一个切片可以通过被分离来强调显示,分离的效果是该切片从饼状图中的其它切片中伸出.使用序列的Offset属性来设置切片到饼状的中间位置的偏移.偏移以饼状的半径百分比的方式测量.



分离切片可以以编程的方式进行设置,但是仅能在序列上设置.

- Visual Basic

```
    'Get the appropriate ChartData object.
    Dim cd As ChartData =
C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData
    'Sets the offset for the first series to 10% of the pie's radius
    cd.SeriesList(0).Offset = 10
    'Resets the exploded slices.
    cd.SeriesList(0).Offset = 0
```

- C#

```
    //Get the appropriate ChartData object.
    ChartData cd = c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData;
    //Sets the offset for the first series to 10% of the pie's radius
    cd.SeriesList[0].Offset = 10;
    //Resets the exploded slices.
    cd.SeriesList[0].Offset = 0;
```

7.10.4 饼状图表的 3D 效果

C1Chart 的 3D 效果可以用在饼状图表上来创建每一个序列在高度上的假象.有些时候,这些图表被称为 Ribbon 图表.通过使用深度,海拔,旋转,和阴影属性,您可以增强您的区域图表的效果,通过创建视觉深度来突出它们.

为了访问一个饼状图表的 3D 视图,调整 View3D 对象的一些属性 View3D 对象是 PlotArea 对象的一个成员,然后 PlotArea 又是 ChartArea 对象的一个成员.通过调整 View3D 对象的属性,深度,高地,旋转,阴影,您可以自定义 3D 视图.

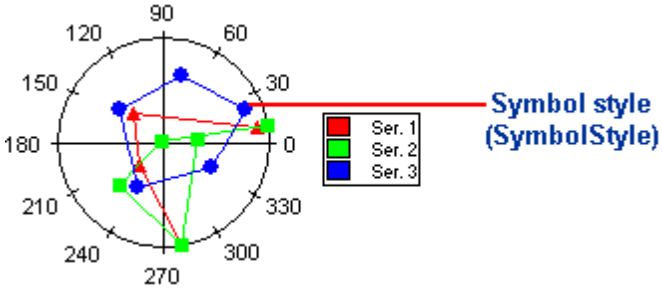
请注意 Depth 属性是所有 3D 图表类型逻辑的关键. Elevation 和 Rotation 属性修改用户查看图表的方式.所以正是 Depth 属性实际上支配了一个图表是否是 3D 的.通过为 Depth 属性使用一个非 0 值,并且设置 Elevation 和 Rotation 属性的值为 0.您可以创建一个 3D 图表,虽然什么事情起来都没有改变.实际上您只是在看图表的正前方表面,就像一个标准的区域图表的视觉效果一样.

同时请注意,对一些数据进行 3D 视图的展示的效果可能是很令人满意的,但是同时其它的数据需要使用 2D 的面板.对于这些场景,调整附加在每一个 ChartGroup 上的 Use3D 属性.

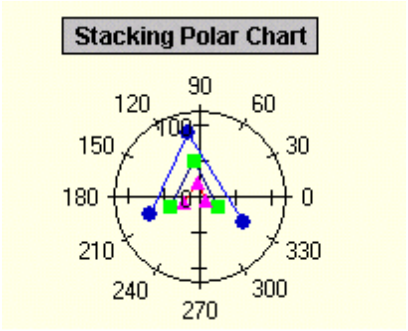
7.11 极地图表

一个极地图表以(theta,r)的形式绘制每一个序列中的 X 和 Y 坐标.其中,theta 是从开始的旋转,r 是从开始的距离.theta 可以通过角度(默认方式)或者弧度来指定.因为 X 轴线是一个圆,所以 X 轴线的最大和最小值是固定的.序列可以独立或者叠加地显示.

使用 LineStyle 属性,每一个序列的填充属性可以被自定义.关于此的更多信息,请参见序列的线和符号样式(240 页).



使用 ChartGroup 对象的 Stacked 属性来创建一个叠加的极地图表.叠加图表通过在前一个序列的值的顶部来叠加显示每一个序列的值来显示数据.



在设计时其设置图表类型为极地图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **Polar.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 **Polar.**
- 另外一种可行的方法是在属性面板的底部中选择图表属性，在 Gallery 中选择图表类型为 **Polar**.

7.11.1 极地图表编程时的考虑

下表列出了在极地图表中的数据序列中的每一个元素.每一个数据序列都需要使用一个 X 数组和 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	在角度或者弧度值中持有 X 轴线的位置.
Y	持有 Y 轴线的位置.
Y1	对极地图表无效.
Y2	对极地图表无效.
Y3	对极地图表无效.

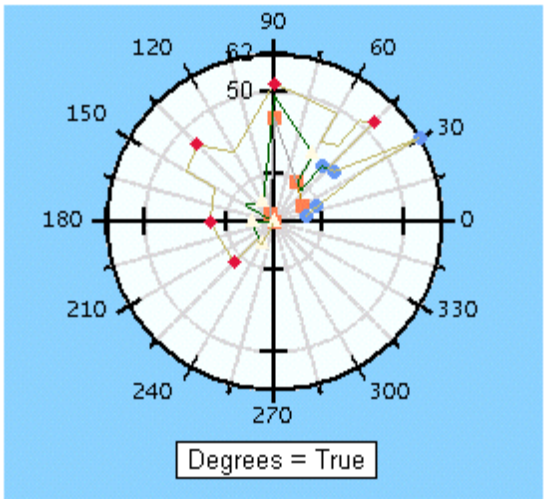
7.11.2 特殊的极地图表属性

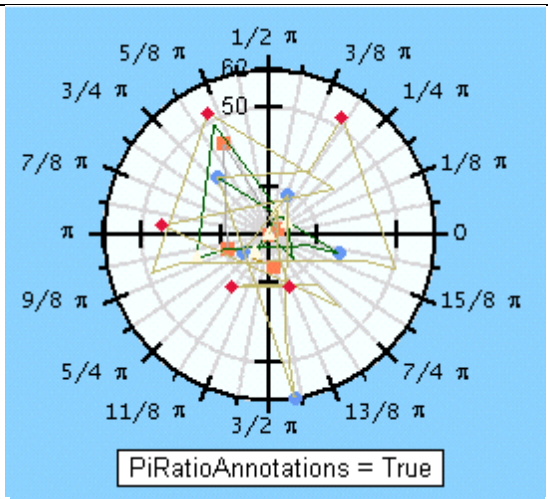
在设计时通过 ChartGroupsCollection Editor 的 polar 节点可以访问下面的属性.

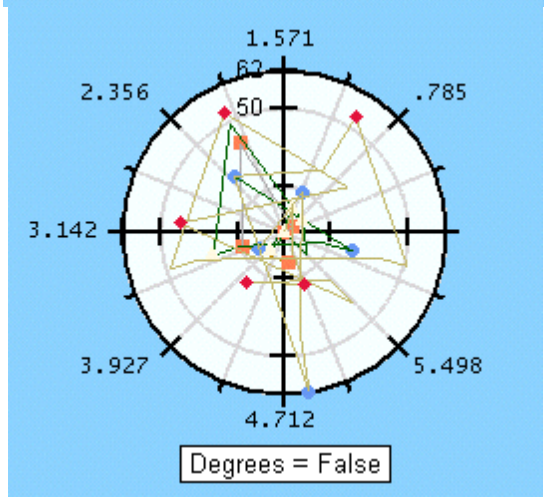
图表的角度或者弧度

Polar 类的 Degrees 属性用来设置极地图表的 X 数据值以角度(true)或者弧度(False)的方式反映角度. 在设计时通过 ChartGroupsCollection Editor 的 polar 节点可以访问到 Degrees 属性.

如果 Degrees 属性被设置为 False,那么图表将会反映弧度值.C1Chart 提供了以圆周率的比例而不是弧度的方式来注解图表.设置 Polar 类的 PiRatioAnnotations 属性为 True 将会以圆周率的比例的方式来注解 X 值.







设置开始角度

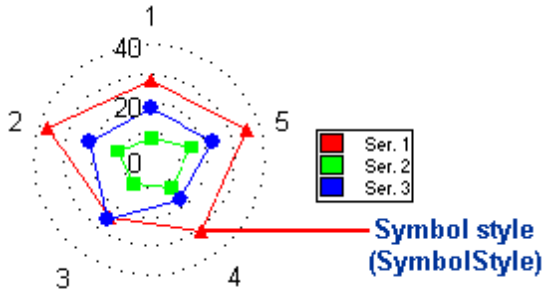
使用 Polar 类的 Start 属性来指定极地图表的开始角度.默认的角度是 0 度.设置一个比 0 大的角度值,会以指定的量以逆时针方向移动图表的开始处.例如,设置 Start 属性值为 90 度,将会以逆时针方向移动极地图表 90 度.

7.12 雷达图

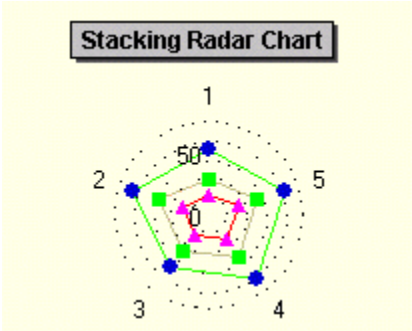
一个雷达图以一个雷达警戒线(除了标签外 X 值被忽略)的方式绘制每一个数据中的 Y 值.如果数据含有 n 个唯一的点,那么图表平面会被分割成 n 个相同角度的分段,然后会在每一个 n/360 角度增量处绘制一个雷达警戒线(代表每一个点).默认地,代表第一个点的雷达会被垂直地绘制(在 90 度上).序列可以独立地或者叠加绘制.

雷达图可以在辐射方向上随着 X 轴主要网格线来显示刻度线,刻度线由 Y 轴线的刻度属性控制.关于刻度线属性的更多信息,请参见轴线刻度线(207 页).

使用 LineStyle 属性,每一个序列的填充属性可以被自定义.关于此的更多信息,请参见序列的线和符号样式(240 页).



使用 ChartGroup 对象的 Stacked 属性来创建一个叠加的雷达图表.叠加图表通过在前一个序列的值的顶部来叠加显示每一个序列的值来显示数据.



在设计时其设置图表类型为雷达图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor.**在编辑器右边的面板上,设置 ChartType 属性为 **Radar.**
- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 Gallery 中选择图表类型为 **Radar.**
- 另外一种可行的方法是在属性面板的底部中选择图表属性，在 Gallery 中选择图表类型为 **Radar.**

7.12.1 雷达图表编程时的考虑

下表列出了在雷达图表中的数据序列中的每一个元素.每一个数据序列仅使用一个 Y 数组.给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	除了标签外都被忽略
Y	持有雷达图的 Y 轴线数据,Y 数据数组中点的数目决定了图表中雷达警戒线的数目.
Y1	对雷达图表无效.
Y2	对雷达图表无效.
Y3	对雷达图表无效.

7.12.2 特殊的雷达图属性

雷达图含有一些特殊的属性,这些属性可以用来设置雷达的角度,设置起始角度,创建填充的雷达图,和决定是否在雷达图中使用平面 Y 坐标网格线.

图表的角度或者弧度

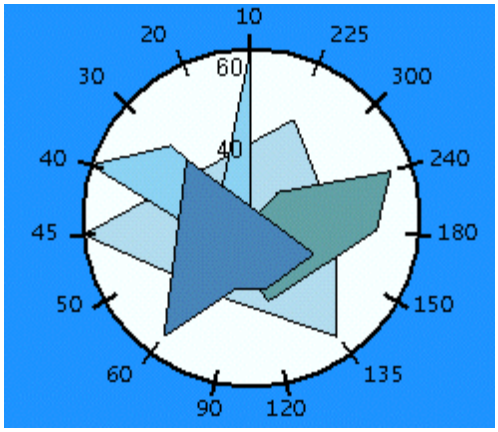
Radar 类的 Degrees 属性用来设置雷达图表的 X 数据值以角度(true)或者弧度(False)的方式反映角度. 在设计时通过 **ChartGroupsCollection Editor** 的 Radar 节点可以访问到 Degrees 属性.

设置开始角度

使用 Radar 类的 Start 属性来指定雷达图表的开始角度.默认的角度是 0 度.设置一个比 0 大的角度值,会以指定的量以逆时针方向移动图表的开始处.例如,设置 Start 属性值为 90 度,将会以逆时针方向移动雷达图表 90 度.

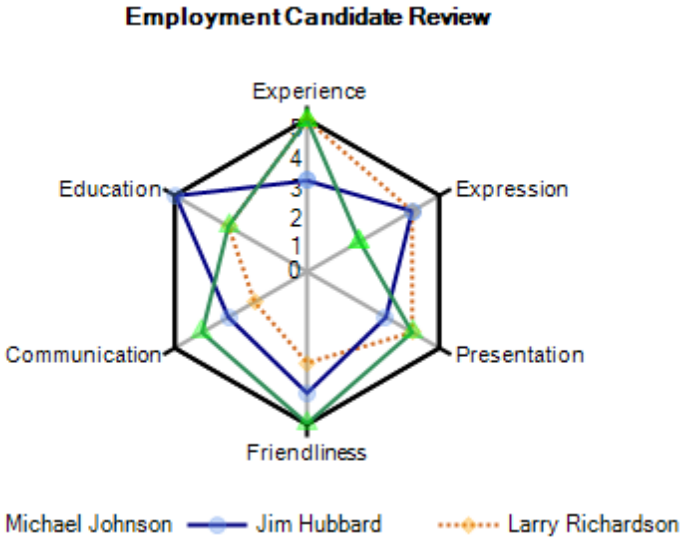
填充的雷达图

一个填充的雷达图以一个雷达警戒线(除了标签外 X 值被忽略)的方式绘制每一个数据中的 Y 值.如果数据含有 n 个唯一的点,那么图表平面会被分割成 n 个相同角度的分段,然后会在每一个 n/360 角度增量处绘制一个雷达警戒线(代表每一个点).每一个序列被绘制在前一个序列的"顶部",序列可以独立地或者叠加绘制.填充的雷达图和雷达图是类似的,除了在开始出和点之间的空间是被填充的,并且符号不会被显示.为了创建一个填充的雷达图,设置 Radar 类的 Filled 属性为 True.



平面网格线

您可以通过设置 FlatGridLines 属性为 True 来在雷达图中显示平面的 Y 坐标网格线.



下面的代码展示了以编程的方式设置该属性的值.

- Visual Basic

```
C1Chart1.ChartGroups(0).Radar.FlatGridLines = True
```

- C#

```
c1Chart1.ChartGroups[0].Radar.FlatGridLines = true;
```

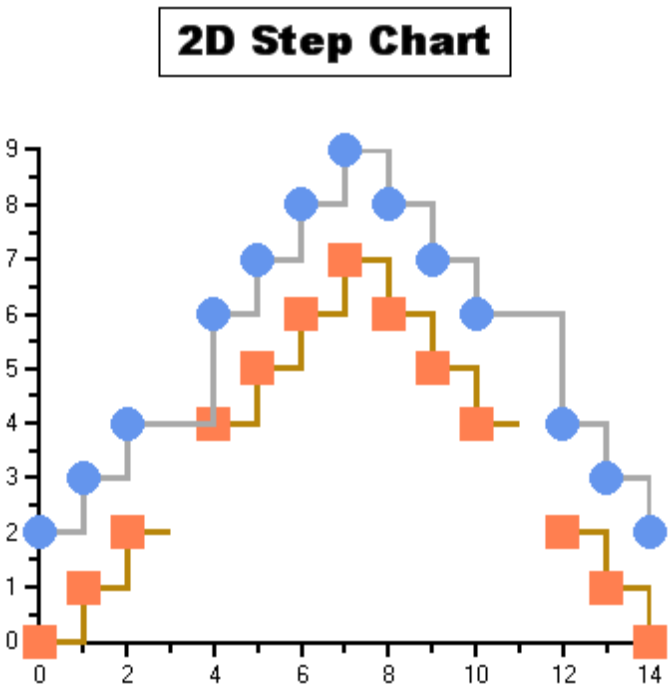
7.13 步进图表

一个步进图表是 XY-Plot 图表的另一种形式.步进图表经常在 Y 值以离散量改变的场景下被使用,在一些 X 上值会有一些突然的改变的时候.一个简单,日常的例子是绘制一个带有时间的支票薄收支平衡的图表.当每一次储蓄发生,每一次使用支票,支票登记薄的平衡(Y 值)会随着时间(X 值)流逝而突然的改变,而非逐步的改变.在一些没有储蓄发生或者使用支票的时期,平衡(Y 值)随着时间流逝依然保持不变.

类似于线和 XY 绘制,步进图表的外观可以被自定义,通过为每一个序列使用线和符号样式,来改变颜色,符号大小,和线的厚度.符号可以被完全的消除从而增强突出点之间的关系,或者可以包含符号来指出真实的数据值.如果出现数据洞,步进图表表现的像预期一样,在数据洞的 X 值上使用序列线来表明已知的信息. 符号和线会在非空洞的数据再次出现时继续显示.

像大多数的 XY 样式绘制图表一样,步进图表可以在恰当的时机叠加显示.

下面的图表展示了 2D 步进图表:



在设计时其设置图表类型为步进图表类型

- 在属性窗口中展开 ChartGroups 节点,通过点击 ellipsos 按钮来打开 **ChartGroups Collection Editor**.在编辑器右边的面板上,设置 **ChartType** 属性为 **Step.**

- 另外一种改变图表类型的方法是右键点击既存的图表,并且选择 Chart Properties. 在 **Gallery** 中选择图表类型为 **Step.**
- 另外一种可行的方法是在属性面板中选择图表属性，在 **Gallery** 中选择图表类型为 **Step.**

7.13.1 步进图表编程时的考虑

下表列出了在步进图表中的数据序列中的每一个元素.每一个数据序列仅使用一个 Y 数组. 给其它数组添加值不会影响这个图表,但是给这些数组填充数据可能使得在转换图表到其它图表类型时的工作变得简单,如果其它图表类型使用这些数组的话.

属性	描述
X	持有 X 轴线的位置.
Y	持有 Y 轴线的位置.
Y1	对步进图表无效.
Y2	对步进图表无效.
Y3	对步进图表无效.

7.13.2 步进图表的 3D 效果

C1Chart 的 3D 效果可以用在步进图表上来创建每一个序列在高度上的假象.通过使用深度,海拔,旋转,和阴影属性,您可以增强您的区域图表的效果,通过创建视觉深度来突出它们.

为了访问一个区域图表的 3D 视图,调整 View3D 对象的一些属性 View3D 对象是 PlotArea 对象的一个成员,然后 PlotArea 又是 ChartArea 对象的一个成员.通过调整 View3D 对象的属性,深度,高地,旋转,阴影,您可以自定义 3D 视图.

请注意 Depth 属性是所有 3D 图表类型逻辑的关键. Elevation 和 Rotation 属性修改用户查看图表的方式.所以正是 Depth 属性实际上支配了一个图表是否是 3D 的.通过为 Depth 属性使用一个非 0 值,并且设置 Elevation 和 Rotation 属性的值为 0.您可以创建一个 3D 图表,虽然什么事情起来都没有改变.实际上您只是在看图表的正前方表面,就像一个标准的区域图表的视觉效果一样.

同时请注意,对一些数据进行 3D 视图的展示的效果可能是很令人满意的,但是同时其它的数据需要使用 2D 的面板.对于这些场景,调整附加在每一个 ChartGroup 上的 Use3D 属性.

8. 设计时制作 2D 图表工具

本章介绍三个可在设计时创建 2 D 图表的方法。您可以使用智能设计器创建图表,图表向导或图表属性设计器。阅读本章后您就可以自己挑选一个最简单的方法来制作图表。

8.1 使用智能设计器

智能设计器允许您在设计窗口快速设置图表属性。这就解决了以前只能在属性窗口中设置图表属性的问题。您可以使用智能设计器的特性在设计时来创建一个功能丰富的 2 D 图表。

智能设计器提供了以下功能:

- 内置的工具栏和编辑器

- 能够通过编辑标签编辑器添加/编辑标签
- 通过拖拽方式在图表区为系列添加标签
- 每个图表元素的标签对于每个图表元素来说提供了更好的用户交互

本章主要介绍智能设计器的功能，以及智能设计器中给每个元素提供标签的功能。

关于如何用智能设计器实现特殊的图表功能,请参阅用智能设计器创建和格式化图表元素

8.1.1 主浮动工具栏

对于 C1Chart 控件智能设计器有三个主浮动工具栏的。浮动工具栏的如下:

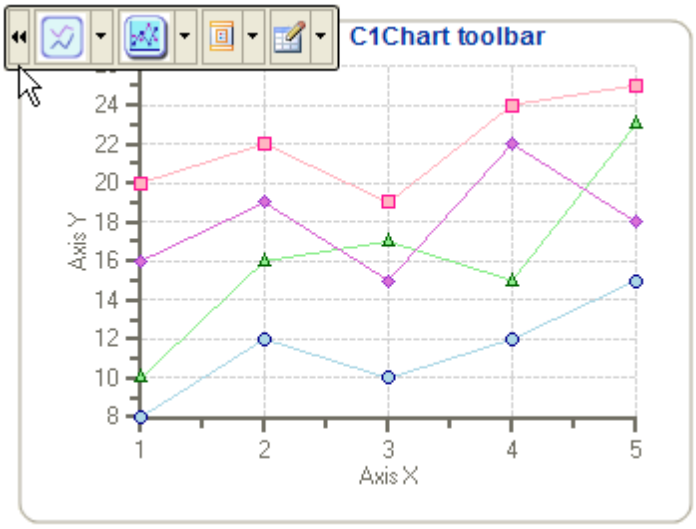
- C1Chart 工具栏
- ChartArea 工具栏
- PlotArea 工具栏

智能设计器的工具栏被称为浮动工具栏，它和典型的工具栏略有不同。智能设计器的工具栏只有工具正在使用中（Ａｃｔｉｖｅ）时出现,而且该控件不能停驻到其他控件。

本节描述 C1Chart 主工具栏按钮的功能。

8.1.1.1 C1Chart 工具栏

C1Chart 控件的主要的 C1Chart 浮动工具栏包括以下命令按钮:关闭（**Close**）、图表类型（**Chart type**）,图表子类型（**Chart sub-type**）、布局（**Layout**）和一个数据（**Data**）按钮。

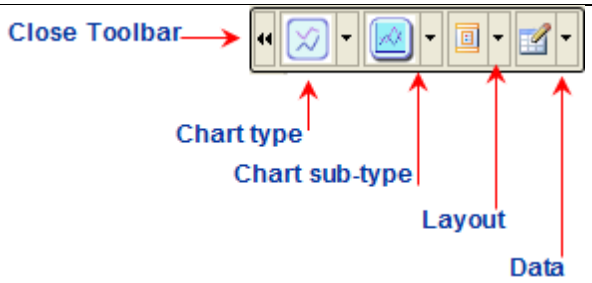


显示 C1Chart 浮动工具栏

显示 **C1Chart** 浮动工具栏只需滑动你的鼠标到 **C1Chart** 控件。

C1Chart 工具栏的命令按钮

下面的图形为 **C1Chart** 工具栏的每个命令按钮提供了一个标签。



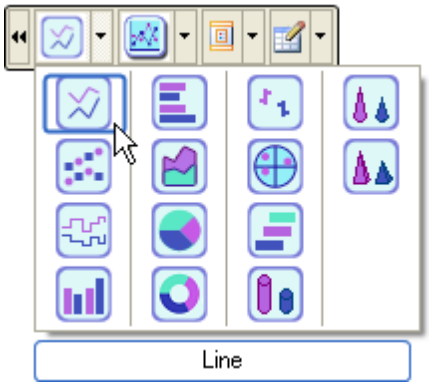
下面列出 C1Chart 工具栏所有的命令按钮并且逐一描述其功能。

关闭按钮

“关闭”按钮一旦点击关闭工具栏  。

图表类型按钮

图表类型命令有一个下拉菜单,其中包含 C1Chart 控件提供所有的图表类型选项。鼠标在每个图表图标上滑动标签就会显示出该图表类型名称。您可以选择下列的任何一种简单的图表类型:行,x - y、步骤、列、条、区域、饼图、圆环图、金融/股票，极坐标图/雷达图,甘特图、柱面、锥面图和角锥体图。



图表子类型按钮

图表子类型命令也有一个下拉菜单,其中包含 C1Chart 提供的所有的图表子类型的选项。鼠标在每个图表图标上滑动就会标签就会显示出选择的子类型图表名称。例如,当在图表类型中选择了线类型,那么如下的复杂的图表子类型就可供选择了:x - y 图线,x - y 图表/线/符号,x - y 图表/线/ 3 D, x - y 图表/线/叠加,x - y /符号/图表/行叠加,x - y /符号/图表/行叠加,x - y 图表/线/ / 3 D 叠加。有独特的子类型图表每个图表类型有独特的子类型图表。

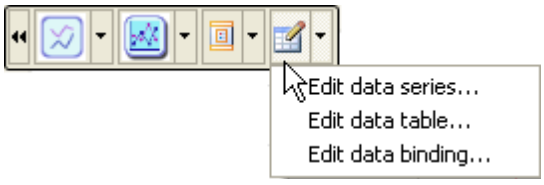
布局按钮

布局命令按钮的下拉菜单包含的页眉,页脚和图例。选择其中一个就会在图表中直接显示出可编辑的页眉,页脚或者图例。

选择页眉或者页脚就会出现一个可编辑的编辑框和格式化用的工具栏 .如果这个格式化用的工具栏没有立刻显示出来，您可以在编辑框中点击鼠标左键，这个工具栏就会出现了 . 下面的图例就是给 C1Chart 控件的图表区域自动添加页眉编辑框。

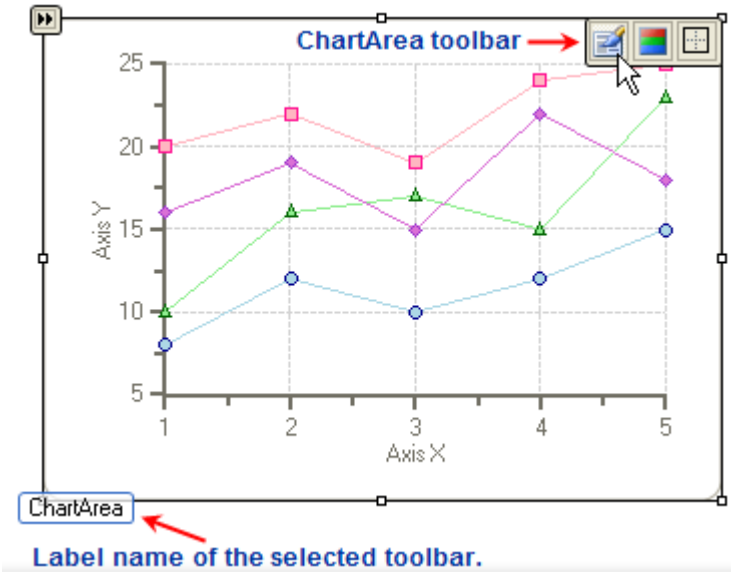
数据按钮

数据按钮有一个下拉是的菜单，菜单中有三个选项供用户选择：编辑数据系列，编辑数据表和编辑数据绑定。 选择编辑数据系列会出现数据系列用的图表属性设计器。选择编辑数据表会出现给数据表用的图表属性设计器。选择数据绑定就会出现数据绑定用的图表属性设计器。如果你不想编辑数据系列，数据表，数据绑定，你只需点击  按钮就可以切换到图表属性设计器的其他 3 个界面，这时你就可以编辑图表的其他元素了。



8.1.1.2 ChartArea 工具栏

另一个主要浮动工具栏，C1Chart 控件的 ChartArea 浮动工具栏包含：属性、背景和边框命令按钮。下面的图显示了用户选择 C1Chart 控件的 ChartArea ChartArea 工具栏就会出现。当用户选择一个工具栏,我们提供一个标签名，以方便用户使用。



显示 ChartArea 工具栏

滑动你的光标到 ChartArea 就会出现 ChartArea 工具栏

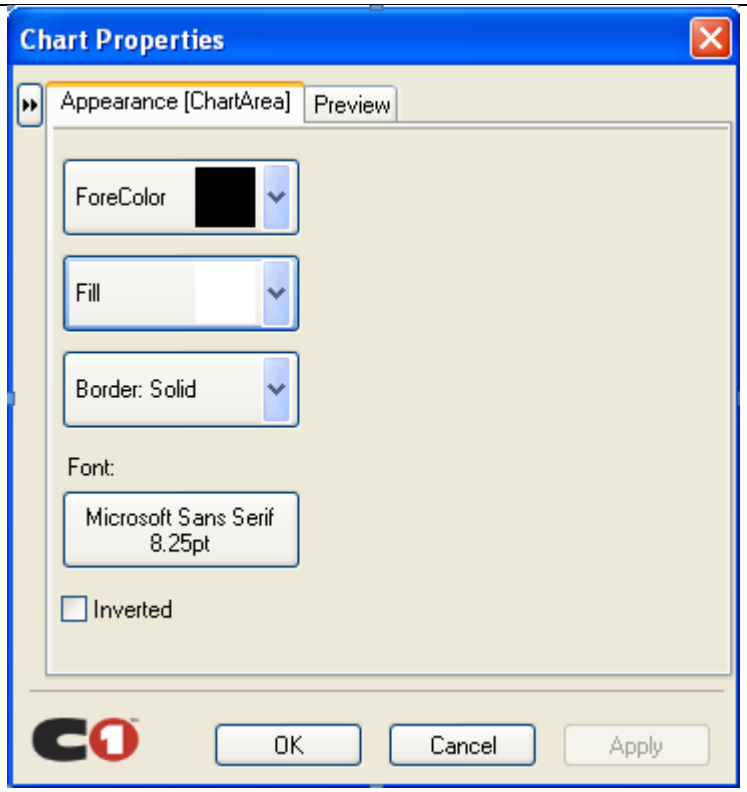
ChartArea 工具栏的命令按钮

下面的图展示了 ChartArea 工具栏的各个按钮。

下面的章节列出了 ChartArea 工具栏所有可用的命令按钮，并逐一描述其功能。

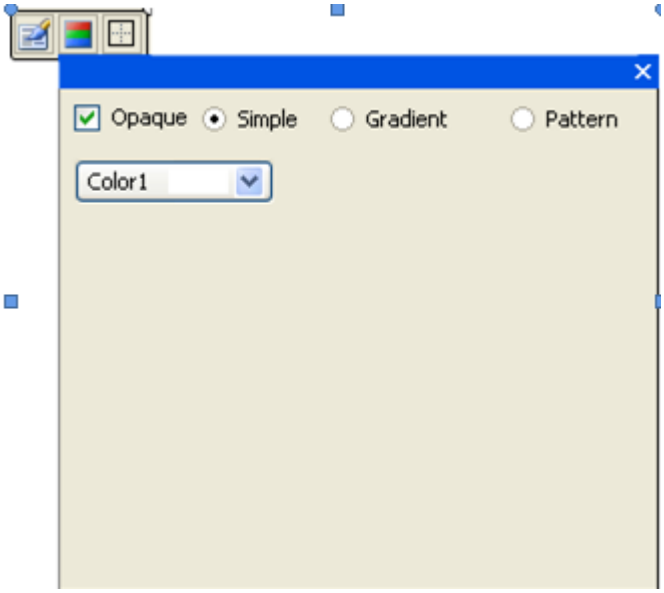
属性按钮

当用户点击属性按钮，ChartArea 的图表属性编辑器就会出现。



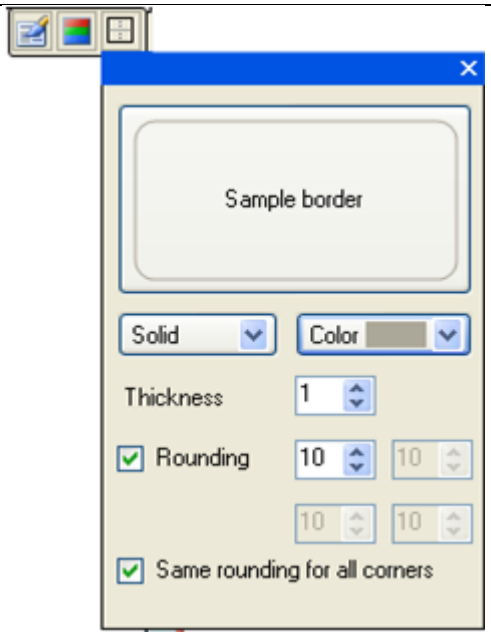
背景按钮

背景命令按钮点击出现下拉框,其中包含三个不同类型的背景样式和一个颜色下拉列表，用户可以选择不同的颜色作为 ChartArea 的背景色。



边框按钮

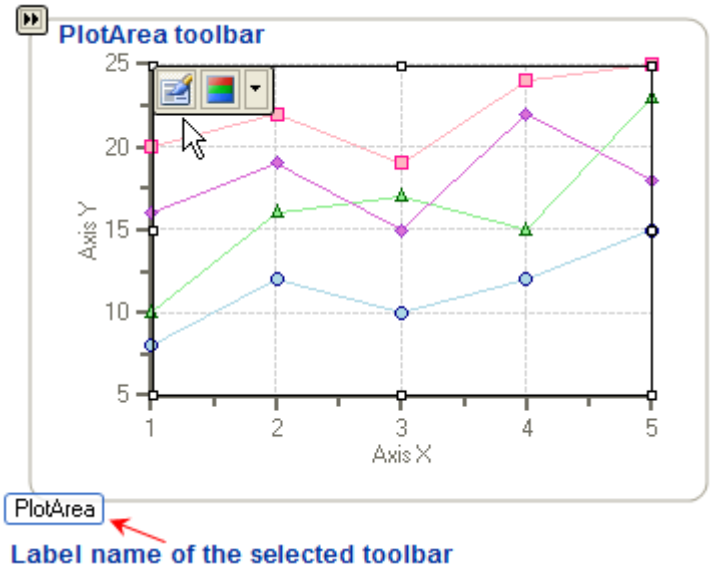
边框按钮点击出现一个下拉框,其中包含给 ChartArea 设定边框用的可编辑的边界样式和颜色的。



8.1.1.3 PlotArea 工具栏

另一个主浮动工具栏, C1Chart 控件的 PlotArea 工具栏包含一个属性项,背景项,和一个包含添加/编辑标签命令项的下拉菜单用以新增和编辑图表的标签,另外还包含一个倒转图表的命令项能让图表倒转。

下面的图显示了用户选择 C1Chart 控件的 PlotArea　PlotArea 工具栏就会出现。当用户选择一个工具栏,我们提供一个标签名，以方便用户使用。



显示 PlotArea 浮动工具栏

滑动你的光标到 PlotArea 就会出现 PlotArea 工具栏

PlotArea 浮动工具栏的命令按钮

下面的图展示了 PlotArea 工具栏的每个按钮

下面的章节列出了 PlotArea 工具栏所有可用的命令按钮，并逐一描述其功能。

属性按钮

PlotArea 工具栏的属性按钮的功能和 ChartArea 工具栏的功能完全一样

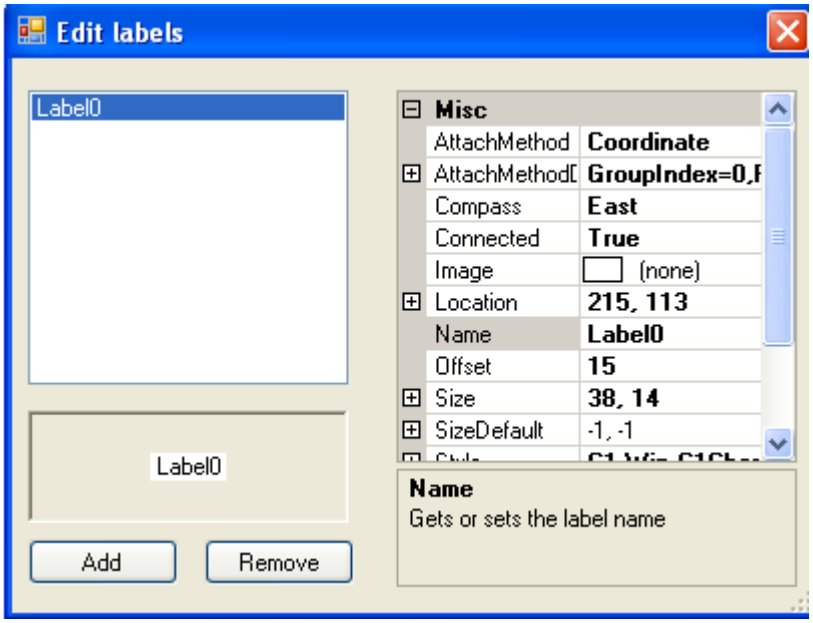
背景按钮

PlotArea 工具栏的背景按钮的功能和 ChartArea 工具栏的功能完全一样

下拉菜单

PlotArea 工具栏的下拉菜单包含了两个菜单项：反转和新增 / 编辑标签。反转命令反转 Y 和 x 轴坐标。关于反转图表的更多信息，请参阅反转和颠倒图表坐标（219 页）。

点击添加/编辑标签命令项会出现标签编辑器。在标签编辑器中可以新增或编辑现有标签。



更多的标签编辑器信息请参看,给图表添加标签（335 页）

8.1.2 二级浮动工具栏

智能设计师器为 C1Chart 附加功能提供了四个二级浮动工具栏，例如 Header、Footer、标签和标题。二级浮动工具栏包括如下工具栏:

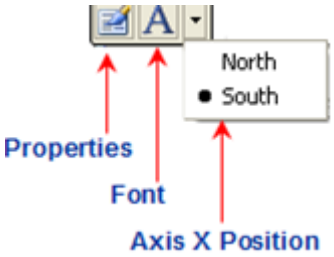
- 轴工具栏
- Header 工具栏
- Footer 工具栏
- 标签工具栏
- 图例工具栏

本节描述 C1Chart 的二级浮动工具栏的各个按钮的功能

8.1.2.1 轴坐标工具栏

浮动工具上的 X 轴和 Y 轴的元素包括以下命令按钮:属性、字体,和一个 C1Chart 控件的 X 轴和 Y 轴位置。下面的图展示了 X 或 Y 轴工具栏上的命令按钮。

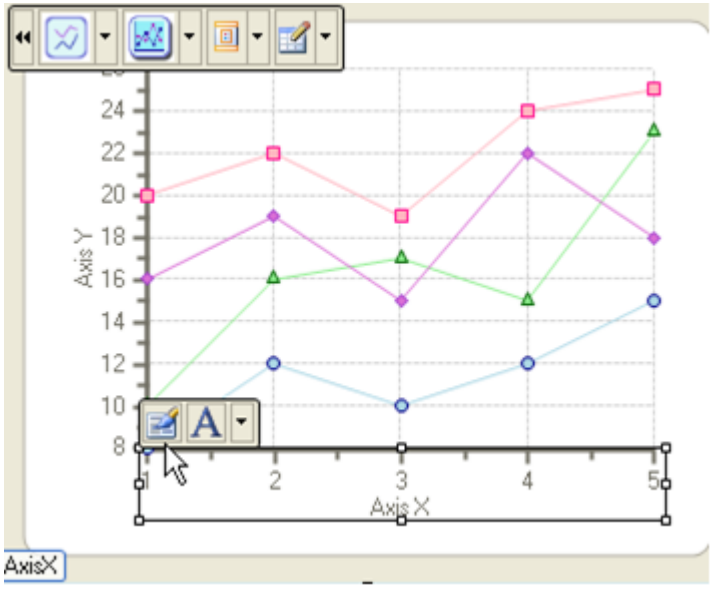
X 轴工具栏 Y 轴工具栏



显示 X 轴或 Y 轴浮动工具栏

显示 X 轴或 Y 轴浮动工具栏只需滑动你的鼠标 X 轴或 Y 轴区域

下列例子说明滑动鼠标停留在 X 轴的区域会出现 X 轴浮动工具栏。



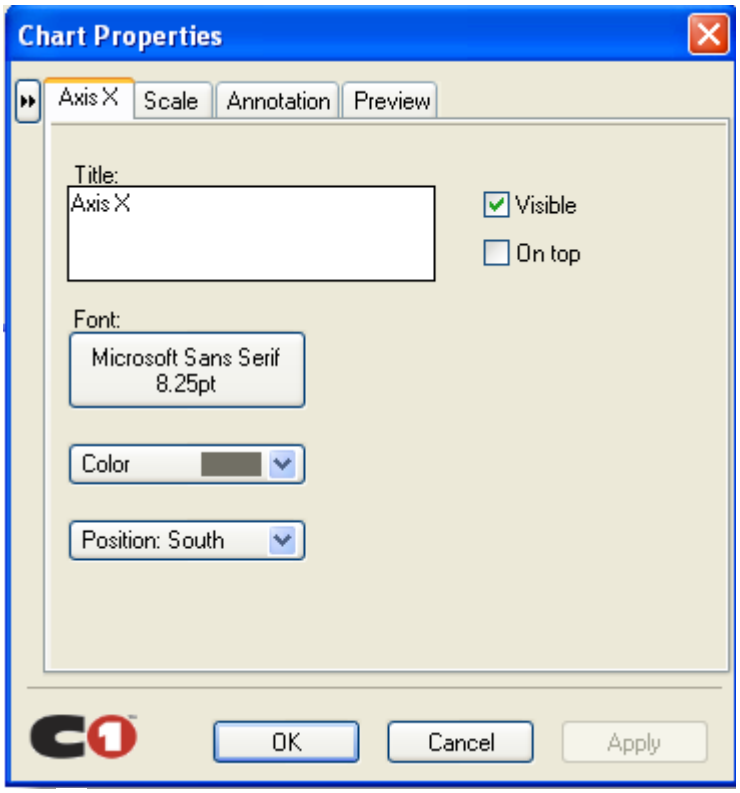
X 轴 / Y 轴工具栏的命令按钮

下面章节中列出了 X 轴 / Y 轴工具栏的各个命令按钮，并逐一描述其功能。

属性按钮

当用户单击 X 轴 / Y 轴工具栏的属性命令按钮 就会出现图表属性的对话框用于设定 X 轴 / Y 轴的元素。

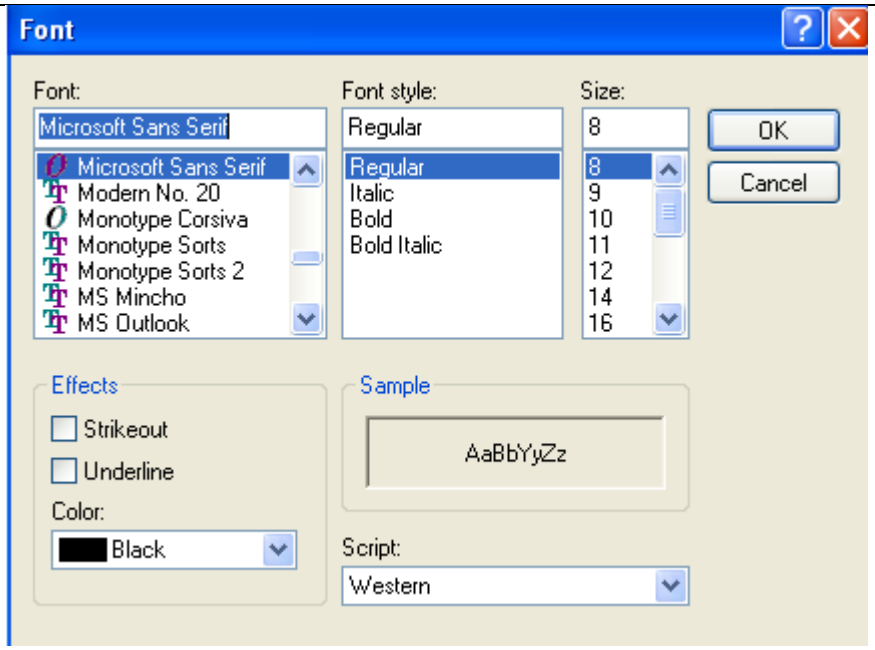
下图表示出了点击 X 轴工具栏上的属性命令按钮时出现的图表属性对话框。



点击  按钮在图表属性设计器的左窗口打开图表元素的导航树视图。

字体按钮

点击字体命令按钮以字体对话框的形式显示 X 轴和 Y 轴的字体。在这里,可以修改轴上的字体格式。

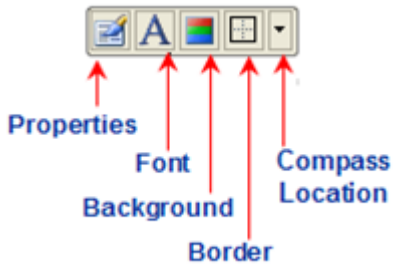


位置按钮

位置命令按钮有一个下拉列表,包含轴的位置列表(西,东、北、或南)供用户选择。Y 轴的位置定位 Y 轴：西的位置是指 Y 轴位于图表的左边，东位置是指 Y 轴位于图表的右边。X 轴的位置定位位 X 轴：南位置是指 X 轴在图表下边，北位置是指 X 轴在图表上边。默认的 Y 轴位置是西，X 轴位置是南。

8.1.2.2 Header 和 Footer 的工具栏

浮动工具栏上 Header 和 Footer 包括以下命令按钮:属性、字体、背景、边框和 C1Chart 控件的 Header 和 Footer 的位置。下图 Header 和 Footer 的工具栏的每个命令按钮提供了标注。



显示 Header 和 Footer 浮动工具栏

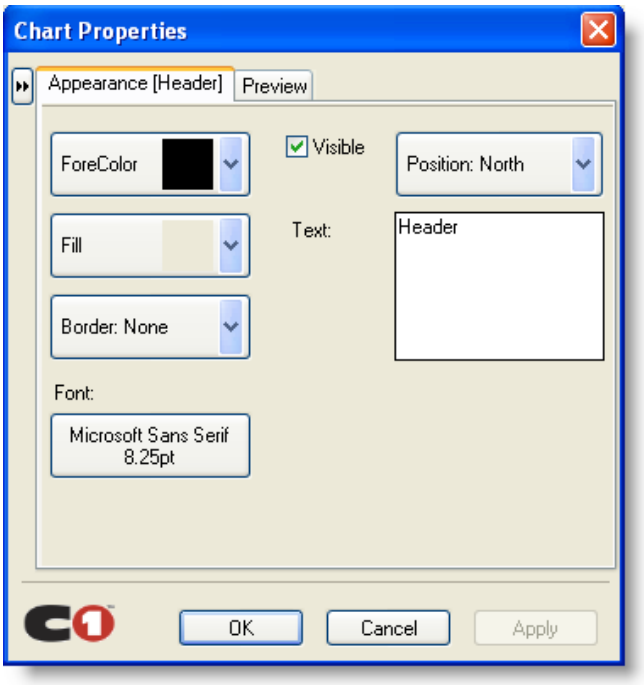
为了使 Header 和 Footer 浮动工具栏出现，你必须从 C1Chart 工具栏的下拉菜单中选择 Header 和 Footer。

Header 和 Footer 的浮动工具栏的命令按钮

下面的章节列出了 **Header** 和 **Footer** 的工具栏所有可用的命令按钮，并逐一描述其功能。

属性按钮

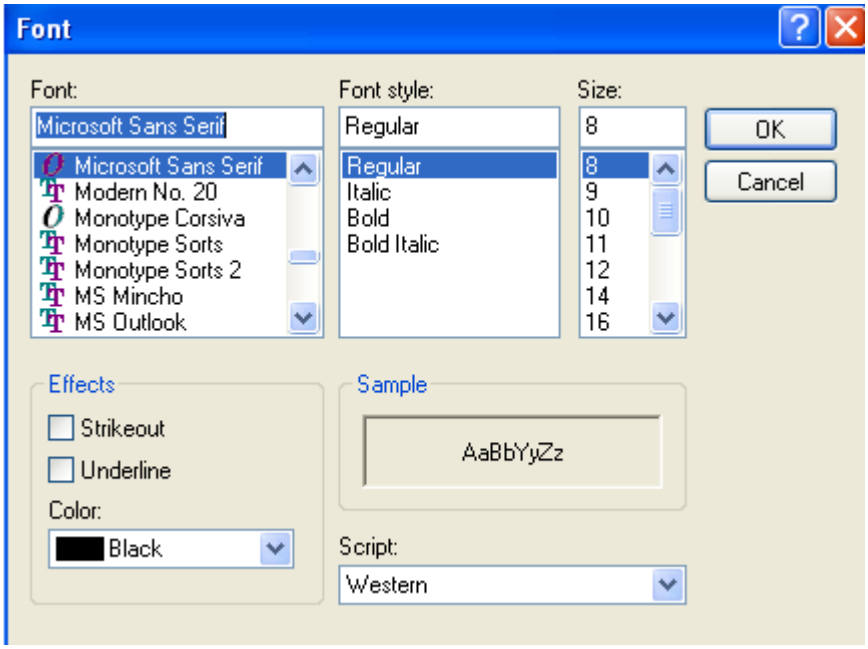
用户单击属性命令按钮会显示 Header 和 Footer 的的图表属性设计器



点击  按钮在图表属性设计器的左窗口打开图表元素的导航树视图。

字体按钮

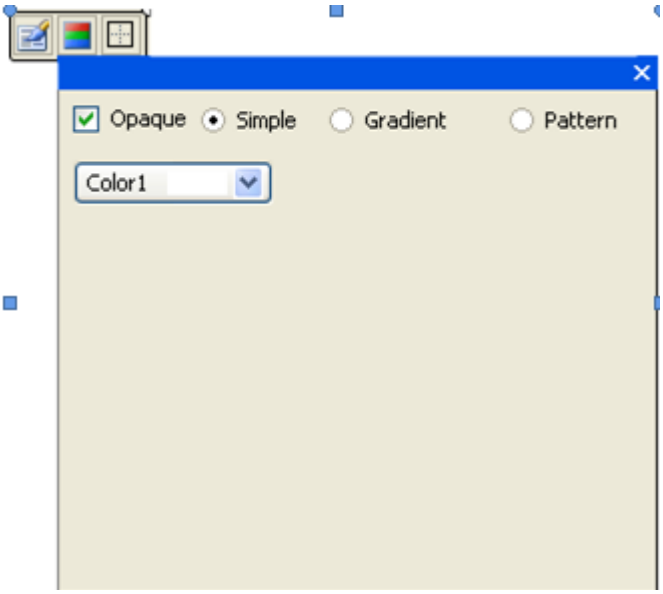
点击字体命令按钮以字体对话框的形式显示 Header 和 Footer 的字体。在这里,可以修改 Header 和 Footer 上的字体格式。



背景按钮

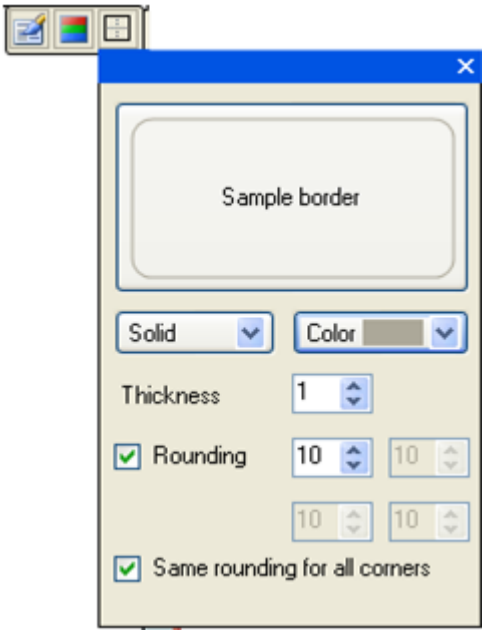
背景命令按钮点击出现下拉框,其中包含三个不同类型的背景样式和一个颜色下拉列表，用

户可以选择不同的颜色作为 Header 和 Footer 的背景色。



边框按钮

边框按钮点击出现一个下拉框,其中包含给 Header 和 Footer 设定边框用的边界样式和颜色。



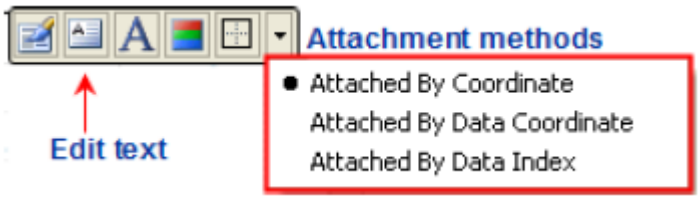
位置按钮

位置命令按钮有一个下拉列表,包含一个相反的方向列表(西,东、北、或南)供用户选择。这些方向会定位 Header 和 Footer 的位置：北是图表的上方，南是图表的下方，西是图表的左侧，东是图表的右侧。默认的位置：Header 是北，Footer 是南。

8.1.2.3 标签栏

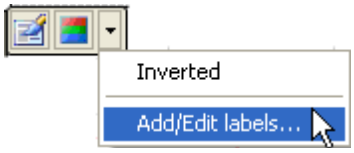
浮动工具栏上标签栏和 Header/Footer 工具栏几乎一样，除了它有一个编辑文本用的文本编辑命令按钮，并且下拉菜单里的选项也不一样。它的下拉菜单有：用坐标添加，用数据坐标添加，和数据索引添加的命令项。用坐标添加的功能可以在图表的任何位置加上标签，从图表的左上角

到标签之间的像素数可以被定义。用数据坐标添加的功能可以在 PlotArea 内的任何地方添加标签，该数据坐标可以被定义。用数据索引添加项可以在图表的特定数据点添加标签，坐标系和点系列以及 ChartGroup 都可以被指定。关于设计时用标签的添加方法的更详细的信息请参照添加和定位表标签（199 页）。关于用通过编程的方法使用添加标签的方法的详细信息见用像素坐标添加标签（200 页），用数据坐标添加坐标标签（201 页），用数据点添加图表标签，或者用数据点或者 Y 值添加图表标签（201 页）。下图显示了标签栏的命令按钮。



显示标签浮动工具栏

您必须从 PlotArea 工具栏选择 新增/编辑标签选项，并且用标签编辑器增加一个标签，然后标签工具栏就会出现了。



标签浮动工具栏的命令按钮

下面的章节列出了标签工具栏所有可用的命令按钮，并逐一描述其功能。

属性按钮

用户点击标签工具栏的属性按钮，标签编辑器就会出现。在标签编辑器中你可以新增或者编辑已经存在的标签。

文本编辑按钮

标签工具栏上的文本编辑按钮用于编辑标签上的文本。

背景按钮

背景按钮的作用和 C1Chart 控件工具栏上的按钮功能相同。

边框按钮

边框按钮的作用和 C1Chart 控件工具栏上的按钮功能相同。

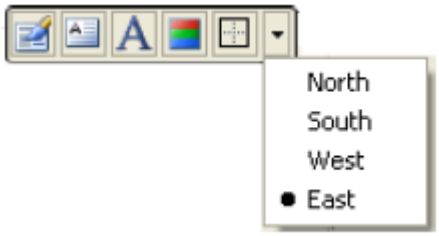
下拉菜单

标签工具栏上的下拉菜单有三个选项供用户选择：用坐标添加，用数据坐标添加,和数据索引添加。用坐标添加指定（X，Y）坐标添加标签， 用数据坐标添加定义数据坐标添加标签，用数据索引添加定义数据索引添加标签。

8.1.2.4 标签页的图例工具栏

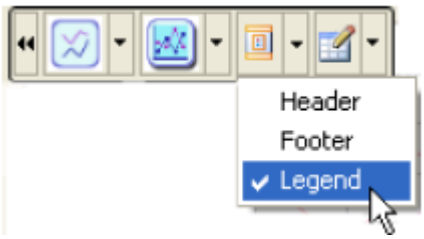
图例浮动工具栏和 Header/Footer 工具栏相同，但是有一个文本框编辑按钮。标签工具栏同样有这个命令按钮。

下图展示了图例工具栏



显示图例浮动工具栏

在图例工具栏的下拉菜单中选择标题项，图例浮动工具栏就会出现。



下面的章节列出了图例工具栏所有可用的命令按钮，并逐一描述其功能。

属性按钮

用户点击图例浮动工具栏的属性按钮，图表属性设计器就会出现。

文本编辑按钮

图例浮动工具栏上的文本编辑按钮用于编辑图例上的文本。

背景按钮

背景按钮的作用和 C1Chart 控件工具栏上的按钮功能相同。

边框按钮

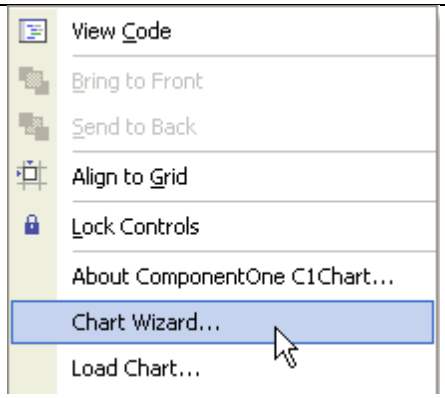
边框按钮的作用和 C1Chart 控件工具栏上的按钮功能相同。

下一章节介绍图表向导（Chart Wizard）的界面以及如何使用它在设计时建造 2D 的图表.

8.2 使用图表向导

使用图表向导只需要简单的 3 个步骤就能引导你做出一个图表。你可以选择不同的图表类型，给图表建立 header，footer，标题和 X，Y 轴，你还可以建立和编辑图表的数据。图表向导还有一个预览 tab，修改图表后可以在预览 tab 直接看到你的图表。

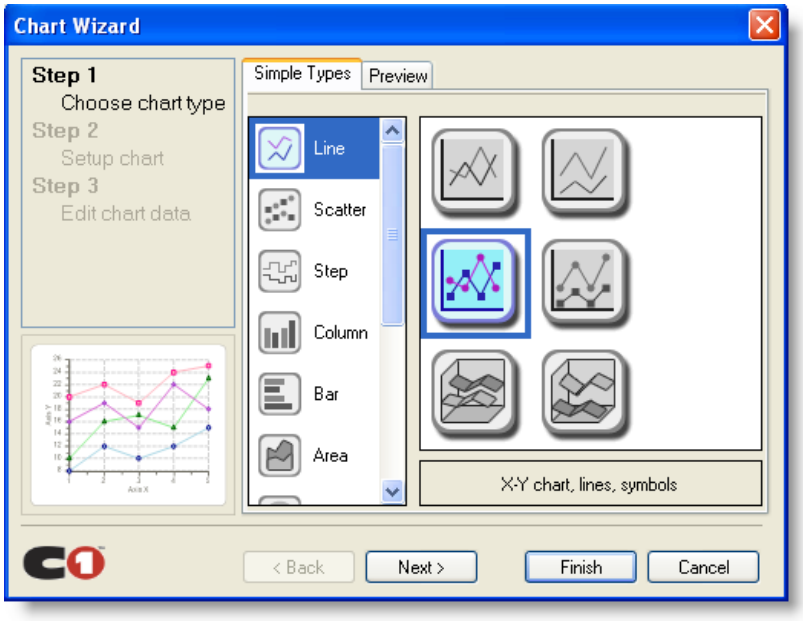
想要在设计时打开图表向导，鼠标右键点击 C1Chart 控件，在右键菜单中选择图表向导。



8.2.1 步骤 1. 选择图表类型

图表向导中提供了几种不同的图表类型 ,这样用户就可以在设计时快速的制作不同类型的图表。图表类型包括：线，散点图、柱状图、条状图、区域图、饼图、圆环图、股票图，极坐标图/雷达图,甘特图、柱面、锥面图和角锥体图。另外，每种图表类型还有独立的子类型，这样就可以进一步选择图表类型。关于每个图表类型的更详细信息，请参照定制 2D 图表（80 页）

简单的图表类型 tab

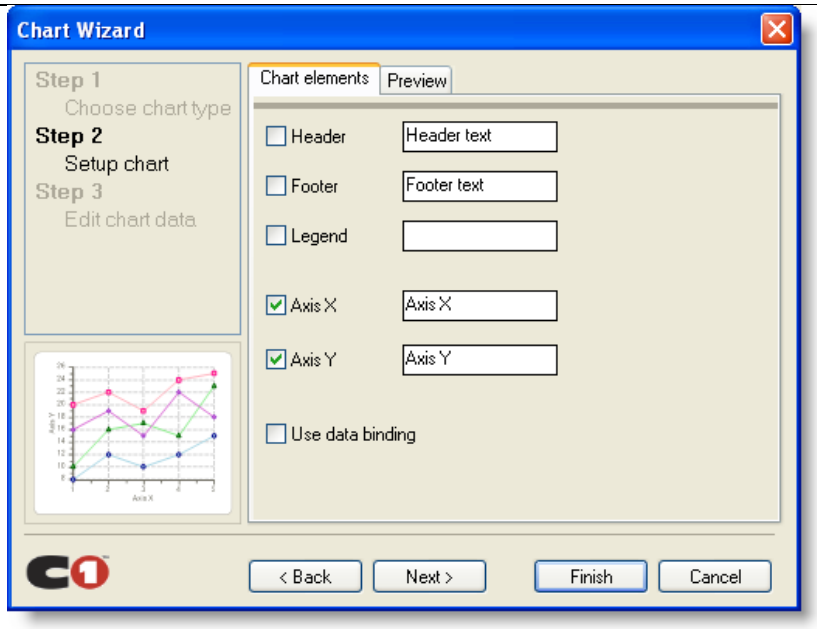


选择了图表类型，点击下一步来定义新图表的其他信息。

8.2.2 步骤 2：建造图表

图表向导的下一步是建造和修改图表的 Header，Footer，图例和 X 轴/Y 轴的题目。在 X 轴或 Y 轴旁边的文本框中输入就可以定义 X 轴或 Y 轴的题目。点击选择框（Checkbox）可以让对应的图表元素在图表上可用。关于这些元素的更详细信息，参加客户化图表元素（231 页）。如果你想把你的图表连接到一个已经存在的数据可以，你需要选择使用数据绑定（User data binding）选择框。

图表元素 tab



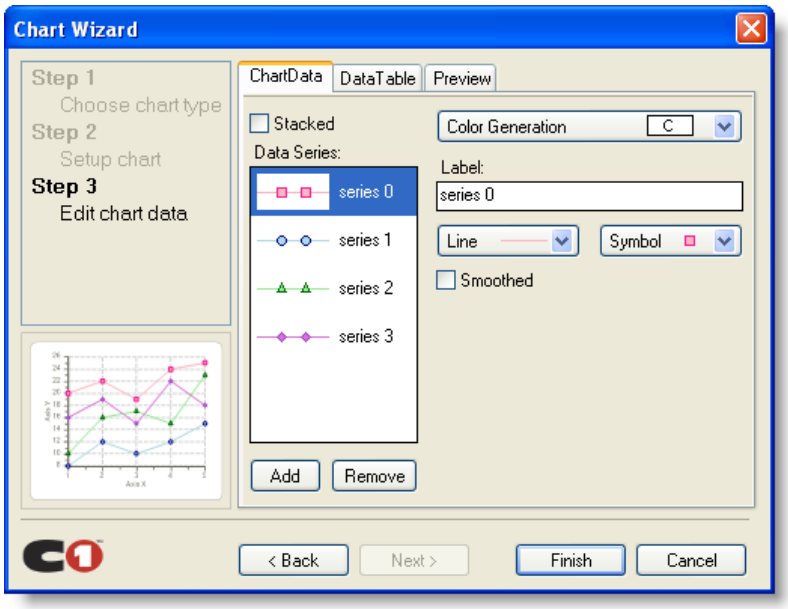
图表元素设定之后，点击下一步按钮进入编辑图表数据步骤。

8.2.3 步骤 3.编辑图表数据

图表向导的下一步是用来建造和修改图表数据区域的数据图表系外观的。这一步中包括图表数据（ChartData），数据表（ DataTable）和预览 tab。

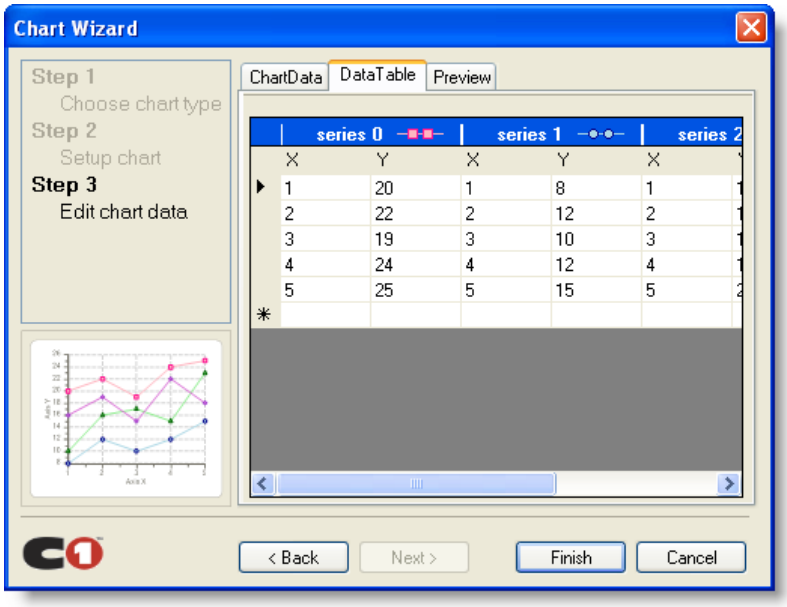
图表数据（ChartData）

在图表数据 tab，你可以增加和移除数据系，定义每个数据系的外观，例如标签，线类型和符号。如果希望某个数据系的线显示平滑，你可以选择 Smoothed 选择框。在数据系列表中选择一个系列，然后将 Smoothed 选择框的值设定为真（True），这样就可以将该数据系的线设定为平滑的了。



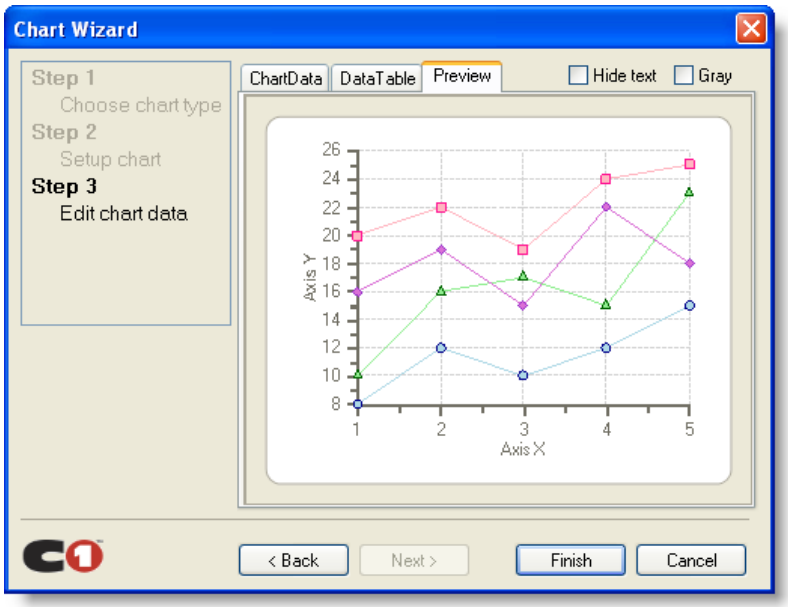
数据表（DataTable）tab

你可以选择数据表（DataTable）tab 来看和图表关联的说有数据。你想改变这些数据，也可以在这个 tab 完成。



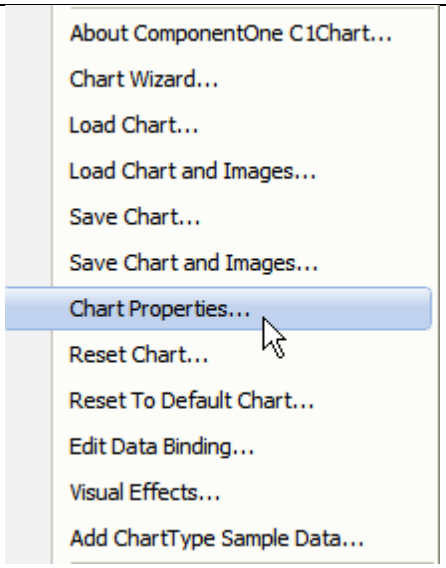
预览（Preview）tab

最后，选择预览 tab 就可以看到所做的图表。图表制作完成后点击结束（Finish）按钮。



8.3 使用图表属性设计器

图表属性设计器提供简单和交互式的方法来创建新的图表或者修改已经存在的图表。和图表向导相似，图表属性设计器在步骤 1 包含相同的功能： 选择图表类型，步骤 3 包含编辑图表数据。但是，图表属性设计器还包含 X 和 Y 轴其他的属性设定以及 header，footer，标题，图表数据和图表的点区域的外观设定。



在设计时右键点击 C1Chart 控件并从右键菜单选择图表属性右键菜单就能打开图表属性设计器。

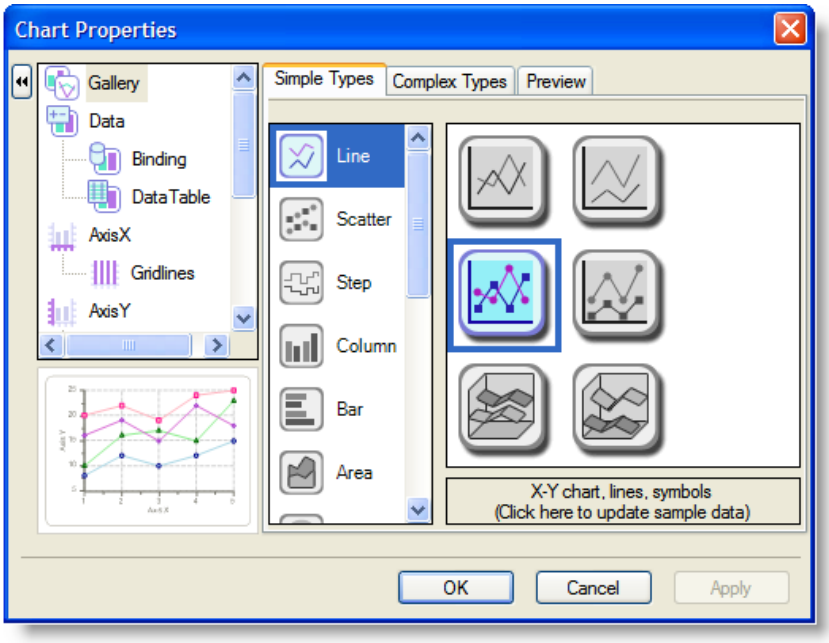
图表属性设计器提供更多的选项，能够帮助你开发设计图表时处理具体的细节。下面的章节展示了图表属性设计器的界面，并且逐一介绍每个功能。

8.3.1 图库

图表属性对话框的左面板上的图库项目给一个图表提供图表类型选项和子类型选项。关于所有图表类型的描述，请参见定制 2D 图表（80 页）。你可以从不同的简单图表类型中选择，也可以点击复杂图表类型为你的图表增加更多的功能。

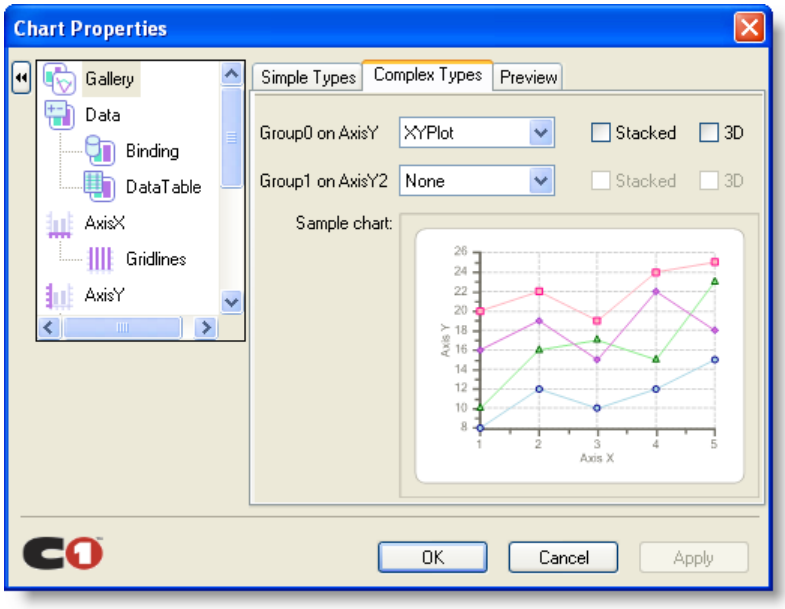
8.3.1.1 简单类型 Tab

在简单类型 tab 中，你可以选择一种简单图表类型，然后你可以在简单图表类型旁边的类表框中选择一种图表。



8.3.1.2 复杂图表类型 tab

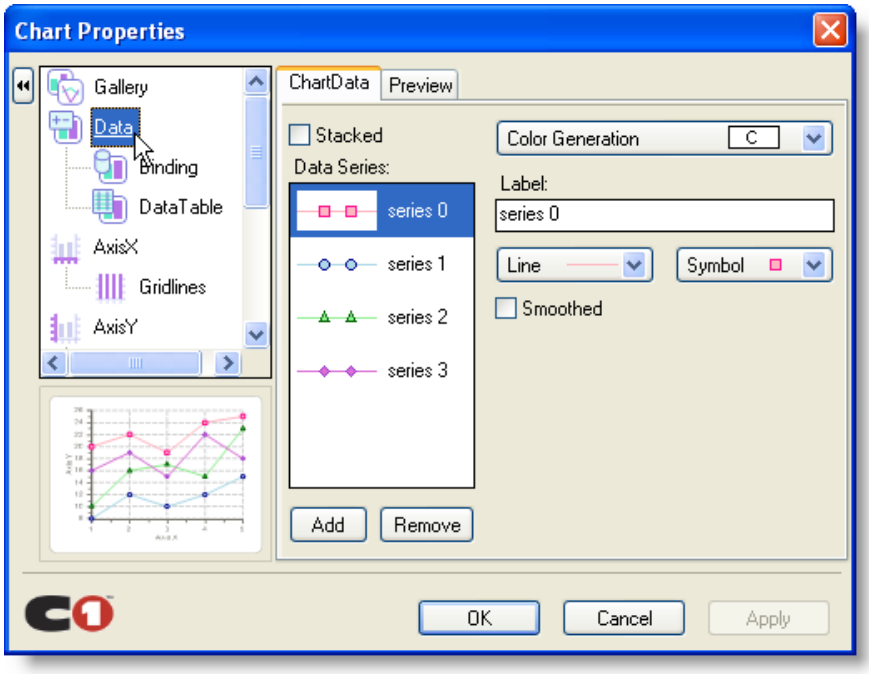
在复杂图表类型 tab 您可以指定您是否想要一个或两个图表组。同样,您可以给每个图表组在下拉框中选择图表的类型。对于每一组图表,你可以选择是否做成堆叠 并且/或者 3 D 的。下图显示了复杂类型 tab 在图表属性对话框中的图库部分的元素。



注意:如果你没有给数据(组 1)选择一个图表类型，数据(组 1)就不会出现在图表属性对话框左窗口的列表框中

8.3.2 数据元素

图表特性对话框左窗口的数据元素，提供了一个图表数据（ChartData）和预览 tab 用于创建或修改你的图表数据系并且预览您的当前图表。



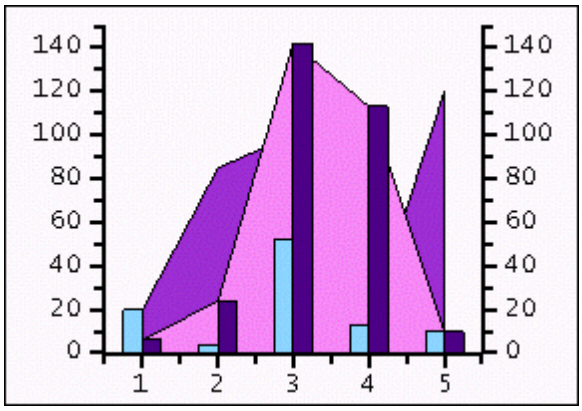
8.3.2.1 图表数据 tab

通过选图表数据（ChartData）tab，你可以在你的图表添加或删除数据系列,给每个数据系列指定标签和颜色,您可以通过单击堆栈复选框堆栈你的数据。要想选择一个数据源,点击绑定元素并选择一个数据源。想要修改表中的内容,单击数据表（DataTable）就可以对数据做出任何修改。想要了解 ChartDataSeries 对象的详细信息，请参照定义 ChartDataSeries 对象(159 页)或输入和修改图表数据(160 页)或 SeriesList 属性。

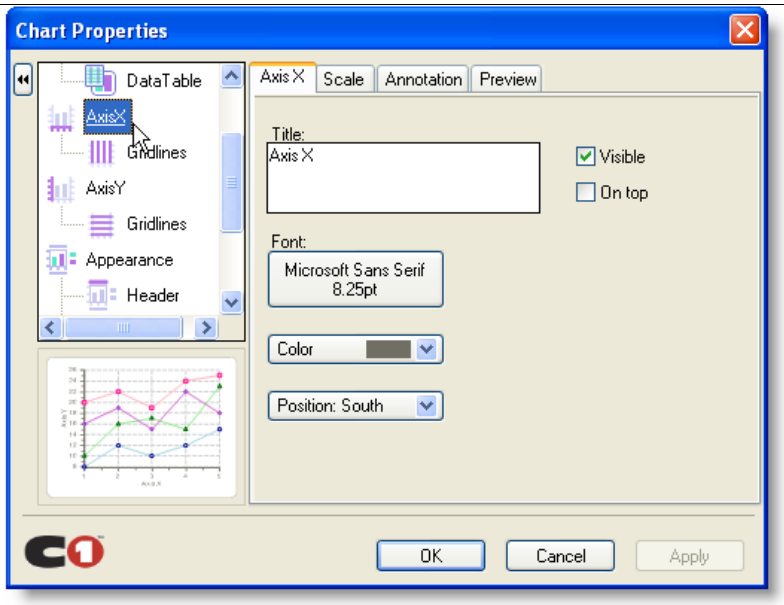
8.3.3 X 轴,Y 轴,和 Y2 轴元素

您可以选择轴 X,Y 轴,或轴 Y2，然后修改或创建轴的格式类型、范围以及轴的注释。每个轴元素(AxisX,AxisY,AxisY2)在图表对话框的右上方窗口中包含以下 tab:名称(AxisX,AxisY,或 AxisY2)、范围、注释和预览 tab。

只有第二个图表组(组 1)添加了数据，Y2 轴才会出现在图表属性设计器的左窗口。Y2 轴中的数据决定了轴的最大和最小值，这和图表组(0)的数据确定 Y 轴的最大和最小值的是一样的。可以通过手动设置来强迫两个轴对齐。下图显示了一个包含 Y2 轴的图表。

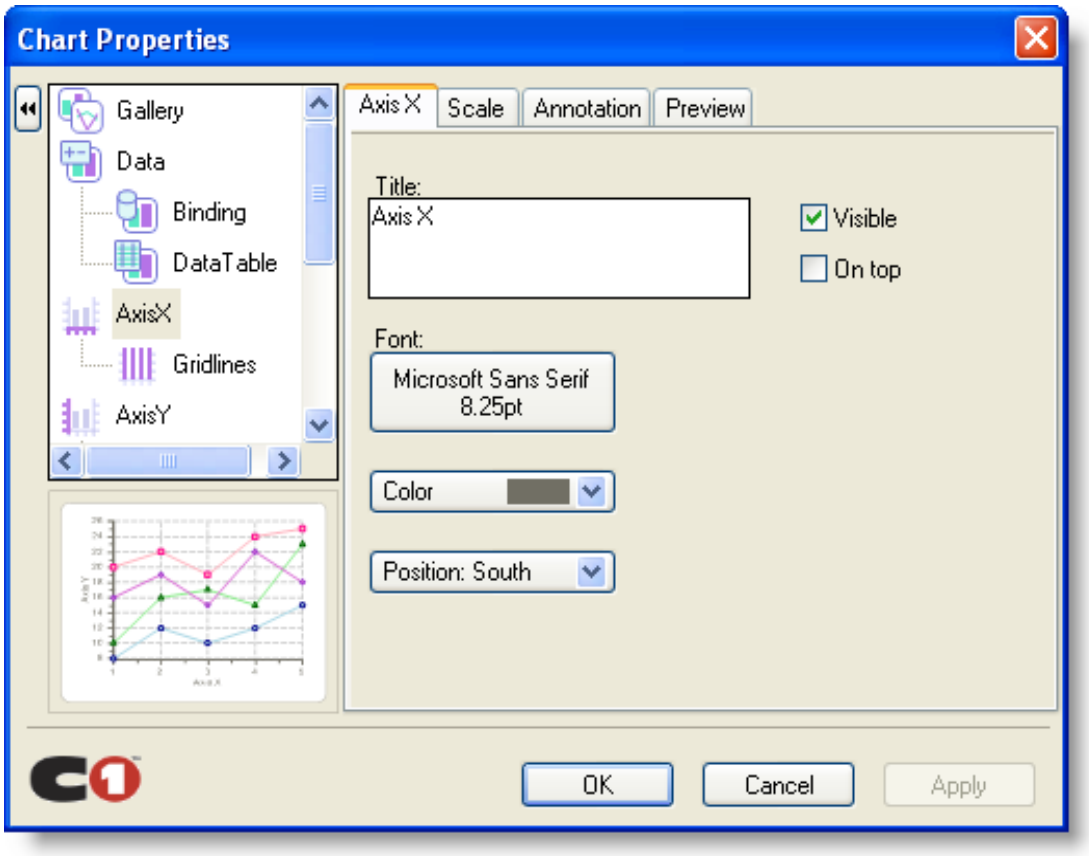


为了使 Y2 轴元素出现在图表属性对话框的左窗口，你必须在图库元素的复杂图表类型 tab 中给组 1 选择一个图表类型。并且,给组 1 选择完图表类型后 ,你需要给组 1 添加一系列的数据。想了解更多信息,请参考复杂类型 tab(第 141 页)。



8.3.3.1 X 轴，Y 轴或 Y2 轴 tab

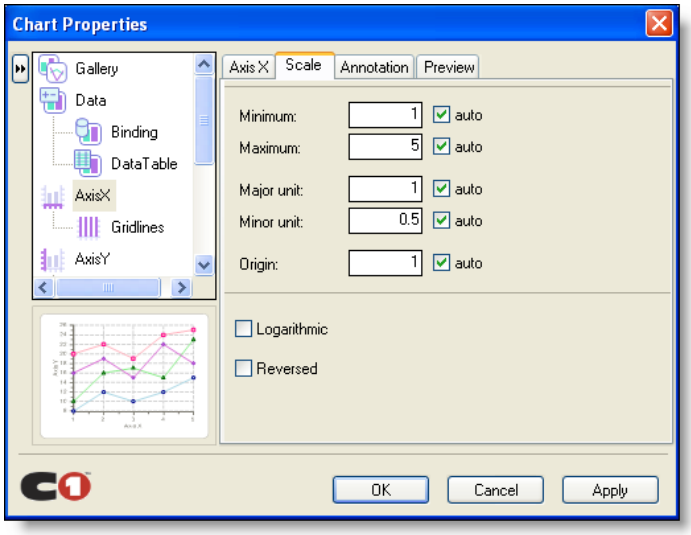
下面的屏幕截图是选择 x 轴时候的例子，在轴 tab 中,你可以指定轴的标题、字体、颜色和位置。此外,你还可以通过选择 On Top 复选框，让选择轴的标题出现在顶部。



8.3.3.2 范围 tab

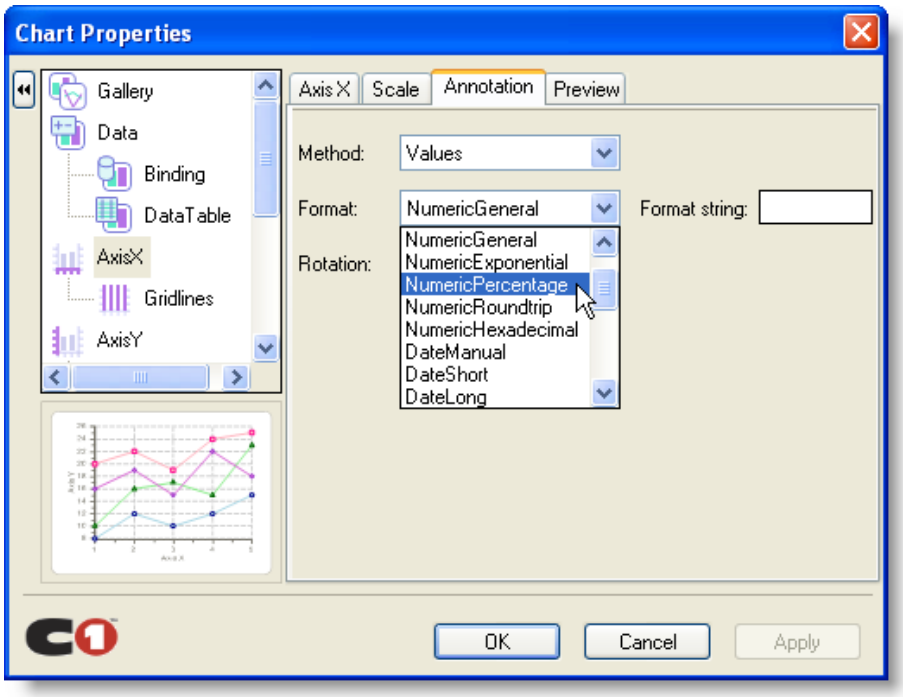
范围 tab 允许您设定下列属性 :最大值和最小值 ,主要单位 ,副单位和起始。下面的例子中，

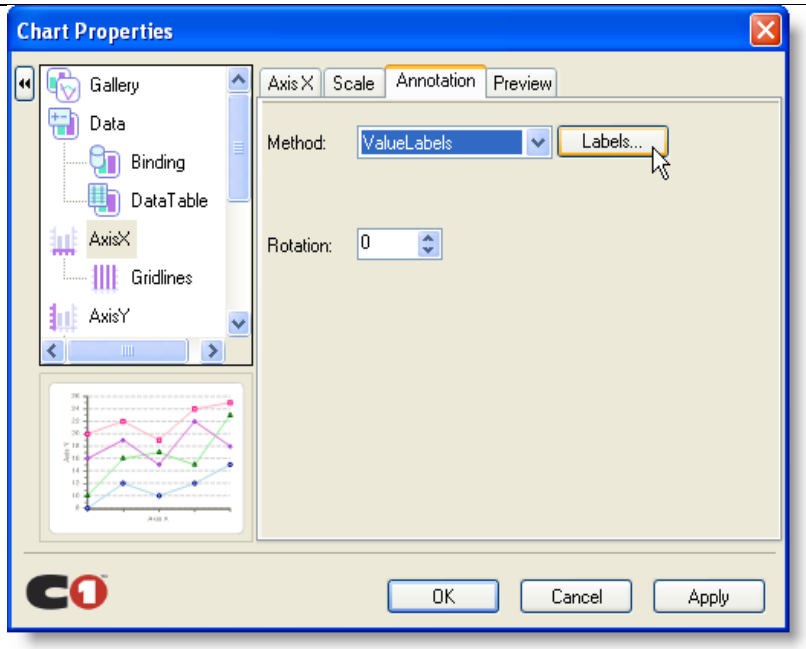
自动按钮被选择了，所以每个属性的所有值都被设定成了默认值。选择反转（Reversed）和对数（Logarithmic）选择框，你的轴还能被反转并且/或者设定为对数坐标轴



8.3.3.3 注释 tab

注释 tab 允许你定义是否给 X 轴、Y 轴或者 Y2 轴设置值或者 ValueLabels 的显示方式。如果你在方法(Method)下拉框中选择了值(Values) ,那么一个格式(Format)下拉框就会显示出来。在下拉框中你能够选择不同类型的格式，并且在格式串(Format string)编辑框中你能够定义格式串。更多的信息请参阅给注释设值（221 页）。格式类型的更多信息请参阅 FormatEnum 枚举。

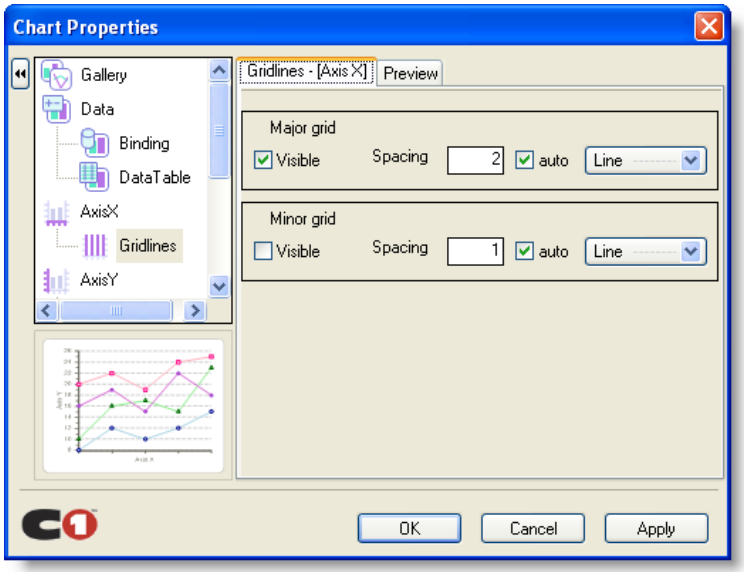




如果你在方法（Method）下拉框中选择了 ValueLabels,就会出现一个 Labels 按钮，点击 Labels 按钮，ValueLabels 集合编辑器对话框就会显示出来。在对话框中，你可以增加标签。关于 ValueLabels 集合编辑器更详细信息请参阅 ValueLabels 集合编辑器（56 页）。

8.3.3.4 格线 tab

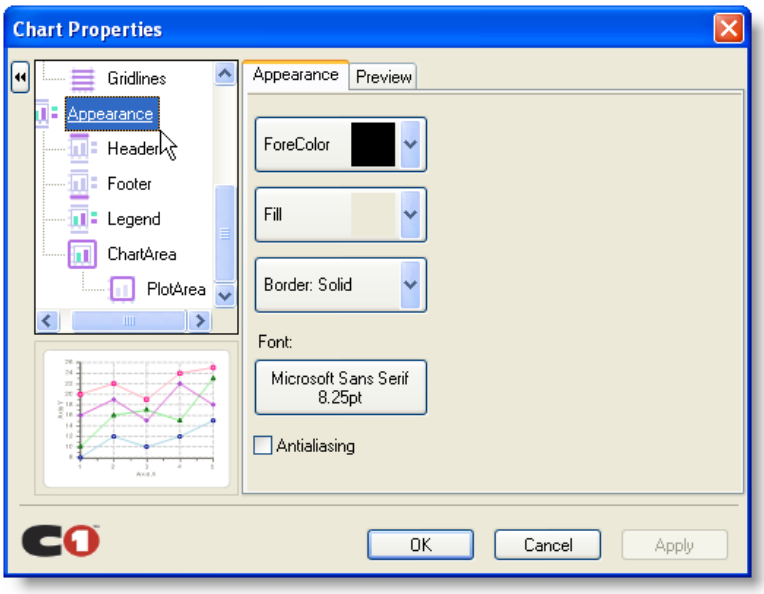
在格线 tab 中，你可以给每个轴添加一个主的或者副的格线，也可以两个都添加，这样就提高图表的可读性。如果你想看到这些格线，只需简单的把主格线（Major Grid）和副格线（Minor Grid）区域的可见（Visible）选择框选择上就可以了。通过在间隔（Spacing）编辑框中输入数字，你可以指定每个主格线或者副格线之间的间距。



外观元素

8.3.4 外观元素

图表属性设计器中，左窗口上的外观元素提供了外观 tab，外观 tab 中的属性，可以用来修改图表的前景色、填充色、边框样式、字体以及是否启用消除锯齿功能。图表元素的子元素，可以用来修改图表的 header，footer，标题，ChartArea 或 PlotArea 的外观。



8.3.4.1 表头外观 tab

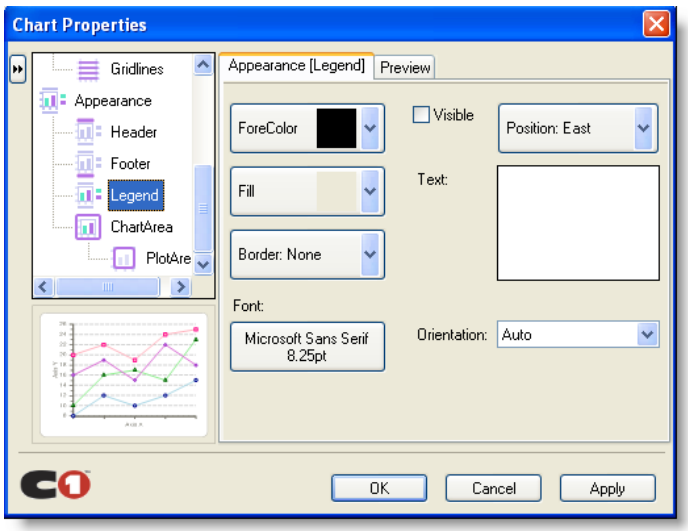
表头 tab 包含和表尾 tab 相同的属性：前景色，填充色，边框，字体，位置，可见和文本。

8.3.4.2 表尾外观 tab

表尾 tab 包含和表头 tab 相同的属性：前景色，填充色，边框，字体，位置，可见和文本。

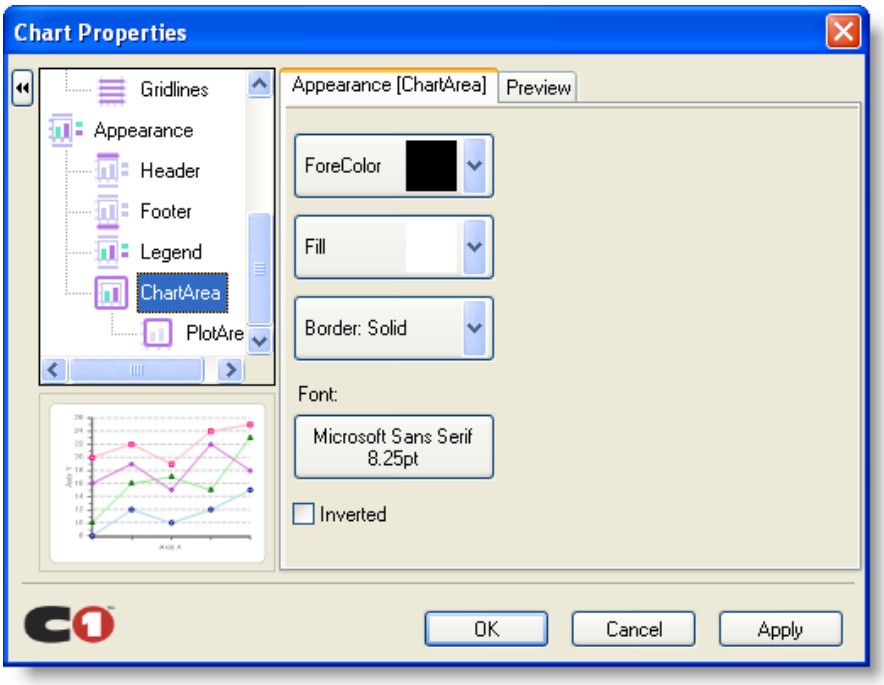
8.3.4.3 图例外观 tab

图例 tab 和表头 tab /表尾 tab 有相同的属性，但是多了一个方向属性。



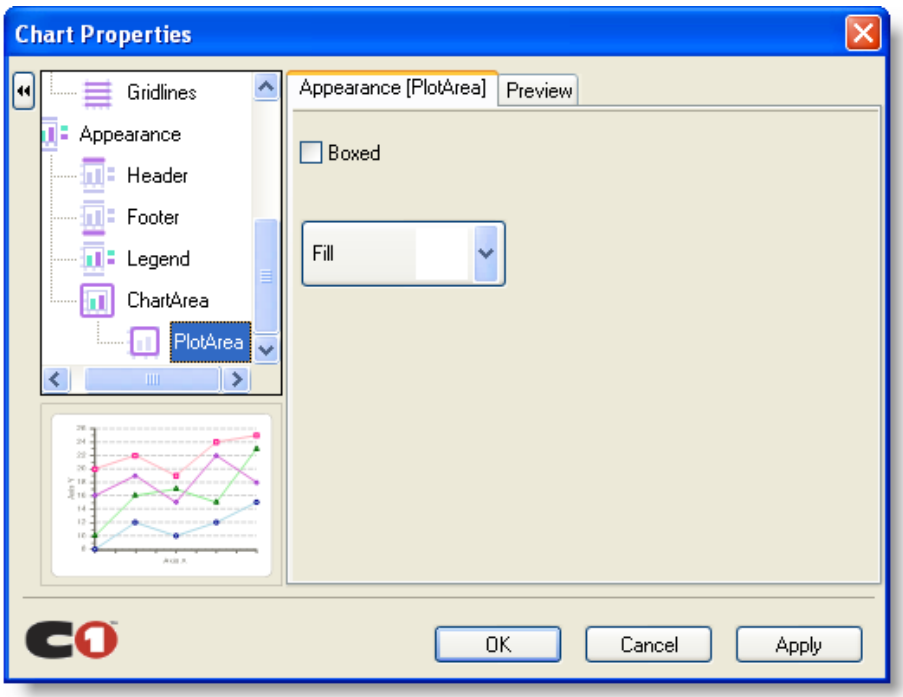
8.3.4.4 ChartArea 外观 tab

ChartArea tab 包含前景色，填充色，边框和字体属性，多了一个倒置属性，用于倒置图表的轴。



8.3.4.5 PlotArea 外观 tab

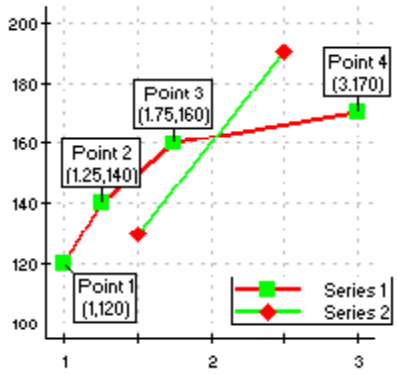
PlotArea 元素包含一个 Fill 和 Boxed 属性。点击 Boxed 选择框会把图表的图形区框起来。



9. 绘制数据

2D 图表显示一般布局提供的数据。数据被装载在 ChartDataArray 对象中，这些对象都有一个类型（type）对象。数据可以用提前装载的数组，也可以在设计时填充数据。

每个图表类型都有一种通用的布局。布局包括一个 X 数组，一个 Y 数组，和 Y1，Y2，Y3，Y4 数组。所有这些数组都可以装载数据，也可以为空。 例如画出一个 XY 图表只需要一个数组系列，只要 X 和 Y 数组有值就可以了，其他的 Y 数组可以为空。另外，不像一组数据的图表，每个数据系列可以有不同数量的点，并且不需要一定和 X 值匹配，这样就能做出下面的图。



General Data

2 Series, Maximum of 4 Points

4	Series 1 Num. Points			
1	1.25	1.75	3	Series 1 X-value
120	140	160	170	Series 1 Y-value
2	Series 2 Num. Points			
1.5	2.5			Series 2 X-value
130	190			Series 2 Y-value
Point 1	Point 2	Point 3	Point 4	

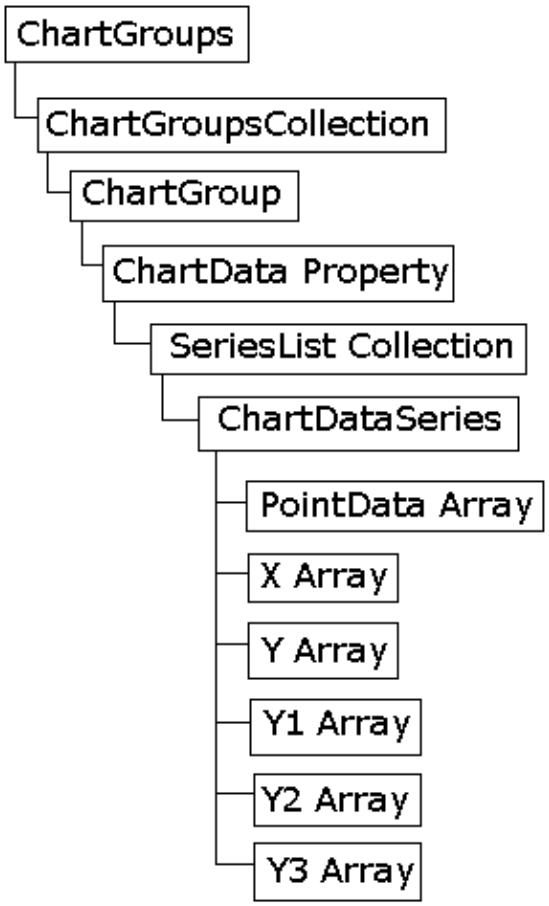
为方便起见,一个 PointData 属性(PointF 的一个数组)也可以被用来提供 X 和 Y 的数据。PointF 的值并不是独立的 X 和 Y 的数组,仅仅是另一种形式的数据输入和输出。PointData 属性值的变化会更改 X 和 Y 的数组,反之亦然。

在作图之前，一个重要的总体设计必须有下面的特征

- 每个系列的每个点都有自己的 X 和 Y 值。
- 每个系列可以包含不同数量的点。

9.1 定义图表数据（ChartData）对象

C1Chart 有一个特殊的层，该层包含数据关联的类，集合和属性。 本章节对每个数据对象的细节和如何访问的信息详细地逐一介绍。



9.1.1 定义 ChartGroup 对象

图表中的数据需要被组织到 ChartGroup 对象。每个图表包含两个 ChartGroup（尽管大部分图表只是使用第一个 ChartGroup），其中每个都被视为一个单独的实体。ChartGroup 允许超过一个图表显示在 ChartArea 并且提供多种对图表特有属性的访问。

在 C1Chart 中，一个 ChartGroup 对象代表一个图表组。ChartGroup 对象被组织到 ChartGroupsCollection 中，ChartGroupsCollection 可以通过 ChartGroups 对象来访问。这个集合提供两种方法访问 ChartGroups.

首先单独的图表组可以通过集合来访问。例如，下面的代码列出了如何通过 ChartGroupsCollection 来访问单独的图表组。

- Visual Basic

```
C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartType = Chart2DTypeEnum.XYPlot
```

- C#

```
c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartType = Chart2DTypeEnum.XYPlot;
```

其次 ,ChartGroup 对象的 Group0 和 Group1 属性返回相关的 ChartGroup,并且允许用下列的代码规避长集合名。

- Visual Basic

```
C1Chart1.ChartGroups.Group1.ChartType = Chart2DTypeEnum.XYPlot
```

- C#

```
c1Chart1.ChartGroups.Group1.ChartType = Chart2DTypeEnum.XYPlot;
```

如下面的代码所示，ChartGroupsCollection 允许通常的迭代方法：

- Visual Basic

```
Dim cg As ChartGroup For Each cg In C1Chart1.ChartGroups.ChartGroupsCollection
cg.ChartType = Chart2DTypeEnum.XYPlot
Next
```

- C#

```
ChartGroup cg;
foreach ( cg In c1Chart1.ChartGroups.ChartGroupsCollection )
cg.ChartType = Chart2DTypeEnum.XYPlot;
```

9.1.2 定义 ChartData 对象

ChartGroup 对象还包含 ChartData 对象。ChartData 对象包含 Hole 属性, FunctionsList 属性，能返回 ChartDataSeries 对象的 SeriesList 属性，PointStylesList 属性和 TrendsList 集合属性。这个 ChartGroup 对象基本上能够访问所有数据相关的对象和图表的所有属性：

- Visual Basic

```
C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData
```

- C#

```
c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData;
```

FunctionsList 集合包含了绘制图表数据和图表区域的函数。FunctionsList 属性能够得到现在的 ChartData 对象关联的 FunctionsCollection 对象。

PointStyles 属性提供了一个机制可以用不同的视觉效果标注特殊的数据点，和同一个数据系列的其他点区分开来。PointStyles 包含在 PointStylesCollection 中。

SeriesList 属性返回一个 ChartDataSeriesCollection，ChartDataSeriesCollection 是一个 ChartDataSeries 对象的集合。ChartDataSeries 对象包含图表的所有系列和数据。Series 对象包含 ChartDataArray，后者持有图表数据。

TrendLine 对象将趋势线分为两组，包括回归和非回归。回归趋势线表示的是包括多项式，指数，对数和傅立叶函数这类关于数据的函数趋势.

ChartData 的对象,还可以访问 Hole 属性，数据的 Hole 可以打破数据集的连续性。

9.1.3 定义 ChartDataSeries 对象

C1Chart 最重要的对象之一就是 ChartDataSeries 对象。这个数据系列包含图表中的所有数据和许多重要的数据相关的属性。

ChartDataSeries 包含在 ChartData 类型的 SeriesList 对象集合中，ChartData 对象是 ChartGroup 的一个对象。

- Visual Basic

```
C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData.SeriesList(0)
```

- C#

```
c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData.SeriesList[0];
```

这个系列对象对数据和数据访问来说非常重要，原因有两个：

1. 首先，ChartData 对象的 SeriesList 集合中系列的数量，决定了在能够显示在图表中数据系列的数量。如果一个 x-y 的图表希望有五套图，那么在 SeriesList 就应该有五个系列。
2. 其次，系列中 ChartDataArray 对象的 x、y、y1、y2、y3 定义了多少个值就决定了那个系列有多少个点，和如何绘制到图表中。如果图表 0 系列的 x 和 y 数组对象都有 10 个值，1 系列 x 和 y 数组都有 5 个值，那么第一个系列就会有 10 个点，第二个系列就会有 5 个点。

需要注意的是一个系列点的数量是数据组对象中长的那个元素的数量。例如，一个 XY 图表使用了 X 和 Y 数组。如果 X 有 5 个元素，Y 有 3 个元素，那么点的数量是 5，Y 值剩下的 2 个元素被假定为数据漏洞。即使 Y1 数组有十个元素，点的数量还是 5，因为 XY 图表不使用 Y1 数组。但是，如果图表类型变为 HiLo，那么 XY 和 Y1 都要被使用，所以点的数量是 10.X 和 Y 数组不存在的元素都被假定为数据漏洞。

9.1.4 定义 ChartDataArray 对象

每个系列包含六个 ChartDataArray 对象。X、Y、Y1、Y2、Y3 数组对象持有数据组和 X 轴，Y 轴 ,Y1 轴 ,Y2 轴 ,Y3 轴对应。第六个数据组对象是 PointData 对象。这个对象的数据是 PointF 数组或者 Point 值。这个对象用于作为 X，Y,Y1，Y2，Y3 数据组对象的替代选择。

每个 ChartDataArray 都是一个只读对象，只能通过两种方法修改。CopyDataIn 和 CopyDataOut 的方法拷贝数据到这些对象，并且从这些对象中返回数据组。

X,Y,Y1，Y2，Y3 数据组对象接收数据类型，这样不同的.NET 数据组类型可以被输入到 ChartDataArray 对象中（除了 PointData）。

9.2 输入和修改图表数据

在 ChartDataSeries 对象中 ,六个 ChartDataArray 对象包含有绘制图表用的数据组。X,Y,Y1，Y2，Y3 数据组对象持有并且返回 X 和 Y 轴用的数据组，同时 PointData 数据组对象持有并返回 X 轴和 Y 轴用的 PointF 和 Point 对象。

9.2.1 加载和提取图表数据

CopyDataIn 方法

在 C1Chart 中，X，Y,Y1，Y2,Y3 数据组对象接收对象数据类型。因此，不同的.NET 类型数据组可能被输入到 ChartDataArray 对象中(除了 PointData)。ChartDataArray 的 CopyDataIn 方法接收一个对象数据类型（可以是一组不同的类型）并把它装载到数据组中。

一个好的实现应该在图表绘制之前维护数据系列的一套数组，用数组的值更新 C1Chart 数据组对象。这样就能够完全控制数据，并可以批处理数据值。以下代码给出了一个简单的例子，说明数据组如何被构建并设定到图表的数据组对象中去的：

- Visual Basic

```
Dim xp(9) As Single
Dim yp(9) As Single
Dim i As Integer
For i = 0 To 9
xp(i) = i
yp(i) = i * i
Next i
With C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData.SeriesList(0)
.X.CopyDataIn(xp)
.Y.CopyDataIn(yp)
End With
```

- C#

```
float xp(10);
float yp(10);
int i;
for( i=0; i < 10; i++)
{
xp[i] = i;
yp[i] = i * i;
}
ChartDataSeries cds =
c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData.SeriesList[0];
cds.X.CopyDataIn(xp);
cds.Y.CopyDataIn(yp);
```

CopyDataOut 方法

CopyDataIn 方法给 ChartDataArray 对象装载数据组，而 CopyDataOut 方法从数据组对象中提取数据。这个方法返回一个对象数据类型，包含数据组。

- Visual Basic

```
Dim xpo As Object
Dim cds As ChartDataSeries
cds = C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData.SeriesList(0)
xpo = cds.X.CopyDataOut()
```

- C#

```
object xpo;
ChartDataSeries cds;
cds = c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData.SeriesList[0];
xpo = cds.X.CopyDataOut();
```

为了保证这个对象转换的数据类型正确，CopyDataOut 方法也能够接收 System.Type 参数，该参数产生一个特定类型的数组。因为参数需要实际的数据类型，代码必须包含 GetType() 函数(typeof()在 c#)来传输正确的类型。在接下来的例子中，使用 CopyDataOut 方法从图表的 ChartDataArray 对象中返回数据组。但是数据组的类型是 Single：

- Visual Basic

```
Dim xpo As Single()
Dim ypo As Single()
With C1Chart1.ChartGroups.ChartGroupsCollection(0).ChartData.SeriesList(0)
xpo = .X.CopyDataOut(GetType(Single()))
ypo = .Y.CopyDataOut(GetType(Single()))
End With
```

- C#

```
ChartDataSeries cds;
cds = c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData.SeriesList[0];
float[] xpo = cds.X.CopyDataOut(typeof(float[]));
float[] ypo = cds.Y.CopyDataOut(typeof(float[]));
```

注意： C#用户，CopyDataOut 的实现有一点不同。C#中，数组数据类型不能被设定到类型对象的变量中。如果对象被转换成了正确的数据类型，数组就能设定到对象中。

- Visual Basic

```
Dim xpo() As Single
xpo = CType(C1Chart1.ChartGroups.ChartGroupsCollection(0), Single())_
.ChartData.SeriesList(0).X.CopyDataOut(GetType(Single()))
```

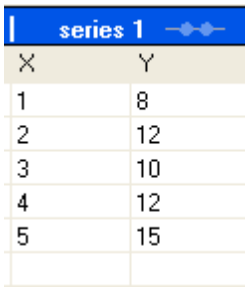
- C#

```
    float[] xpo;
    xpo =
(float[])c1Chart1.ChartGroups.ChartGroupsCollection[0].ChartData .SeriesList[0].X.CopyDataOut(type
of(float[]));
```

9.2.2 改变数据组的数据元素

将单独的数据元素集成到数据组的方法涉及到的 ChartDataSeries 和 ChartDataArray 对象。

使用 ChartDataArray 集合编辑器使得在.NET 属性窗口中设定单独的值非常的容易。X,Y,Y1，Y2，Y3 和 PointData 数据组都有数据组编辑器。点击 SeriesList 集合编辑器（在 ChartGroupsCollection 编辑器的 ChartData 节点下可用）旁边的省略符号就可以打开这些编辑器。和.NET 集合编辑器相同，ChartDataArray 编辑器在文本框的左边有数组的索引，右边显示数组的值。



注意 :ChartDataArray 集合编辑器有让显示格式在日期，日期时间和时间之间转换的函数。如果 ChartDataArray 值是日期时间类型，点击列头这个值就能在日期和时间格式之间转换。

运行时 ChartDataArray 元素和一般数据组元素一样是可用的，改变数组对象的值，只需简单改变一个数组的值。

- Visual Basic

```
    C1Chart1.ChartGroups.Group0.ChartData.SeriesList(0).X(4) = 4
```

- C#

```
    c1Chart1.ChartGroups.Group0.ChartData.SeriesList[0].X[4] = 4;
```

从 ChartDataArray 返回值需要一个相似的过程：

- Visual Basic

```
    Dim xval As Single
    xval = C1Chart1.ChartGroups.Group0.ChartData.SeriesList(0).X(4)
```

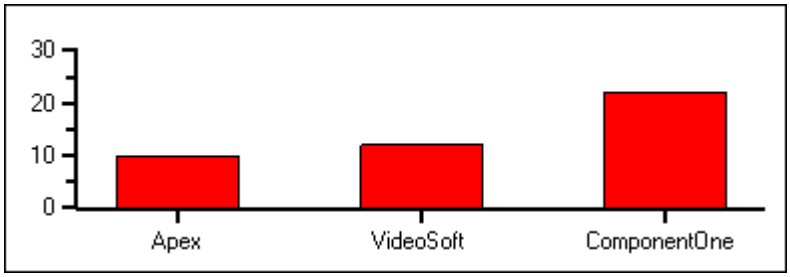
- C#

```
    float xval;
    xval = c1Chart1.ChartGroups.Group0.ChartData.SeriesList[0].X[4];
```

9.2.3 显示字符串值作为注释

有时，用字符串的值给 X 轴和 Y 轴作为注释，其他轴用数字作为注释，这是非常方便的。

例如，假设一个数据源包含两列，一个是公司名称（字符串），另一个是这个公司购买 ComponentOne 产品的数量。创建一个条形图来展示这些数据，你可以在 Y 轴创建一个产品数量的系列，然后创建一个数组用于计算数量，并把这个数组作为 X 值。然后给 X 轴创建 ValueLabels，每个公司名称和计算的数量相匹配，从而产生图表如下：



由于这种情况经常发生，C1Chart 接收 ChartDataArray 的字符串，ChartDataArray 会自动列举字符串的值，并且创建相关轴的 ValueLabels。使用这个功能，就有可能（并且非常容易）把 ChartDataSeries 中的 DataField 属性直接设定为一个字符串的值 ,并且也能得到字符串的值。

- Visual Basic

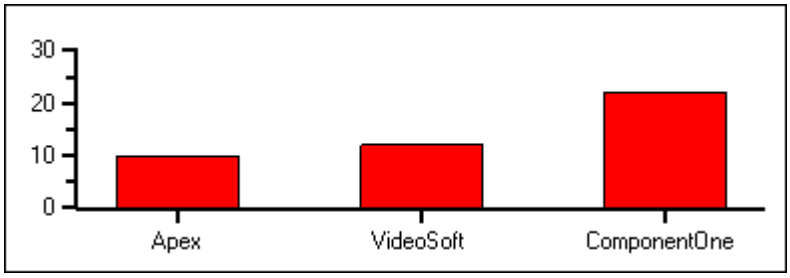
```
C1Chart1.ChartGroups(0).ChartData.SeriesList(0).X.DataField = "CompanyNames"
C1Chart1.ChartGroups(0).ChartData.SeriesList(0).Y.DataField = "UnitsPurchased"
```

- C#

```
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].X.DataField = "CompanyNames";
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].Y.DataField = "UnitsPurchased";
```

9.2.4 混合 ChartDataArray 输入

C1Chart 允许一些 ChartDataArrays 被绑定，其他的使用 ChartDataSeries 的 AutoEnumerate 属性 ,或者使用 ChartDataArray 的 CopyDataIn()方法来设定。当数据绑定时，图表使用所有的绑定数据作为定义系列数据的数据源 ,直接使用这些系列和其他图表属性 ,在图表的绘图区绘制出来。当 AutoEnumerate 属性设定为 True，ChartDataArrays/系列的 X 值自动设定为索引值。尽管有不同的方法设置数据系列，提供了最大灵活性，还是需要删除额外的默认系列。



例如 ,上面的条形图 ,只包含一个系列(包含一个公司购买 ComponentOne 产品的数量)。所以如果使用了数据绑定给 C1Chart 提供输入，那么 4 个默认图表系列中的 3 个就要被删除，

删除这些系列可以在设计时做，也可以在运行时做。

9.2.5 使用 Point 值设定 X 和 Y 数据组

对使用 X 和 Y 轴(XY-Polt ,Bar ,等等)的图表 ,PointData 属性非常方便 ,ChartDataSeries 对象的 Point 属性允许输入一组 PonitF 点值。这个属性对使用点数据的应用非常方便，但是这个数组只能设定 X 和 Y 数据组，这个属性不能改变 Y1，Y2,Y3 数据组的值。

下面的例子创建了一个 PointF 数组，并且把它装载到 ChartDataSersies 中：

- Visual Basic

```
Dim ps() As PointF = _
{ _
New PointF(30.0F, 20.0F), _
New PointF(60.0F, 24.0F), _
New PointF(90.0F, 42.0F), _
New PointF(120.0F, 13.0F), _
New PointF(150.0F, 10.0F) _
}
Dim s As New ChartDataSeries()
C1Chart1.ChartGroups.Group1.ChartData.SeriesList.Add(s)
s.PointData.CopyDataIn(ps)
```

- C#

```
PointF [] ps =
{
new PointF(30.0F, 20.0F),
new PointF(60.0F, 24.0F),
new PointF(90.0F, 42.0F),
new PointF(120.0F, 13.0F),
new PointF(150.0F, 10.0F)
};
ChartDataSeries s = new ChartDataSeries();
c1Chart1.ChartGroups.Group1.ChartData.SeriesList.Add(s);
s.PointData.CopyDataIn(ps);
```

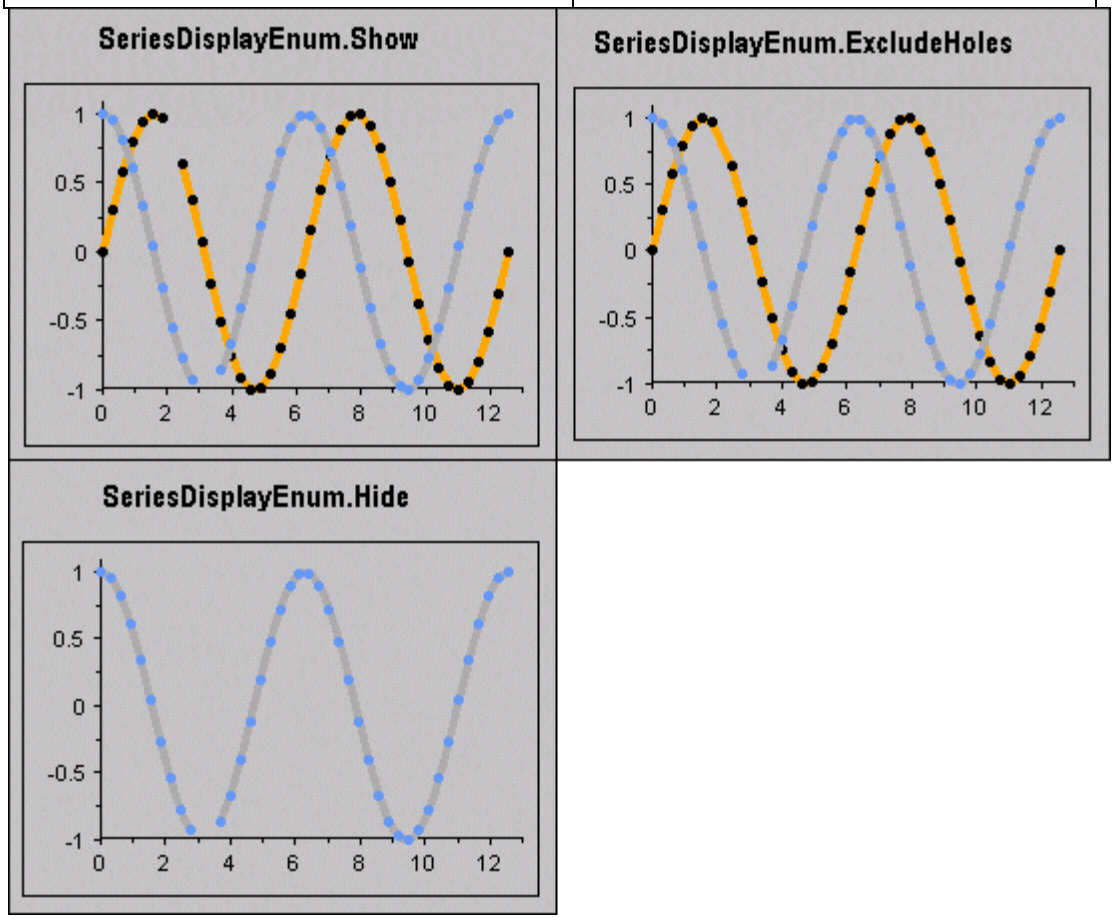
9.3 定制 ChartDataSeries

本章节描述在 ChartArea 中 ChartDataSeries 如何显示和隐藏排除漏洞，以及如何在 C1Chart 的特定图例中如何排除一个系列

9.3.1 显示、排除或隐藏一个系列

假设有成百个系列需要显示在图表中，由于图表只能这么大，必须控制显示的系列。ChartDataSeries 的 Display 属性提供了这种功能。Display 属性接受一个 SeriesDisplayEnum 枚举类型。把这个属性设定成不同的值，就允许一个系列被显示，被隐藏或者被排除

值	描述
SeriesDisplayEnum.Show	系列显示在 ChartArea 中，这是默认值。
SeriesDisplayEnum.Hide	系列不显示在 ChartArea 中，但是 ChartArea(最大值和最小值)不会因为系列没有显示而改变。
SeriesDisplayEnum.Exclude	系列不会显示在 ChartArea 中，但是 ChartArea 会因为系列没有显示而改变大小。
SeriesDisplayEnum.ExcludeHoles	系列显示，但是 Data Hole 值被忽略。



9.3.2 从图例中除去一个系列

有时图形的形状，线和曲线不代表数据，更希望代表数据区域，例如代表误差范围，等等。在这种情况下，多余的系列可能被加到图形图表中，但是应该在图例中除去，每个 ChartDataSeries 包含一个叫做 LegendEntry 的 Boolean 属性，如果设定成 True，也就是默认

值，该系列将列在图例中，如果是 False,这个系列还是会被画出来，但是不会别列到图例中。

9.4 绘制不规则的数据

数据集,尤其当他们是动态创建的,有时会有不规则的属性,很难让一个绘图控件来处理。

因此，在把数据绘制到图表之前，花费了大量的时间，计算这些不一致的数据。虽然 C1Chart 不能自动改变这些值，但是它能够接受不规则的数据，然后用一种逻辑和一致的方式来处理这些数据。

9.4.1 不规则堆积图表数据

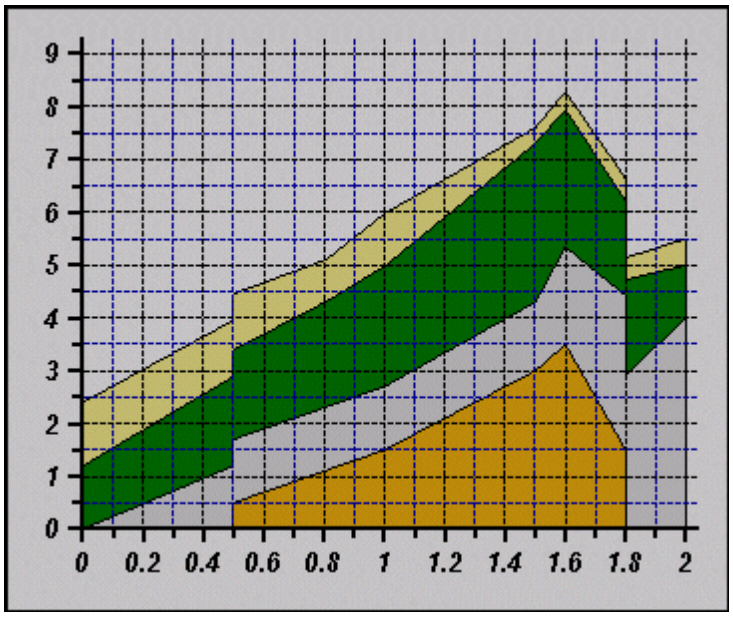
堆积图表是指图表在一个基础的系列上，堆积一个或多个系列。图表中所有 Y 值都指的是和前一个系列 Y 值的间距。当数据装载在 X 的值与数组中的完全匹配时，堆积图表是一个非常简单是事情，所有的系列都堆积得非常整齐。

C1Chart 在一般布局中为接收数据提供了更多的自由。一般布局意味着不是所有的系列都必须有完全相同的 X 值，这种自由使图表接收了一些非标准的数据，在使用堆积图表时就会遇到一些困难。

9.4.2 不规则 X 轴数据

如果数据集中的第一个系列不包含和其他系列的最小的点和最大的点相匹配的点，那么堆积图表就会遇到更大的困难。

如果图表的第一个系列没有足够数量的 X 值，不能成为一个连续的系列，那么这个系列将不能像下图一样有一个明确的开始点。C1Chart 用一种逻辑的方式处理这些非规则的数据，第一个系列数据的 Y 值不存在，将作为 0 处理，然后每个连续的系列都被添加了这些数据从而产生了中断，下图就是个最好的示例。



9.4.3 插入 Y 值数据

堆积图表时如果一个连续系列的 X 值和第一个系列不一致也会给推挤图表增加困难。例如，

图表的第一个系列 X 值有 2,4,6 和 8，而第二个系列有值 3,4,5 和 9.图表在判断在哪个位置放置第二个系列的第一个点的时候就会遇到难题。第二系列的点的值必须添加 Y 值数据来定位点在第一个系列的头两个点之间的某处，而这个数据时数据集中没有的。

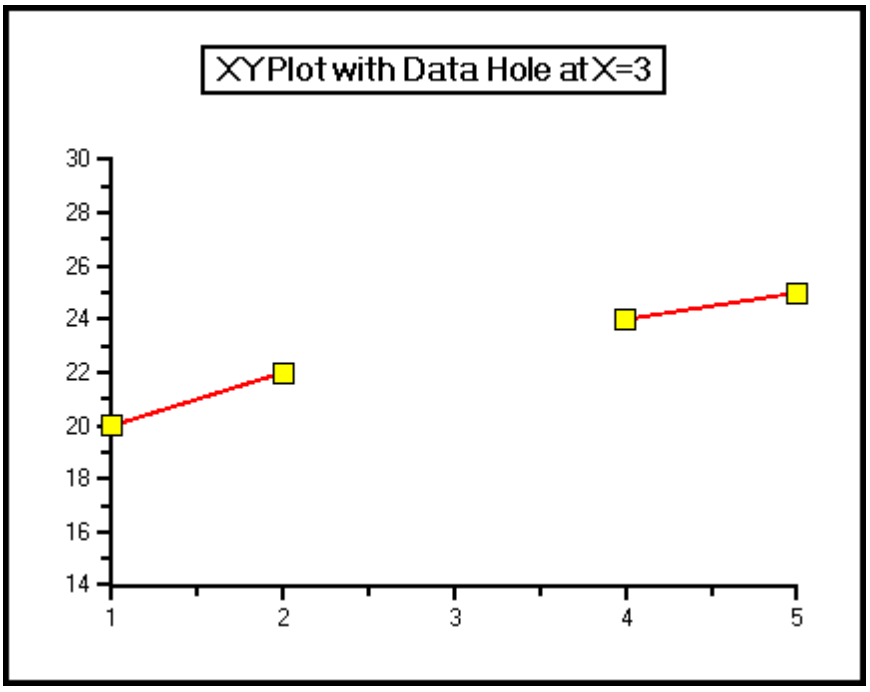
不表示数据集中的数据，C1Chart 以逻辑的方式处理这些不规则数据。根据线的走势，C1Chart 在第一个系列的前两个点之间插入一个虚拟的点。这个插入的点的值加到第二个系列的 Y 值上从而找到堆积图表用的新的点。

9.5 指定 Data Hole

使用组织好的数据作图，有时候必须在图表中标示出不可用的特殊的点。例如，你能标示出缺少的信息。一个 Data Hole 是对正在绘制的数据一个打断，用于在图表中标示出缺失的数据。

在 C1Chart 中，Hole 是指一个数据点不具有其他数据所有的显著特性，或者在一个数据组中是没有用的数据。例如，假设 Hole 的值被设定为-1000，如果一个系列的第三个点的数据不可用，一个中断或一个漏洞就会产生，绘制这个点的 Y 值到-1000.

默认的 Hole 值对 Visual Basic 来说是 Single MaxValue ，对 C#是 float MaxValue。下图是一个在 X—Y Plot 图表中数据漏洞的例子



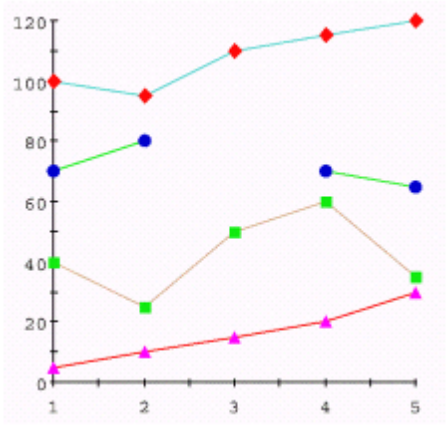
9.5.1 忽略 Data Hole

当数组中的数据有着相同的长度，所有的数组都很一致的情况下，编程是非常简单的。即使有一些数据点丢失了，也不希望标示出这些丢失的数据。在这种情况下，还是可以使用 Data Hole 作为丢失数据的值，C1Chart 可以很简单的完全忽略它们，就好像数据组元素完全没有 Data Hole 一样。

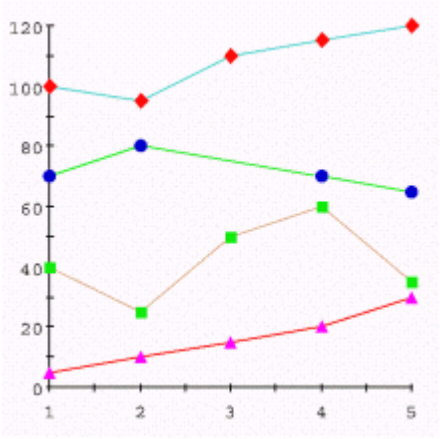
ChartDataSeries 的 Display 属性控制一个系列是否显示。这个属性接收一个 SeriesDisplayEnum 枚举。 如果 Display 属性设定成了 SeriesDisplayEnum.Show，那么 Data Hole 会像上图一样排除到图表之外。 如果属性设定为 SeriesDisplayEnum.ExcludeHoles，系

列会显示 ,但是漏洞被忽略。这就意味着如果显示的是 XY-Plot ,Data Hole 前面的点和 Data Hole 后面的点直接连起来如下图所示。注意即使绘制的线穿过了 Data Hole，Data Hole 点还是没有被画出来的。

SeriesDisplayEnum.Show



SeriesDisplayEnum.ExcludeHoles



9.6 绘图函数

C1Chart 有一个绘图函数的内置引擎。因为不同的应用使用不同类型的函数给，C1Char 为许多的应用提供了不同类型的函数。

提供的函数有两种类型

1. 一个变量的显式函数。
   - 一个变量的显式函数　例如 y=f(x) （参照 YFunction 类）
   - 几个例子包括：有理函数（rational），线性函数 （linear），多项式函数（polynomial），二次函数 （ quadratic），对数函数（logarithmic），和指数函数（exponential）。
   - 科学家和工程师经常使用的函数，这些函数可以运用在财务，预报，性能测定等等
2. 参数函数
   - 由一对方程式定义的函数，例如 y=f1(t) 和 x=f2(t) 这里 t 是函数 f1 和 f2 的变量/坐标

- 参数函数是特殊类型的函数，因为 X 和 Y 坐标有两个独立的函数定义，是分开的变量
- 参数函数在数学和功能领域代表不同的情况，热传输，水力学，电磁理论，行星运动和相对论理论的一些方面使用了一些这样的函数

关于参数函数的更多信息 参照 ParametricFunction 类在 C1Chart 中，一个代码串可以用来计算 Y 函数和参数函数要使用一个代码串来计算 Y 函数和参数函数，你必须执行下面之一

- 一个包含 C#或者 VB 源代码的解释用字串
- 事件
- 一个支持 C1.Win.C1Chart.ISimpleFunction 层的类

C1Chart 有一个 FunctionBase 集合编辑器， 这个编辑器可以通过 C1Chart 的属性在设计时访问。FunctionBase 集合编辑器由 Windows 窗口组成，可以允许用户很方便地编辑/创建函数。 使用这个编辑器，用户可以新增/删除一个过多个函数，选择函数种类（Y 函数或者参数函数）并定义代码类型和代码的语言，就能命名一些函数。关于 FunctionBase 集合编辑器的更多信息，参照 FunctionBase 集合编辑器（47 页）。

例子：说明如何创建 Y 函数和参数函数的完整实例，参照实例，FExplorer，位于 http://helpcentral.componentone.com/Samples.aspx.

9.6.1 使用代码串定义函数

当一个解释用代码串用于定义一个函数类中的一个函数（YFunction 或者 ParametricFunction），代码串被编译，编译后的代码动态地被包含到应用中。执行速度和其他编译的代码一样。

这样就有一些优点：

- 当定义函数时，内置的 C#函数或者 VB 函数可以被使用
- 简单地操作代码串就能创建函数。
- 使用 SaveChartToFile() 或者 SaveChartToString() 方法能将函数作为图表布局的一部分保存下来，并且可以作为数据的一部分直接装载到其他工程中。

但是也存在一些缺点：

- 由于要预编译第一次分配和改变代码内容时比较浪费时间
- 源代码的类型只能是 C#或者 VB

为了简单，使用一个变量的显式函数 YFunction 类对象．这个对象有一个 c o d e 属性，CodeText。对 YFunction 对象，单独的变量始终被假设为 X

对参数函数 ,必须使用 ParametricFunction 类对象定义一对方程式。这个对象有两个属性，每个坐标有一个。属性 CodeTextX 和 CodeTextY 接受编码，每个单独变量始终被假设为 t

对上面的两个类对象，单独的变量的值被假定为从最小值到最大值之间是均匀分布的，使用 PlotNumPoints 定义数据点的数量。

对于复杂的函数，使用 CodeText, CodeTextX 和 CodeTextY 提供 3 种编码类型。编码类型包括 Formula, Method 和 Unit，下面的章节逐一进行介绍。

9.6.1.1 Formula 编码类型

对应 Formula 编码类型，一个函数的 CodeText 属性必须包含使用一个参数计算函数值的代码。代码必须写到一行并且代表方程式的右半边。

- Visual Basic

```
Dim yf As C1.Win.C1Chart.YFunction = New C1.Win.C1Chart.YFunction()
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Formula
yf.CodeLanguage = C1.Win.C1Chart.FunctionCodeLanguageEnum.VB
yf.CodeText = "x*x"
yf.MinX = -5
yf.MaxX = 5
yf.LineStyle.Color = Color.Red
yf.LineStyle.Thickness = 3
C1Chart1.ChartGroups(0).ChartData.FunctionsList.Add(yf)
```

- C#

```
C1.Win.C1Chart.YFunction yf = new C1.Win.C1Chart.YFunction();
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Formula;
yf.CodeText = "x*x";
yf.MinX = -5;
yf.MaxX = 5;
yf.LineStyle.Color = Color.Red;
yf.LineStyle.Thickness = 3;
c1Chart1.ChartGroups[0].ChartData.FunctionsList.Add(yf);
```

9.6.1.2 Method 编码类型

对于 Method 编码类型，一个函数类的 CodeText 属性必须包含方法的主体，这个方法用于计算函数的值并且能明确地返回值。希望的返回值的类型是 Double。注意 VB 语法要求在编码中每个声明的结尾都需要有 vbNewLines

- Visual Basic

```
Dim code As String = _
"Dim x2 As Double = x*x" & vbNewLine & _ "if x<0 then " & vbNewLine & _ " return x" & vbNewLine & _ "else" & vbNewLine & _ " return 0.5*x2" & vbNewLine & _
" End If"
Dim yf As C1.Win.C1Chart.YFunction = New C1.Win.C1Chart.YFunction()
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Method
yf.CodeLanguage = C1.Win.C1Chart.FunctionCodeLanguageEnum.VB
yf.CodeText = code
yf.MinX = -5
yf.MaxX = 5
yf.LineStyle.Color = Color.Blue
yf.LineStyle.Thickness = 3
C1Chart1.ChartGroups(0).ChartData.FunctionsList.Add(yf)
```

- C#

```
string code ="double x2 = x*x;" +
"if( x<0)" +
" return x;" +
"else" +
" return 0.5*x2;";
C1.Win.C1Chart.YFunction yf = new C1.Win.C1Chart.YFunction();
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Method;
yf.CodeText = code;
yf.MinX = -5;
yf.MaxX = 5;
yf.LineStyle.Color = Color.Blue;
yf.LineStyle.Thickness = 2;
c1Chart1.ChartGroups[0].ChartData.FunctionsList.Add(yf);
```

9.6.1.3 Unit 编码类型

对于 Unit 编码类型，一个函数的 CodeText 属性必须包含完全编译的 Unit 文本。Unit 必须在 UserFunction namespace 中包含"Calculator"类，而且必须执行 ISimpleFunction 接口。

- Visual Basic

```
Dim code As String = _
"Namespace UserFunction" & vbNewLine & _
" Class Calculator" & vbNewLine & _
" Implements ISimpleFunction" & vbNewLine & _
" Public Function Calculate(x As Double) As Double _" & vbNewLine & _ " Implements ISimpleFunction.Calculate" & vbNewLine & _ " Dim x2 As Double = x*x" & vbNewLine & _ " if( x<0)" & vbNewLine & _ " return -x" & vbNewLine & _ " else" & vbNewLine & _ " return -0.5*x2" & vbNewLine & _ " End If" & vbNewLine & _ " End Function" & vbNewLine & _ " End Class" & vbNewLine & _ "End Namespace"
Dim yf As C1.Win.C1Chart.YFunction = New C1.Win.C1Chart.YFunction()
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Unit
yf.CodeLanguage = C1.Win.C1Chart.FunctionCodeLanguageEnum.VB
yf.CodeText = code
yf.MinX = -5
yf.MaxX = 5
yf.LineStyle.Color = Color.Green
yf.LineStyle.Thickness = 3
C1Chart1.ChartGroups(0).ChartData.FunctionsList.Add(yf)
```

- C#

```
string code =
"namespace UserFunction" +
"{" +
" class Calculator : ISimpleFunction" + " {" +
" public double Calculate(double x)" + " {" + " double x2 = x*x;" + " if( x<0)" + " return -x;" + " else" + " return -0.5*x2;" + " }" + " }" + "}";
C1.Win.C1Chart.YFunction yf = new C1.Win.C1Chart.YFunction();
yf.CodeType = C1.Win.C1Chart.FunctionCodeTypeEnum.Unit;
yf.CodeText = code;
yf.MinX = -5;
yf.MaxX = 5;
yf.LineStyle.Color = Color.Green;
yf.LineStyle.Thickness = 2;
c1Chart1.ChartGroups[0].ChartData.FunctionsList.Add( yf);
```

9.6.2 计算函数的值

下面的内容介绍如何使用 CalculateX 和 CaleculateY 事件或者 ISimpleFunction 接口的 Calculate 方法计算恰当的函数的值。

9.6.2.1 使用事件计算函数的值

不想把源代码加到 CodeText, CodeTextX 和 CodeTextY 属性，那么就可以使用事件委托定义一个事件方法来计算恰当的函数的值。

当使用事件委托，对于 YFunction 类对象，在 CalculateY 事件中计算函数的值。对于 ParametricFunction 类对象 ,两个事件 ,CalculateX 和 CalculateY 必须被设定 ,用于计算坐标。非空的事件委托说明了事件应该被用于计算对应的函数的值，即使对应的 CodeText 属性在 Yfunction 或者 ParametricFunction 类对象中已经被定义了。 为了使用事件，程序员必须提供合适的事件处理程序。

- Visual Basic

```
    Private Sub Button_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs)
Handles Button.Click
    Dim yf As C1.Win.C1Chart.YFunction = New C1.Win.C1Chart.YFunction()
    AddHandler yf.CalculateY, AddressOf Function_Calculate
    yf.MinX = -5
    yf.MaxX = 5
    yf.LineStyle.Color = Color.DarkBlue
    yf.LineStyle.Thickness = 3
    C1Chart1.ChartGroups(0).ChartData.FunctionsList.Add(yf)
    Private Sub Function_Calculate(ByVal sender As Object, _ ByVal e As
C1.Win.C1Chart.CalculateFunctionEventArgs)
    e.Result = e.Parameter * e.Parameter * e.Parameter ' y = x*x*x
    End Sub
```

- C#

```
    private void button_Click(object sender, System.EventArgs e)
    {
    C1.Win.C1Chart.YFunction yf = new C1.Win.C1Chart.YFunction();
    yf.MinX = -5;
    yf.MaxX = 5;
    yf.LineStyle.Color = Color.DarkBlue;
    yf.LineStyle.Thickness = 2;
    yf.CalculateY += new C1.Win.C1Chart.CalculateFunctionEventHandler( Function_Calculate);
    c1Chart1.ChartGroups[0].ChartData.FunctionsList.Add( yf);
    }
    void Function_Calculate( object sender, C1.Win.C1Chart.CalculateFunctionEventArgs e)
    {
    e.Result = e.Parameter * e.Parameter * e.Parameter; // y = x*x*x
    }
```

9.6.2.2 使用 Calculate 方法计算函数的值

作为使用 code string 或者 event 的替代方法，程序员还可以执行 C1.Win.C1Chart.ISimpleFunction 层定义一个对象实例。这个对象必须从该层继承并且实现一个公共的函数名，Calculate。这个 Calculate 方法有单独的变量（Double）作为一个参数，并且返回单独的变量（Double）。

对于 YFunction 类对象，CustomFunction 属性必须被设定为 ISimpleFunction 执行对象。对于 ParametricFunction 类对象，CustomFunctionX 和 CustomFunctionY 属性必须被设定为相应对象执行的 ISimpleFunction 接口。

- Visual Basic

```
    Private Sub Button_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs)
Handles Button.Click
    Dim yf As C1.Win.C1Chart.YFunction = New C1.Win.C1Chart.YFunction()
    yf.CustomFunction = New CustomFunction()
    yf.MinX = -5
    yf.MaxX = 5
    yf.LineStyle.Color = Color.DarkGreen
    yf.LineStyle.Thickness = 3
    C1Chart1.ChartGroups(0).ChartData.FunctionsList.Add(yf)
    End Sub
    Public Class CustomFunction
    Implements C1.Win.C1Chart.ISimpleFunction
    Public Function Calculate(ByVal x As Double) As Double _ Implements
C1.Win.C1Chart.ISimpleFunction.Calculate
    Return -x * x * x ' y = - x*x*x
    End Function
    End Class
```

- C#

```
private void button_Click(object sender, System.EventArgs e)
{
C1.Win.C1Chart.YFunction yf = new C1.Win.C1Chart.YFunction();
yf.MinX = -5;
yf.MaxX = 5;
yf.LineStyle.Color = Color.DarkGreen;
yf.LineStyle.Thickness = 2;
yf.CustomFunction = new CustomFunction();
c1Chart1.ChartGroups[0].ChartData.FunctionsList.Add( yf);
}
class CustomFunction : C1.Win.C1Chart.ISimpleFunction
{
public double Calculate( double x)
{
return -x*x*x; // y = -x*x*x
}
}
```

9.7 使用趋势线

趋势线由图表的 TrendLine 对象提供 ,可以分为两组 ,包括回归和非回归。在 2D 的图表中，趋势线一般使用 X-Y 先，条状图，或者散点图。

非回归的趋势线有移动平均 ,平均 ,最小和最大。移动平均趋势线是在指定时间内平均分布。

回归趋势线表示的是包括多项式，指数，对数和傅立叶函数这类关于数据的函数趋势。

9.7.1 创建趋势线

在设计时创建趋势线

使用趋势线集合编辑器（TrendLine Collection Editor）可以在设计时创建趋势线。使用这个集合编辑器，你可以添加，修改和删除趋势线，关于趋势线集合编辑器的更多信息，请参见 TrendLine Collection Editor （54 页).

以编程方式创建趋势线

以编程的方式创建趋势线，创建 TrendLine 对象实例并且设置它的属性。趋势线创建可以使用构造器或者使用 TrendLinesCollection 的 AddNewTrendLine.TrendLinesCollection()方法。

下面的编码给图表中的 ChartGroup(0)添加了红色的趋势线。

- Visual Basic

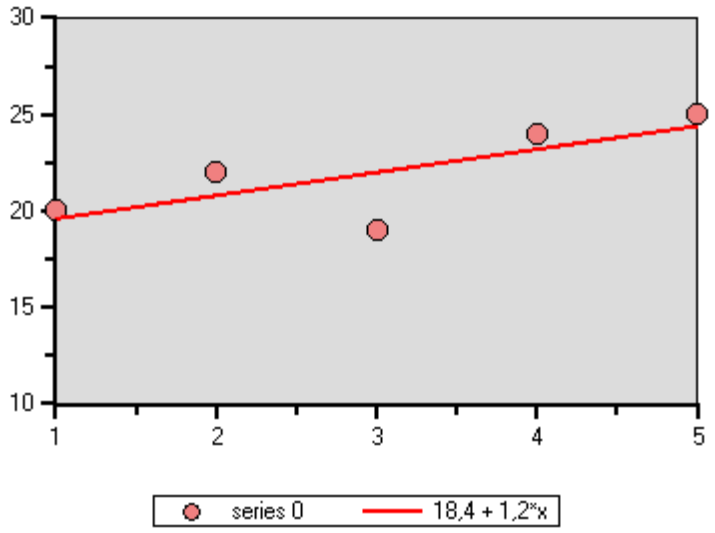
```
Dim tl As C1.Win.C1Chart.TrendLine = New C1.Win.C1Chart.TrendLine()
tl.SeriesIndex = 0
tl.LineStyle.Color = Color.Red
tl.LineStyle.Thickness = 2
tl.Text = "{#FORMULA}"
c1Chart1.ChartGroups(0).ChartData.TrendsList.Add(tl)
```

- C#

```
C1.Win.C1Chart.TrendLine tl = new C1.Win.C1Chart.TrendLine();
tl.SeriesIndex = 0;
tl.LineStyle.Color = Color.Red;
tl.LineStyle.Thickness = 2;
tl.Text = "{#FORMULA}";
c1Chart1.ChartGroups[0].ChartData.TrendsList.Add( tl);
```

图例如下

红色的先连接了第一个和最后一个数据点。



9.7.2 回归趋势线

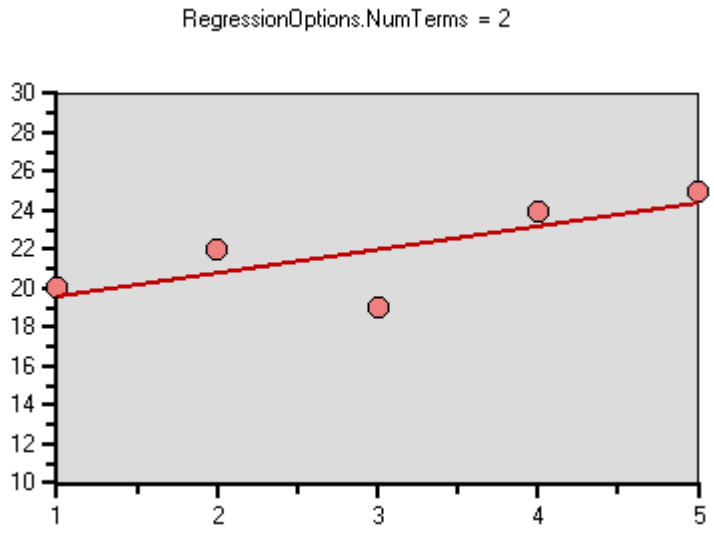
回归趋势线有两个相关的类对象，分别为 RegressionOptions 和 RegressionStatistics.这些对象描述了趋势线所希望的形式，通过统计合计结果趋势线。

9.7.2.1 回归选项

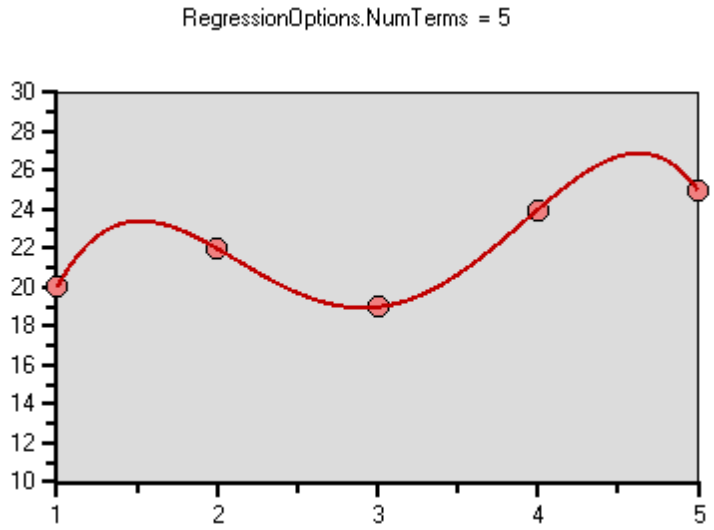
RegressionOption 对象允许回归模型的规范。NumTerms 属性定义一个回归所需要的系数的个数，和回归有关的仅是项的不等的个数（多项式和傅里叶）。

对于一个多项式回归，项的个数要比由此产生的多项式的阶多一。一个多项式的最多的项数是数据点的数量，最小的项数是二（直线）

下面的图表示了一个直线的回归（由于有 2 个项）

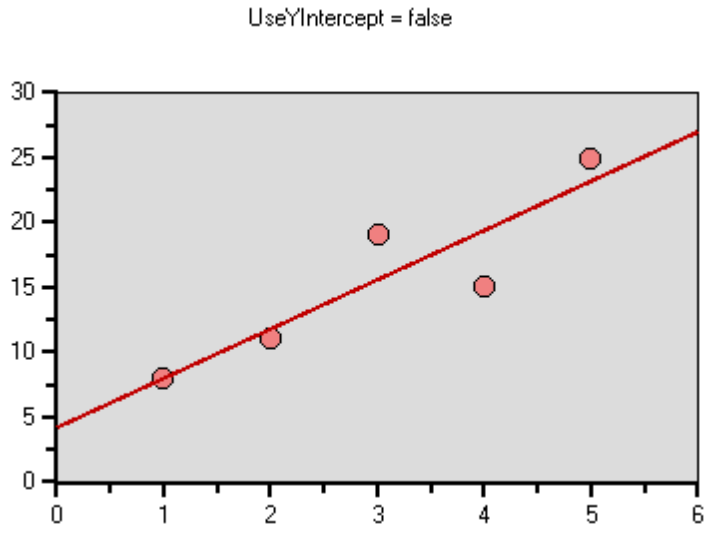


下面的图表示一个多项式回归（项数超过 2）

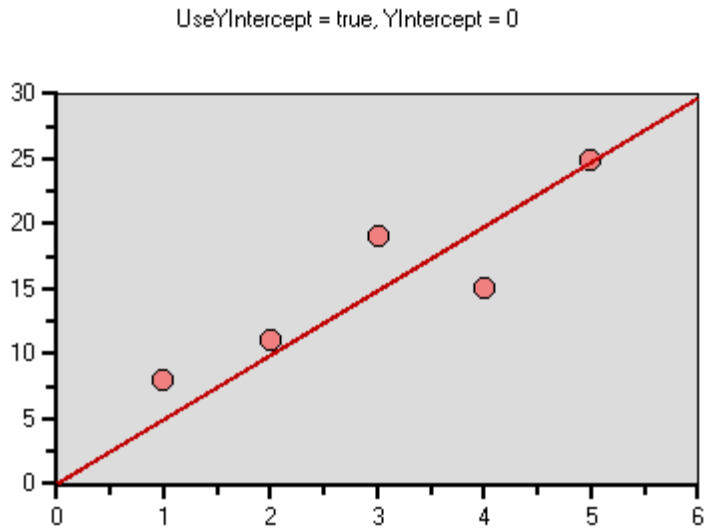


UseYIntercept 属性控制多项式回归的第一个项数是否是固定的 ,当 UseYIntercept 是 True，趋势线截取 Yintercept 属性定义的点作为 Y 轴上 X = 0 的点

下图描述了 UseYIntercept 属性是 False 时的一个非固定的直线回归线



下图表示了一个固定的直线回归 ,因为趋势线设定为截取 RegressionOptions.Yintercept 属性定义的在 Y 坐标上 X=0 的线。



9.7.2.2 回归统计

RegressionStatistics 属性返回了一个包含回归模型的统计结果的 RegressionStatistics 对象。当趋势线不是回归的或者现在的数据不能解决回归 ,RegressionStatistics 属性返回 Nothing(VB) 或者空（ C# ）。

下面的表定义了统计方程式所需要的值。

值	定义
n	点的个数
nt	回归系数的个数
x 值	错误！对象不能从编辑域代码中不能被创建。

y 值	错误！对象不能从编辑域代码中不能被创建。
mean y 值	错误！对象不能从编辑域代码中不能被创建。
Fitted y 值	错误！对象不能从编辑域代码中不能被创建。

下面的表表示 RegressionStatistics 对象的名称，描述和语法。

名称	描述	公式	公式（UseYIntercept=true）
Ssr	平方和取决于回归统计	错误！对象不能从编辑域代码中不能被创建。	错误 对象不能从编辑域代码中不能被创建。
Sse	平方和取决于错误	错误！对象不能从编辑域代码中不能被创建。	错误 对象不能从编辑域代码中不能被创建。
Rsq	确定系数	错误！对象不能从编辑域代码中不能被创建。	错误 对象不能从编辑域代码中不能被创建。
DF	自由度数	错误！对象不能从编辑域代码中不能被创建。	错误 对象不能从编辑域代码中不能被创建。
F	F 值（F 统计）	错误！对象不能从编辑域代码中不能被创建。	错误 对象不能从编辑域代码中不能被创建。

9.7.3 自定义趋势线

实现一个自定义的趋势线，一个类必须实现创建 ICustomTrendLine 接口。这个类的实例被分配给一个 TrendLine 对象的 CustomTrendLine 属性。这种情况下趋势线的所有的点都必须由类完全定义，并且 TrendLineType 属性的设定是不重要的。下面的示例编码实现了一个和数据范围同步的自定义趋势线。

- Visual Basic

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _
Handles Button1.Click
Dim tl As C1.Win.C1Chart.TrendLine = _
c1Chart1.ChartGroups(0).ChartData.TrendsList.AddNewTrendLine()
tl.LineStyle.Thickness = 3
tl.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.Dash
tl.CustomTrendLine = New CustomTrendLine()
End Sub
Public Class CustomTrendLine_ Implements C1.Win.C1Chart.ICustomTrendLine
Private _x() As Double
Private _y() As Double
Public Sub Calculate(ByVal tl As C1.Win.C1Chart.TrendLine, ByVal x() As Double, _ ByVal y() As
Double)Implements C1.Win.C1Chart.ICustomTrendLine.Calculate
If x Is Nothing Or x.Length = 0 Or y Is Nothing Or y.Length = 0 Then
_x = Nothing
_y = Nothing
Return
End If
Dim xmin As Double = x(0), xmax = x(0)
Dim ymin As Double = y(0), ymax = y(0)
Dim i As Integer
For i = 1 To x.Length - 1
If x(i) < xmin Then
xmin = x(i)
ElseIf x(i) > xmax Then
End If
If y(i) < ymin Then
ymin = y(i)
ElseIf y(i) > ymax Then
ymax = y(i)
End If
Next
_x = New Double(4) {}
_y = New Double(4) {}
_x(0) = xmin
_y(0) = ymin
_x(4) = _x(0)
_y(4) = _y(0)
_x(2) = xmax
_y(2) = ymax
_x(1) = _x(0)
_y(1) = _y(2)
_x(3) = _x(2)
_y(3) = _y(0)
End Sub
Public Function GetXValues() As Double() _ Implements
C1.Win.C1Chart.ICustomTrendLine.GetXValues
Return _x
End Function
Public Function GetYValues() As Double() _ Implements
C1.Win.C1Chart.ICustomTrendLine.GetYValues
```

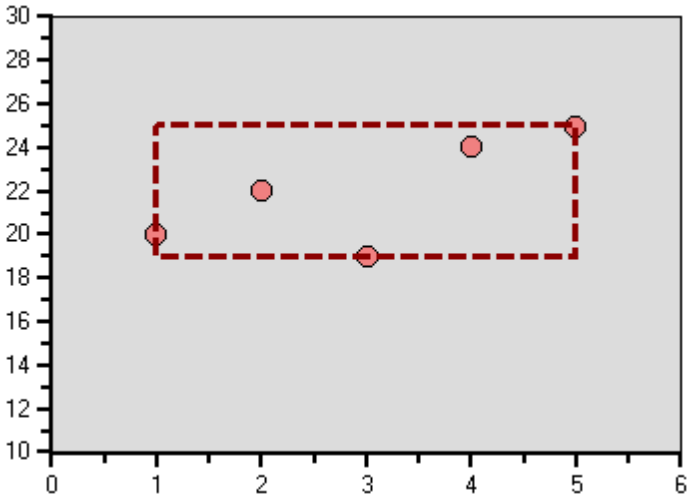
- C#

```
        private void button1_Click(object sender, System.EventArgs e)
        {
        C1.Win.C1Chart.TrendLine tl =
c1Chart1.ChartGroups[0].ChartData.TrendsList.AddNewTrendLine();
        tl.LineStyle.Color = Color.DarkRed;
        tl.LineStyle.Thickness = 3;
        tl.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.Dash;
        tl.CustomTrendLine = new CustomTrendLine();
        }
        public class CustomTrendLine : C1.Win.C1Chart.ICustomTrendLine
        {
        private double[] _x;
        private double[] _y;
        public void Calculate( C1.Win.C1Chart.TrendLine tl, double[] x, double [] y)
        {
        if( x==null || x.Length==0 || y==null || y.Length==0)
        {
        _x = null; _y = null;
        return;
        }
        double xmin = x[0], xmax = x[0];
        double ymin = y[0], ymax = y[0];
        for( int i=1; i<x.Length; i++)
        {
        if( x[i] < xmin)
        xmin = x[i];
        else if( x[i] > xmax)
        xmax = x[i];
        if( y[i] < ymin)
        ymin = y[i];
        else if( y[i] > ymax)
        ymax = y[i];
        }
        _x = new double[5];
        _y = new double[5]; _x[0] = xmin; _y[0] = ymin;
        _x[4] = _x[0]; _y[4] = _y[0];
        _x[2] = xmax; _y[2] = ymax;
        _x[1] = _x[0]; _y[1] = _y[2];
        _x[3] = _x[2]; _y[3] = _y[0];
        }
        public double[] GetXValues() { return _x;}
        public double[] GetYValues() { return _y;}
        public double GetY( double x) { return 0;}

        public string Text { get{ return "Custom trend";}}
        }
```

上面的内容做出下面的图表

围绕着数据点创建了一个自定义的红色的虚线的趋势线。



9.8 使用 PointStyles

PointStyle 提供一种机制可以使用不同的视觉效果标注特定的数据点以便和同一个数据系列的其他数据点区别开来。PointStyles 包含在 PointStylesCollection 中。一个 PointStyle 可以被提供给明确的数据点 ,这个数据点从系列中通过点索引来选择或者这个数据点是特殊情况下的点 , 例如 : 系列的 X 最大值 , 所有数据的 y 最大值等等。C1Chart 程序员还可以定义使用 Select 事件的客户化条件。PointStyle 包含和 ChartDataSeries LineStyle, SymbolStyle 以及 Offset 相同的一套 visual 属性。这些属性描述了遇到 PointStyle 的客户化条件是数据点的外观。一个 PointStyle 可以以图例项表示出来。

9.8.1 创建 PointStyles

PointStyles 在设计时能够很容易地通过 PointStyles 集合编辑器被创建或者通过 PointStyle 对象编程实现。

在设计时创建 PointStyles

PointStyles 在设计时可以通过 PointStyle 集合编辑器创建。使用集合编辑器 ,你可以新增 , 修改和删除点类型。关于 PointStyle 集合编辑器属性的更多信息请参照 PointStyle 集合编辑器（52 页）。

用程序创建 PointStyles

下面的代码创建了一个 PointStyle 对象的实例 ,并且设定 LineStyle 和 SymbolStyle 属性。

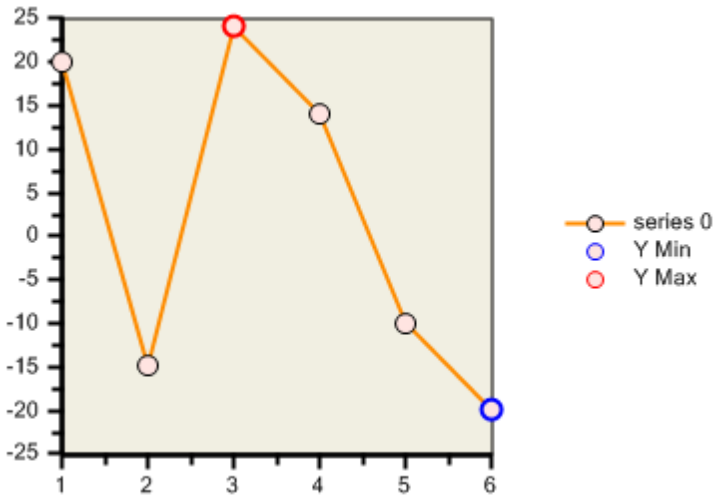
- Visual Basic

```
Dim styles As C1.Win.C1Chart.PointStylesCollection = _
c1Chart1.ChartGroups(0).ChartData.PointStylesList
Dim psmin As C1.Win.C1Chart.PointStyle = styles.AddNewPointStyle()
psmin.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.None
psmin.SymbolStyle.Color = Color.MistyRose
psmin.SymbolStyle.OutlineColor = Color.Blue
psmin.SymbolStyle.OutlineWidth = 2
psmin.SymbolStyle.Size = 10
psmin.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.SeriesMinY
psmin.Label = "Y Min"
psmin.LegendEntry = True
Dim psmax As C1.Win.C1Chart.PointStyle = styles.AddNewPointStyle()
psmax.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.None
psmax.SymbolStyle.Color = Color.MistyRose
psmax.SymbolStyle.OutlineColor = Color.Red
psmax.SymbolStyle.OutlineWidth = 2
psmax.SymbolStyle.Size = 10
psmax.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.SeriesMaxY
psmax.Label = "Y Max"
psmax.LegendEntry = True
c1Chart1.Legend.Visible = True
```

- C#

```
C1.Win.C1Chart.PointStylesCollection styles =
c1Chart1.ChartGroups[0].ChartData.PointStylesList;
C1.Win.C1Chart.PointStyle psmin = styles.AddNewPointStyle();
psmin.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.None;
psmin.SymbolStyle.Color = Color.MistyRose;
psmin.SymbolStyle.OutlineColor = Color.Blue;
psmin.SymbolStyle.OutlineWidth = 2;
psmin.SymbolStyle.Size = 10;
psmin.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.SeriesMinY;
psmin.Label = "Y Min";
psmin.LegendEntry = true;
C1.Win.C1Chart.PointStyle psmax = styles.AddNewPointStyle();
psmax.LineStyle.Pattern = C1.Win.C1Chart.LinePatternEnum.None;
psmax.SymbolStyle.Color = Color.MistyRose;
psmax.SymbolStyle.OutlineColor = Color.Red;
psmax.SymbolStyle.OutlineWidth = 2;
psmax.SymbolStyle.Size = 10;
psmax.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.SeriesMaxY;
psmax.Label = "Y Max";
psmax.LegendEntry = true;
c1Chart1.Legend.Visible = true;
```

PointStyles 添加到了 C1Chart 的第一个系列的数据点。还添加了两个特殊点的类型表示 Y 轴的最小值的点和最大值的点。



9.8.2 创建客户化的 PointStyles

要定义客户化的点类型条件，用户必须设定 PointStyleSelectionEnum.Custom 的 Selection 属性，并且提供 Select 事件的事件处理程序。下面的编码创建了一个客户化的点类型。

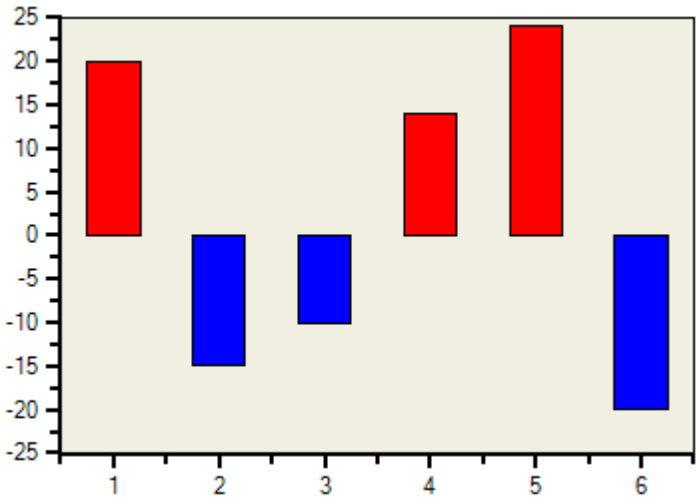
- Visual Basic

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)_
Handles Button1.Click
    Dim ps As C1.Win.C1Chart.PointStyle = New C1.Win.C1Chart.PointStyle()
    ps.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.Custom
    AddHandler ps.Select, AddressOf PS_Select
    c1Chart1.ChartGroups(0).ChartData.PointStylesList.Add(ps)
    End Sub
    Private Sub PS_Select(ByVal sender As Object, ByVal e As _
C1.Win.C1Chart.PointStyleSelectEventArgs)
    Dim ps As C1.Win.C1Chart.PointStyle = CType(sender, C1.Win.C1Chart.PointStyle)
    Dim ds As C1.Win.C1Chart.ChartDataSeries = _
c1Chart1.ChartGroups(0).ChartData(e.SeriesIndex)
    Dim y As Double = Convert.ToDouble(ds.Y(e.PointIndex))
    If (y < 0) Then
    Else
    ps.LineStyle.Color = Color.Red
    End If
    e.Selected = True
    End Sub
```

- C#

```
private void button1_Click(object sender, System.EventArgs e)
{
C1.Win.C1Chart.PointStyle ps = new C1.Win.C1Chart.PointStyle();
ps.Selection = C1.Win.C1Chart.PointStyleSelectionEnum.Custom;
ps.Select += new C1.Win.C1Chart.PointStyleSelectEventHandler(PS_Select);
c1Chart1.ChartGroups[0].ChartData.PointStylesList.Add( ps);
}
void PS_Select( object sender, C1.Win.C1Chart.PointStyleSelectEventArgs e)
{
C1.Win.C1Chart.PointStyle ps = sender as C1.Win.C1Chart.PointStyle;
C1.Win.C1Chart.ChartDataSeries ds = c1Chart1.ChartGroups[0].ChartData[e.SeriesIndex];
double y = Convert.ToDouble( ds.Y[e.PointIndex]);
if( y<0)
ps.LineStyle.Color = Color.Blue;
else
ps.LineStyle.Color = Color.Red;
e.Selected = true;
}
```

Y 值小于 0 定制了蓝色的点类型，y 值大于 0 定制了红色的点类型。



10. 数据绑定

数据绑定是一个允许一个或多个数据使用者以同步的方式和一个数据提供者联接的过程。如果你编辑一个绑定数据集的一部分数值 ,联接到相同数据源的 C1Chart 控件将反映新的数据集。

和许多绑定控件不同，C1Chart 并不使用当前值。当数据绑定时，图表使用所有的绑定数据作为其数据源来定义系列数据，在图表的绘图区使用系列数据和其他的图表属性直接绘制出来。

这个过程需要简单的几步，但是需要图表对象模型的一些知识。

首先，必须创建一个数据源，许多数据源都可用，包括 ADO.NET 数据源对象，例如，DataTable, DataView, DataSet 和 DataViewManager。还有第三方的数据源，例如 ComponentOne DataObjects 组件，包括 C1ExpressTable, C1ExpressVew, C1ExpressConnection, C1DataView, C1DataTableSource 和 C1DataSet 都可以被使用。

关于创建 ADO.NET 数据源的详细信息，请参照.NET Framework 文档。

使用 ComponentOne DataObjects for .NET 的详细信息参照 ComponentOne Studio for .NET 中的 ComponentOne DataObjects for .NET 文档。

下面章节介绍了怎样通过首先设定数据源，然后设定数据表单来给图表绑定数据。

10.1 给 C1Chart 绑定数据

给 C1Chart 绑定数据首先设定数据源，然后设定数据表单。

步骤 1：设定数据源

一旦创建了一个正确的数据源，就可以设定 C1Chart.DataSource 属性，这些可以在设计时做，也可以在运行时做。

在设计时设定 DataSource 属性

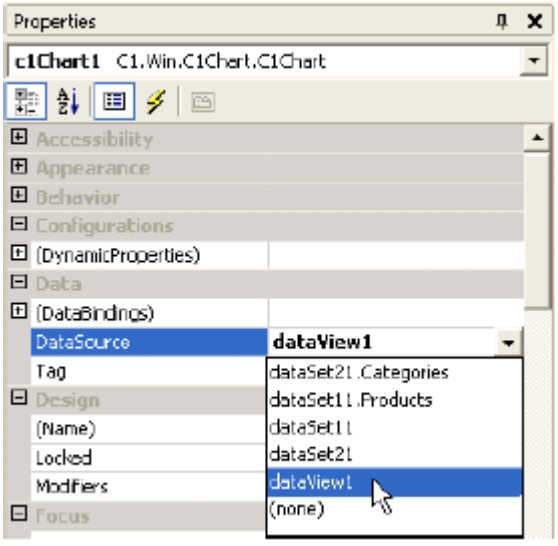
你可以使用下面的任何一种方法在设计时设置 C1Chart.DataSource 属性：

C1Chart 属性窗口

在 C1Chart 属性窗口设定 C1Chart.DataSource 属性：

a. 在属性下拉列表中选择 C1Chart

b. 然后从 C1Chart.DataSource 属性的下拉列表中选择一个值



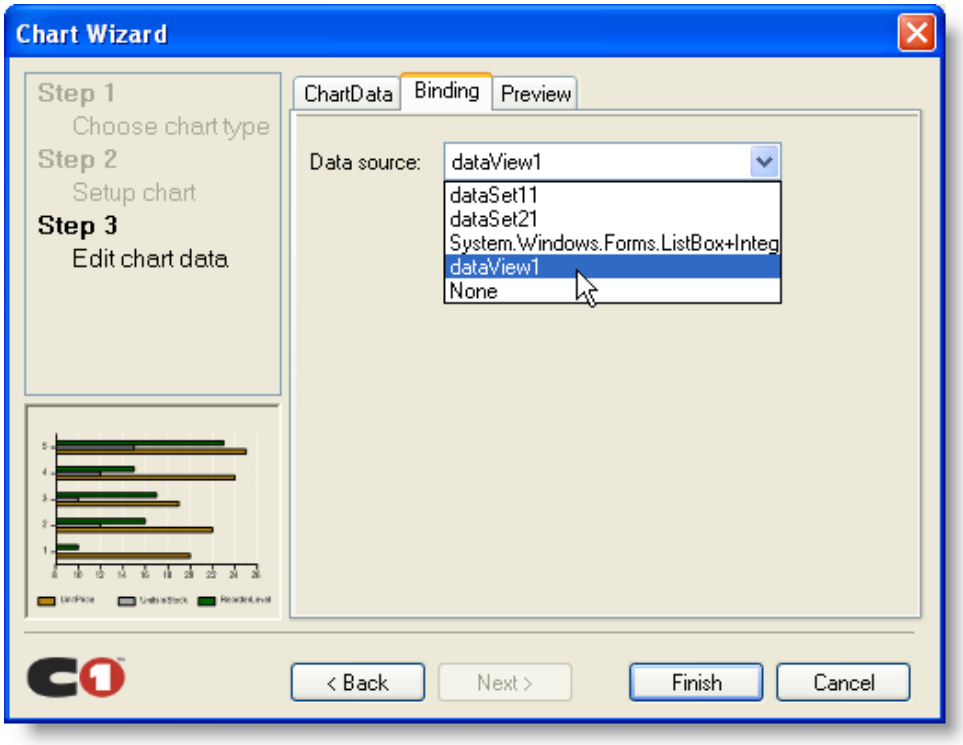
图表向导

C1Chart.DataSource 属性也可以使用图表向导或者图表属性设计器来设定。

使用图表向导来设定 C1Chart.DataSource 属性，要完成下面几步：

a. 在图表向导的步骤 3 中选择 Binding 选项卡。

b. 在 Data source 下拉列表中选择一个值。



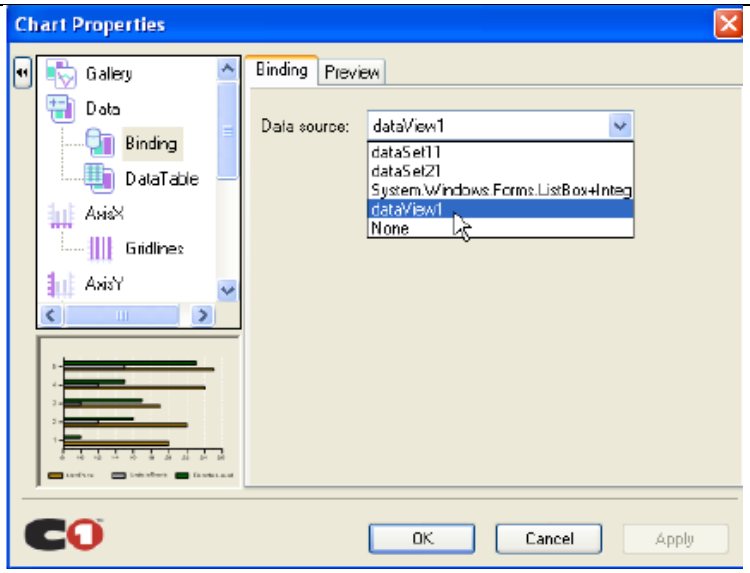
图表属性设计器

使用图表属性设计器设定 DataSource 属性，采用下面几步：

a. 右键点击 C1Chart 控件并选择 Chart Properties 来访问 Chart Properties 编辑器。

b. 展开 Data 元素，选择 Binding.

c. 在 Binding 选项卡中，从 Data source 下拉列表中选择一个值。



通过编程设定 DataSource 属性：

使用下面的代码将图表的 data source 属性设定为数据源对象：

- Visual Basic

```
C1Chart1.DataSource = DataSet11
```

- C#

```
c1Chart1.DataSource = dataSet11;
```

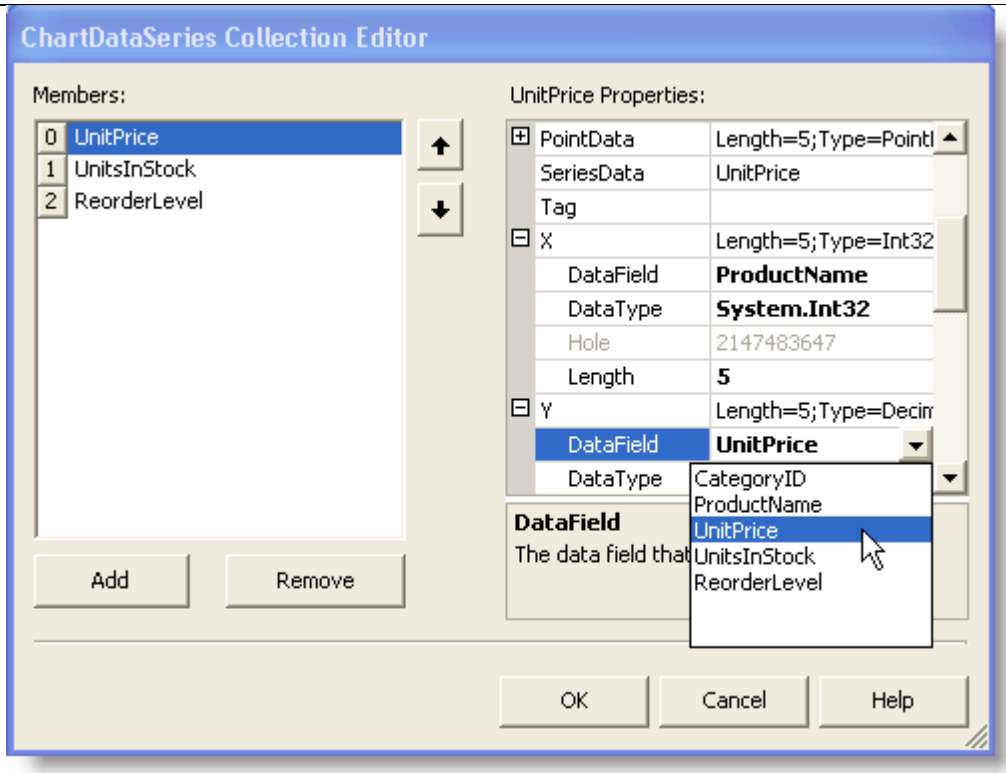
步骤 2：设定数据表单

一旦 C1Chart DataSource 已经被设定了，就必须告诉 **C1Chart** 每个 ChartDataSeries 对应数据表的哪一列。换句话说，必须定义数据源哪一个列要用来定义 X 值，哪一列定义 Y 值，并且根据 ChartType，每个 ChartDataSeries 的 **Y1，Y2，Y3** 对应哪一列。

X 值，Y 值等等，是通过每一个 ChartDataSeries 的 ChartDataArray 对象拿的。这些对象都有一个 DataField 属性，这个属性用来定义数据源中数据列，提供给特定的 ChartDataArray。DataField 属性可以在设计时设定，也可以在运行时设定。

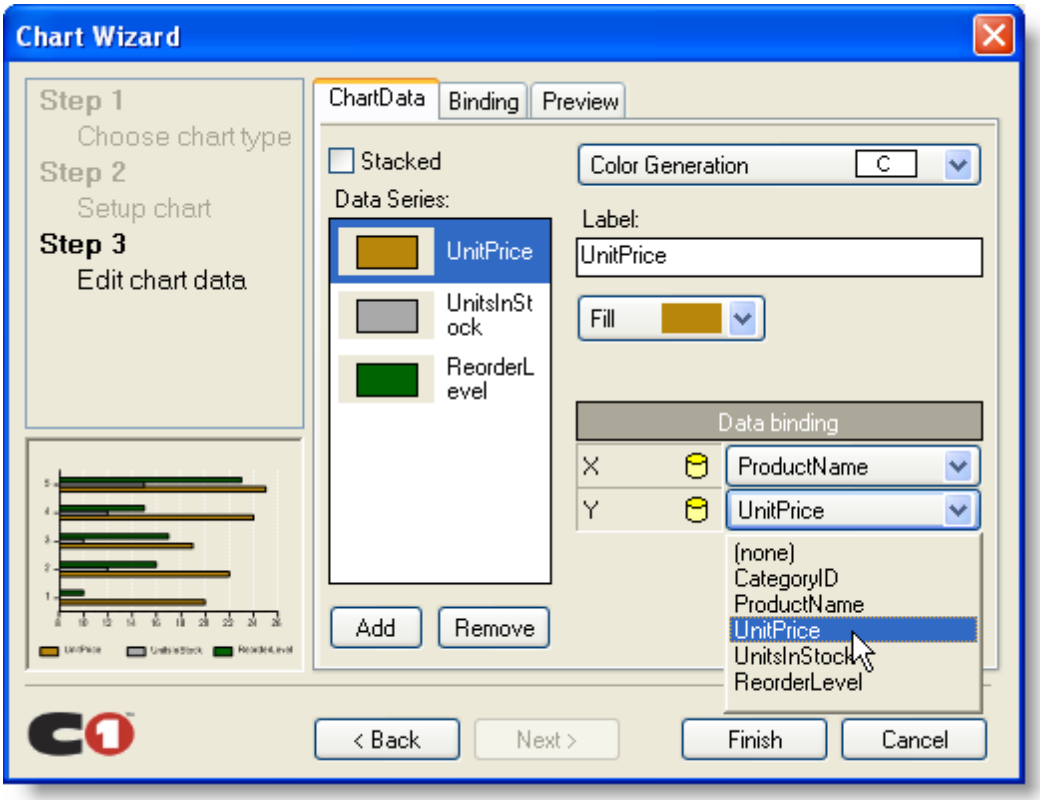
使用设计器设定数据表

在设计时使用 Visual Studio .NET 属性窗口，浏览 ChartGroups, Group0, ChartData, 和 SeriesList ,然后双击 **ellipsis** 按钮 ,打开 SeriesList。这样就打开了 **ChartDataSeries Collection Editor**。选择第一个系列 ,切换到 X 属性是一个 ChartDataArray 对象。展开 X 节点 ,选择 DataField 属性，然后从它的下拉列表中选择一个可用属性。



DataField 也可以通过图表向导和图表属性设计器来设定。下面的图描述了在图表向导中，X 下拉列表和 Y 下拉列表中的 DataField。

下面的图描述了在图表属性对话框中，X 下拉列表和 Y 下拉列表中的 DataField。



使用代码设定数据表格

运行时，每一个 ChartDataSeries 的每一个 ChartDataArray 对象的 DataField 属性都是可用的。注意 DataField 属性被设定为一个字符串的值，这个字符串的值标明了数据源列的名字。

- Visual Basic

```
C1Chart1.ChartGroups(0).ChartData.SeriesList(0).X.DataField = "QuickStop.ShippedDate"
C1Chart1.ChartGroups(0).ChartData.SeriesList(0).Y.DataField_ = "QuickStop.SaleAmount"
```

- C#

```
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].X.DataField = "QuickStop.ShippedDate";
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].Y.DataField = "QuickStop.SaleAmount";
```

请注意字符串的语法。在上面的编码例子中，QuickStop 是包含列的 TableName，ShippedDate 和 SaleAmount 是列的名字。TableName 是否要出现在字符串中由数据源自己所决定。在上面的例子中，一个由很多表的 DataSet 对象被作为了数据源使用，所以需要 TableName。如果在一个 DataSet 对象中，一个特定的表被用作为数据源，那么只需要使用列名。

10.2 直接绑定图表到数据源

在大多数的情况下，数据在绘制之前需要总结，所以在数据源和实际的图表之间有一个层。但是在一些情况下，你想要绘制的数据已经在一个 DataView 或者数据源对象中可用了，在这样的情况，你可以直接把图表绑定到数据源对象。

想要绑定一个 C1Chart 控件到一个数据源 ,你应该设定控件的 DataSource 属性为一个数据源对象，例如一个 DataView 或者 DataTable。然后绑定单独的系列到数据源对象的列，这时可以使用 DataSeries.X.DataField 和 DataSeries.Y.DataField 属性。

使用图表向导，下面的步骤都能做到，但是如果你喜欢用编码实现，下面有一个例子阐明了整个过程：

- Visual Basic

```
' DataBinding 只能在 1.0.20034.13244 及之后的版本中使用
' 获取 Chart 的数据
Dim sql As String = "select * from products"
Dim conn As String = "provider=... nwind.mdb" '
Dim da As New OleDbDataAdapter(sql, conn)
da.Fill(dt)
' 绑定数据源
c1chart.DataSource = dt
' 清空 Chart 中现有的全部系列
Dim dsc As ChartDataSeriesCollection = c1chart.ChartGroups(0).ChartData.SeriesList
dsc.Clear()
' 添加 Unit price 系列
Dim ds As ChartDataSeries = dsc.AddNewSeries()
'ds.AutoEnumerate = true' (本例中不需要设置 X 的 Values 属性)
ds.X.DataField = "ProductName"
ds.Y.DataField = "UnitPrice"
' 添加 Units in stock 系列
ds = dsc.AddNewSeries()
ds.X.DataField = "ProductName"
ds.Y.DataField = "UnitsInStock"
' 设置过滤排序等属性
dt.DefaultView.RowFilter = "CategoryID = 4"
```

- C#

```
// DataBinding 只能在 1.0.20034.13244 及之后的版本中使用
//
// 获取 Chart 的数据
string sql = "select * from products";
string conn = @"provider=... nwind.mdb;";
OleDbDataAdapter da = new OleDbDataAdapter(sql, conn);
da.Fill(dt);
//绑定数据源
c1chart.DataSource = dt;
// 清空 Chart 中现有的全部系列
ChartDataSeriesCollection dsc = c1chart.ChartGroups[0].ChartData.SeriesList;
dsc.Clear();

// 添加 Unit price 系列
ChartDataSeries ds = dsc.AddNewSeries();
//ds.AutoEnumerate = true; // (in case you don't want to set the X values)
ds.X.DataField = "ProductName";
ds.Y.DataField = "UnitPrice";
// 添加 Units in stock 系列
ds = dsc.AddNewSeries();
ds.X.DataField = "ProductName";
ds.Y.DataField = "UnitsInStock";
// 设置过滤排序等属性
dt.DefaultView.RowFilter = "CategoryID = 4";
```

上面的代码取回 NWind Products 表，然后创建了 2 个数据系列，每一个数据系列都绑定了数据源的一个列。

绑定表的最有趣的部分是它维护一个和数据源的动态联接。对数据源的任何改变包括值的变更，过滤，排序，新增或删除记录等等，都会自动的影响图表。

10.3 使用数据绑定函数化编程

在下面的代码当增加一个带 C1Chart 控件的新的工程，到一个窗口中，展示了一个完整的使用数据绑定的函数化程序。请注意创建和操控 **Form_Load** 控件也是十分必要的。这个例子中使用的数据库是 Microsoft 提供和使用的 NorthWind 数据库，也是 ComponentOne Studio for .NET 提供和使用的。如果文件的路径和你的安装路径不符，在执行程序之前，你必须修改数据库。

下面的例子完全使用编码操作数据库和 **C1Chart**。但是，使用 Microsoft .NET IDE 和它的

属性表，在设计时就能完全的完成代码所完成的操作。请注意使用了两个 DataAdapters ，DataAdapters 的 TableMapping 允许一个单独的 DataSet 作为 DataSource 使用，因此相同的表绘制了两个系列。

- Visual Basic

```
Private Sub BindMultipleSeriesViewsAndChartSetup(ByVal chart As C1.Win.C1Chart.C1Chart)
' 需要添加 System.Data.OleDb 命名空间
Dim connect As OleDbConnection = New OleDbConnection()
Dim adapt1 As OleDbDataAdapter = New OleDbDataAdapter()
Dim adapt2 As OleDbDataAdapter = New OleDbDataAdapter()
Dim select1 As OleDbCommand = New OleDbCommand()
Dim select2 As OleDbCommand = New OleDbCommand()
' 设置数据库连接字符串
connect.ConnectionString = _
"Provider=Microsoft.Jet.OLEDB.4.0;" + _
"User ID=Admin;" + _
"Data Source=C:\Program Files\ComponentOne Studio.Net\common\NWIND.MDB;" + _ "Jet OLEDB:Engine Type=5;"
' 设置查询 SQL 语句
select1.CommandText = _
"SELECT SaleAmount, ShippedDate, CompanyName " + _
"FROM [Sales Totals by Amount] " + _
"WHERE (CompanyName = 'Save-a-lot Markets') " + _
"ORDER BY ShippedDate"
select1.Connection = connect
'设置查询 SQL 语句
select2.CommandText = _
"SELECT SaleAmount, ShippedDate, CompanyName " + _
"FROM [Sales Totals by Amount] " + _
"WHERE (CompanyName = 'QUICK-Stop') " + _
"ORDER BY ShippedDate"
select2.Connection = connect
' 需要为 Mapping 对象添加 System.Data.Common 命名空间
adapt1.SelectCommand = select1
Dim ColumnMaps_SaveALot As DataColumnMapping() = _
{ _
```

```
New DataColumnMapping("SaleAmount", "SaleAmount"), _
New DataColumnMapping("CompanyName", "CompanyName"), _
New DataColumnMapping("ShippedDate", "ShippedDate") _
}
adapt1.TableMappings.Add(New DataTableMapping("Table", "SaveALot", _
ColumnMaps_SaveALot))
' 设置 adapter, adapt2.
adapt2.SelectCommand = select2
Dim ColumnMaps_QuickStop As DataColumnMapping() = _

{ _
New DataColumnMapping("SaleAmount", "SaleAmount"), _
New DataColumnMapping("CompanyName", "CompanyName"), _
New DataColumnMapping("ShippedDate", "ShippedDate") _
}
adapt2.TableMappings.Add(New DataTableMapping("Table", "QuickStop", _
ColumnMaps_QuickStop))
' 创建 DataSet 并填充数据
Dim ds As DataSet = New DataSet()
adapt1.Fill(ds)
adapt2.Fill(ds)
' 为 chart 设置 DataSource,DataFields 以及其他属性
chart.Dock = DockStyle.Fill
chart.DataSource = ds
Dim sc As ChartDataSeriesCollection = chart.ChartGroups(0).ChartData.SeriesList
sc.RemoveAll()
' 添加 Save-A-Lot 系列.
Dim s As ChartDataSeries = sc.AddNewSeries()
s.Label = "Save-A-Lot"
s.X.DataField = "SaveALot.ShippedDate"
s.Y.DataField = "SaveALot.SaleAmount"
```

```
        ' 添加 Quick-Stop 系列.
        s = sc.AddNewSeries()
        s.Label = "Quick-Stop"
        s.X.DataField = "QuickStop.ShippedDate"
        s.Y.DataField = "QuickStop.SaleAmount"
        ' 设置坐标轴和图例区
        chart.ChartArea.AxisX.AnnoFormat = FormatEnum.DateShort
        chart.ChartArea.AxisY.AnnoFormat = FormatEnum.NumericCurrency
        chart.ChartArea.AxisY.Min = 0
        ' 设置图表类型为 Bar
        chart.ChartGroups[0].ChartType = Chart2DTypeEnum.Bar
        chart.ChartGroups[0].ShowOutline = false
        ' 设置图例区的显示属性
        chart.Legend.Compass = CompassEnum.North
        chart.Legend.Orientation = LegendOrientationEnum.Horizontal
        chart.Legend.Visible = true
        End Sub
        Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) _ Handles
MyBase.Load
        C1Chart1.Location = New Point(0)
        C1Chart1.Size = Me.ClientSize
        BindMultipleSeriesViewsAndChartSetup(C1Chart1)
        End Sub
```

- C#

```
private void BindMultipleSeriesViewsAndChartSetup(C1.Win.C1Chart.C1Chart chart)
{
//需要添加 System.Data.OleDb 命名空间
OleDbConnection connect = new OleDbConnection();
OleDbDataAdapter adapt1 = new OleDbDataAdapter();
OleDbDataAdapter adapt2 = new OleDbDataAdapter();
OleDbCommand select1 = new OleDbCommand();
OleDbCommand select2 = new OleDbCommand();
//设置数据库连接字符串
connect.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;" +
"User ID=Admin;" +
"Data Source=C:\\Program Files\\ComponentOne Studio.Net\\common\\NWIND.MDB;" + "Jet
OLEDB:Engine Type=5;";
//设置查询 SQL 语句
select1.CommandText =
"SELECT SaleAmount, ShippedDate, CompanyName " +
"FROM [Sales Totals by Amount] " +
"WHERE (CompanyName = \'Save-a-lot Markets\') " +
"ORDER BY ShippedDate";
select1.Connection = connect;
//设置查询 SQL 语句
select2.CommandText =
"SELECT SaleAmount, ShippedDate, CompanyName " +
"FROM [Sales Totals by Amount] " +
"WHERE (CompanyName = \'QUICK-Stop\') " +
"ORDER BY ShippedDate";
select2.Connection = connect;
//需要为 Mapping 对象添加 System.Data.Common 命名空间
adapt1.SelectCommand = select1;
DataColumnMapping [] ColumnMaps_SaveALot =
{
new DataColumnMapping("SaleAmount", "SaleAmount"),
new DataColumnMapping("CompanyName", "CompanyName"),
```

```
        new DataColumnMapping("ShippedDate", "ShippedDate")
        };

        // 设置 adapter, adapt2.
        adapt2.SelectCommand = select2;
        DataColumnMapping [] ColumnMaps_QuickStop =
        {
        new DataColumnMapping("SaleAmount", "SaleAmount"),
        new DataColumnMapping("CompanyName", "CompanyName"),
        new DataColumnMapping("ShippedDate", "ShippedDate")
        };
        adapt2.TableMappings.Add(new DataTableMapping("Table", "QuickStop",
ColumnMaps_QuickStop));
        //创建 DataSet 并填充数据
        DataSet ds = new DataSet();
        adapt1.Fill(ds);
        adapt2.Fill(ds);
        //为 chart 设置 DataSource,DataFields 以及其他属性
        chart.Dock = DockStyle.Fill;
        chart.DataSource = ds;
        ChartDataSeriesCollection sc = chart.ChartGroups[0].ChartData.SeriesList;
        sc.RemoveAll();
        //添加 Save-A-Lot 系列.
        ChartDataSeries s = sc.AddNewSeries();
        s.Label = "Save-A-Lot";
        s.X.DataField = "SaveALot.ShippedDate";
        s.Y.DataField = "SaveALot.SaleAmount";
        // 添加 Quick-Stop 系列.
        s = sc.AddNewSeries();
        s.Label = "Quick-Stop";
        s.X.DataField = "QuickStop.ShippedDate";
        s.Y.DataField = "QuickStop.SaleAmount";
        //设置坐标轴和图例区
        chart.ChartArea.AxisX.AnnoFormat = FormatEnum.DateShort;
        chart.ChartArea.AxisY.AnnoFormat = FormatEnum.NumericCurrency;
        chart.ChartArea.AxisY.Min = 0;
        //设置图表类型为 Bar
        chart.ChartGroups[0].ChartType = Chart2DTypeEnum.Bar;
```

```
chart.ChartGroups[0].ShowOutline = false;
// 设置图例区的显示属性
chart.Legend.Compass = CompassEnum.North;
chart.Legend.Orientation = LegendOrientationEnum.Horizontal;
chart.Legend.Visible = true; chart.Legend.Visible = true;
}
private void Form1_Load(object sender, System.EventArgs e)
{
c1Chart1.Location = new Point(0);
c1Chart1.Size = this.ClientSize;
BindMultipleSeriesViewsAndChartSetup(c1Chart1);
}
```

11. 绘制标签

ChartLabels 是显示在图表数据前面的标签。图表不会调整自身大小来显示下 ChartLabels，因为 ChartLabels 是完全独立的。ChartLabels 在高亮一个重要数据的时候非常有用，也能够用来在数据上或者图表上显示信息。

一个 Label 对象定义了一个独立的长方形区域，可以被附到图表中去。LabelsCollection 包含给一个特定图表定义的所有的标签。一个图表的标签可以在设计时添加到图表中，也可以通过 ChartLabels 对象编程添加。C1Chart 提供一个 Edit Labels 设计器，所以你可以使用设计器很方便地添加，删除，或者修改已经存在的标签。

下面的代码说明了如何通过定义 LabelsCollection 的索引数访问单独的 ChartLabels，

- Visual Basic

```
C1Chart1.ChartLabels.LabelsCollection(0).Text = "First label in collection"
```

- C#

```
c1Chart1.ChartLabels.LabelsCollection[0].Text = "First label in collection";
```

下面的代码说明了如何通过使用 AddNewLabel 的方法创建一个 Lable 对象，

- Visual Basic

```
Dim Clabel As C1.Win.C1Chart.Label
Clabel = C1Chart1.ChartLabels.LabelsCollection.AddNewLabel() 199
```

- C#

```
C1.Win.C1Chart.Label Clabel;
Clabel = c1Chart1.ChartLabels.LabelsCollection.AddNewLabel();
```

一个图表包含的 ChartLabels 的个数没有限制。每个 ChartLabels 都可以自定义标签，内部颜色，边框和附加属性。

Label 对象提供许多的属性帮助定义和定位 ChartLabel。这些属性中比较重要的有：

- Compass 属性定义了标签围绕着锚的位置，例如，设定 Compass 属性为 East 将定位标签在锚的右边。
- AttachMethod 属性定义了使用哪种方法将标签附到图表中去。可以使用 Coordinate, DataCoordinate, DataIndex,或者使用 DataIndexY
- Text 属性定义了在 ChartLabel 中显示的文本。
- Connected 属性是布尔型，它定义了图表标签和添附位置之间是否需要画一条线，如果是 True，要画线。
- Offset 属性定义距离，是在锚的方向，从 ChartLabel 到添加位置的距离，类型是 Long
- DefaultLabelStyle 属性设定一个默认的风格，所有的标签都继承这个风格。例如如果默认风格的背景色被设定为了黑色，所有的要做的标签都会是黑色的背景色。

一个 Label 对象定义了一个独立的能够添附到图表中的长方形区域。LabelsCollection 包含给一个特定图表定义的所有的标签。

11.1 添附和定位图表的标签

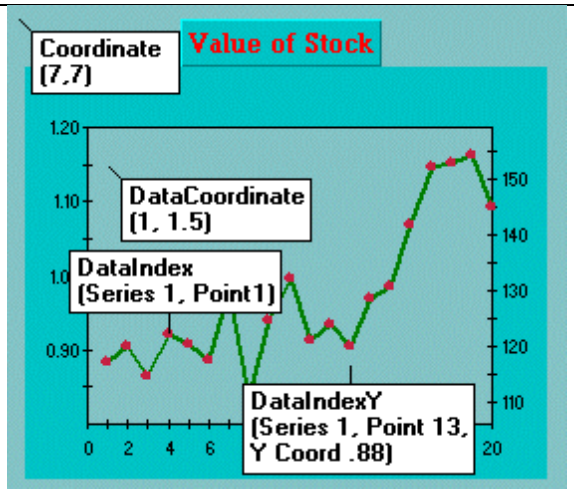
当定义一个 ChartLabel，必须定义如何添附到图表以及相对于它的添附位置的哪个点添附图表。

添附方法的选择的依据是 ChartLabel 的用途。添附方法是带有四个选项的 AttachMethodEnum 值：Coordinate, DataCoordinate, DataIndex, 和 DataIndexY.

ChartLabels 有四种方法添加到图表

- 添附到一个像素（x，y）坐标
- 添附到 ChartArea 的数据点（x，y）坐标
- 添附到 ChartArea 中的一个（系列，点）
- 添附到 ChartArea 中的一个（系列，点，Y 值）

下图表示出了 ChartLabel 的添附方法



下面列出并解释了 ChartLabels 的每个添附方法

- AttachMethodEnum.Coordinate 在图表的任何位置添加标签。定义从图表的左上角到 ChartLabels 的像素数。
- AttachMethodEnum.DataCoordinate 在 PlotArea 的任何位置添加标签，可以定义数据坐标。这个方法在 X 轴坐标上使用 PointLabels 的饼状图上不适用。如果 ChartLabel 的任何部分超出了 ChartArea，会被截掉。
- AttachMethodEnum.DataIndex 在图表的指定的数据点添附标签。要定义系列，点索引和
- ChartGroup 。
- AttachMethodEnum.DataIndexY 在指定数据点的上面或下面的一定距离添附标签。要定义系列，点索引，ChartGroup 和 Y 坐标。这个方法对条状图和堆积条形图非常有用。

使用 Label 类的 AttachMethod 属性来设定添附标签的方法，使用 AttachMethodData 的属性来设定添附的点。这些属性可以使用代码通过 Label 类来访问，也可以在设计时通过 Labels Collection 编辑器或者 Edit Labels 设计器访问。关于在设计时使用图表智能编辑器设定 ChartLabel 方法的详细信息，请参照添附和定位图表标签。

11.1.1 通过像素坐标添附图表标签

使用坐标给图表添附标签可以将标签添附到图表的任何位置。像素数定义的是图表左上角到 ChartLabel 的距离。

添附标签到坐标的一个像素，设定 AttachMethod 属性到 AttachMethodEnum.Coordinate，并且在 AttachMethodData 对象中设定 X 和 Y 属性为添附 ChartLabel 的坐标。例如，下面的代码添附了一个 ChartLabel 到指定的坐标。

- Visual Basic

```
With C1Chart1.ChartLabels.LabelsCollection(0)
.AttachMethod = AttachMethodEnum.Coordinate
.AttachMethodData.X = 250
.AttachMethodData.Y = 250
End With
```

- C#

```
C1.Win.C1Chart.Label lab = c1Chart1.ChartLabels.LabelsCollection[0];
lab.AttachMethod = AttachMethodEnum.Coordinate;
lab.AttachMethodData.X = 250;
lab.AttachMethodData.Y = 250;
```

11.1.2 通过数据坐标添附图表标签

在图表中添附一个 ChartLabel 到一个数据坐标，设定 AttachMethod 属性为 AttachMethodEnum.DataCoordinate，并且在 AttachMethodData 对象中设定添 X 和 Y 属性为添附 ChartLabel 的数据坐标。例如字码的代码添附一个 ChartLabel 到数据坐标(1.1, 15.8):

- Visual Basic

```
With C1Chart1.ChartLabels.LabelsCollection(0)
.AttachMethod = AttachMethodEnum.DataCoordinate
.AttachMethodData.X = 1.1
.AttachMethodData.Y = 15.8
End With
```

- C#

```
C1.Win.C1Chart.Label lab = c1Chart1.ChartLabels.LabelsCollection[0];
lab.AttachMethod = AttachMethodEnum.DataCoordinate;
lab.AttachMethodData.X = 1.1;
lab.AttachMethodData.Y = 15.8;
```

只有图表是 Area 或者 XY-Plot 的时候 ChartLabels 才能添附到一个数据坐标。如果 ChartLabel 的任何部分超出了 ChartArea，会被截掉。

11.1.3 使用数据点添附图表标签

添附一个 ChartLabel 到一个数据点，需要设定 AttachMethod 属性为 AttachMethodEnum.DataIndex，并且在 AttachMethodData 对象中设定 GroupIndex, SeriesIndex 和 PointIndex 属性。这些属性定义了 ChartGroup，系列和点索引。PointIndex 适用于数据组的元素，SeriesIndex 适用于系列组的元素。每个系列有许多组数据。

下面的代码添加了一个 ChartLabel 到指定的数据点。

- Visual Basic

```
With C1Chart1.ChartLabels.LabelsCollection(0)
.AttachMethod = AttachMethodEnum.DataIndex
.AttachMethodData.GroupIndex = 0
.AttachMethodData.SeriesIndex = 1
.AttachMethodData.PointIndex = 2
End With
```

- C#

```
C1.Win.C1Chart.Label lab = c1Chart1.ChartLabels.LabelsCollection[0];
lab.AttachMethod = AttachMethodEnum.DataIndex;
lab.AttachMethodData.GroupIndex = 0;
lab.AttachMethodData.SeriesIndex = 1;
lab.AttachMethodData.PointIndex = 2;
```

11.1.4 通过数据点和 Y 值添附图表标签

这两个方法，通过数据坐标的方法和通过数据点的方法添附 ChartLabel 可以混合使用，这样一个 ChartLabel 就可以添附在 X 坐标由数据点定义而 Y 坐标由一个图表值定义的位置。注意这个方法不适用于饼图。

要使用数据点和 Y 值添附 ChartLabel，设定 AttachMethod 属性为 AttachMethodEnum.DataIndexY，然后在 AttachMethodData 对象中设定 GroupIndex, SeriesIndex, PointIndex，和 Y 属性.

- Visual Basic

```
With C1Chart1.ChartLabels.LabelsCollection(0)
.AttachMethod = AttachMethodEnum.DataIndex
.AttachMethodData.GroupIndex = 0
.AttachMethodData.SeriesIndex = 1
.AttachMethodData.PointIndex = 2
.AttachMethodData.Y = 15.8
End With
```

- C#

```
C1.Win.C1Chart.Label lab = c1Chart1.ChartLabels.LabelsCollection[0];
lab.AttachMethod = AttachMethodEnum.DataIndex;
lab.AttachMethodData.GroupIndex = 0;
lab.AttachMethodData.SeriesIndex = 1;
lab.AttachMethodData.PointIndex = 2;
lab.AttachMethodData.Y = 15.8;
```

在这个例子中，ChartLabel 的位置是这样定义的：

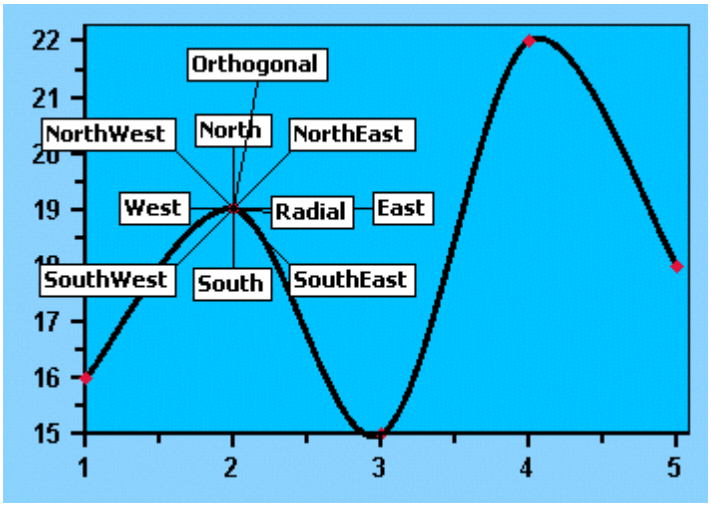
第一个数据系列的第二点的 X 坐标作为图表标签的 X 坐标

Y 属性设定的 Y 坐标作为图表标签的 Y 坐标。

注意：这个方法不适用于饼图。

11.2 固定图表标签

一旦添附到图表中的一个点，设定 Compass 属性来设定 ChartLabel 的固定位置。这个属性有一个枚举类型的 LabelCompassEnum，下图表示出来一些正确的固定位置。



ChartLabels 中 Compass 属性的用法和相位和.NET 中其他对象的行为一样。根据定义的罗盘的方向 Label 被放置到距离点的相对位置。LabelCompassEnum.Radial 设定一般只适用于极线图，雷达图和饼图。它从图表的中间放射状地放置标签。这对一个有很多的标签要显示的图表，例如饼图，是非常方便地功能。LabelCompassEnum.Orthogonal 设定在一个系列线的垂直位置放置标签，例如如果有一个直的水平线作为系列的线，那么 Orthogonal 的位置就和 North Compass 一样。同样有一个直的垂直的线作为系列线，那么 Orthogonal 的位置就和 West Compass 一样。

11.3 定制图表标签

图表标签旋转

图表标签旋转

使用 C1.Win.C1Chart.Style.Rotation 属性可以设定和修改一个 ChartLabel 的旋转。旋转可以在设计时设定 ,使用 ChartLabels 对象的 DefaultLabelStyle 节点或者通过 the Label Collection Editor 设定。关于 Label Collection Editor 的更多信息，参见 Label Collection Editor.

C1.Win.C1Chart.Style.Rotation 可以被设定为 Rotate0, Rotate90, Rotate180, 或者 Rotate270，如果你希望设定非 90,180,270 度的特殊的角度，你可以覆盖这些值，定义围绕连接的标签点顺时针旋转的角度。

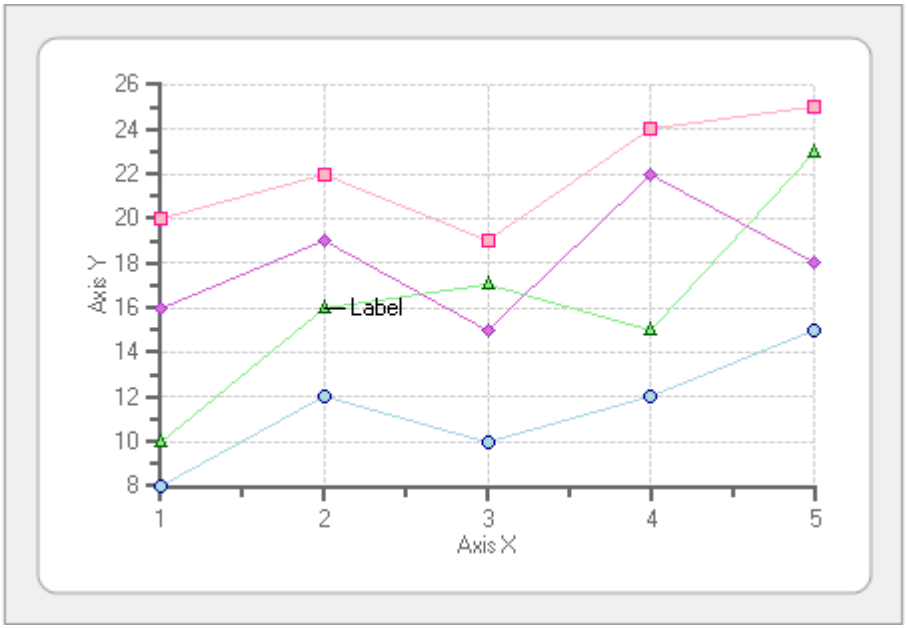
Label.RotationOverride 属性对需要特殊标志的图表非常有用，例如有很多轴的带有不同参数的连续的线像热含量，湿度，温度等等。

注意： Label.RotationOverride 不适用于 Radial 或者 RadialText 的 compass 值。

图表标签的连接线

设定 Label.Connected 为 True 就能够在数据坐标和相关的标签之间显示一条连接线。需要给 Label.Offset 属性设定一个数字的值用来给在数据坐标和标签之间画连接线腾出空间。

下面的图显示出了１０个像素偏移量的连接线。



图表标签边框

使用 BorderStyle 和 Thickness 边框属性可以定制题目的 ChartLabel。这些属性可以在设计时设定，通过使用 ChartLabels Collection Editor 的 Style 节点。关于 ChartLabels Collection Editor 的更多信息参见 Label Collection Editor。这个属性也能够在 DefaultLabelStyle 节点设定。关于如何客户化标签外观的更详细信息参见使用属性窗口修改图表标签的外观。

图表标签颜色

使用.NET Color 属性设定 ChartLabels 的颜色。 颜色属性可以在设计时设定，通过使用 ChartLabels Collection Editor 的 Style 节点。关于 ChartLabels Collection Editor 的更多信息参见 Label Collection Editor。更多信息参见图表颜色。这个属性也能够在 DefaultLabelStyle 节点设定。

图表标签字体

使用字体属性客户化 ChartLabel 的字体。这些属性可以在设计时设定，通过使用 ChartLabels Collection Editor 的 Style 节点。关于 ChartLabels Collection Editor 的更多信息参见 Label Collection Editor。这个属性也能够在 DefaultLabelStyle 节点设定。

11.4 设定默认的标签风格

ChartLabels 对象的 DefaultLabelStyle 给所有现有的和将要有的 ChartLabels 设定默认的风格。这个属性在很多方面都很有用，下面列出两个优点。

首先，将要有的 Labels 都继承这个属性设定的默认风格。第二，由于所有的标签都继承这

个默认风格，如果默认的风格属性变更了，所有的 ChartLabels 都会按新的属性呈现。例如，假设在设计时创建了１００个 ChartLabels 都有绿色的背景色，下面的代码需要添加到应用中。

- Visual Basic

```
C1Chart1.ChartLabels.DefaultLabelStyle.BackColor = Color.Green
```

- C#

```
c1Chart1.ChartLabels.DefaultLabelStyle.BackColor = Color.Green;
```

如果不得不修改单独 ChartLabels 的风格属性，那么这些属性可以在指定的 ChartLabel 对象上手工设定。

12. Chart Area 和 Plot Area 对象

这个章节详细地介绍 ChartArea 对象，并且对如何在 ChartArea 对象中访问，修改和创建属性外观提供了信息．

下表定义了包含在 **ChartArea 对象中的主要对象**．

属性	描述
Alignment	获得或设定轴显示时文字的对齐方式，从 ChartArea 中继承
Compass	允许你设定轴的位置．例如，你也许希望在数据上面显示Ｘ轴而不是显示在下面
C1.Win.C1Chart.Style.Font	设定轴上显示值所用的字体
ForeColor	设定显示轴，刻度标记和值的前景色
OnTop	取得或设定是否轴和格线在图表图的上面显示.
Reversed	取得或设定轴是通常的或者反转的（升序或者降序).
Text	设定一个字符串显示在轴的旁边 (这是描述变量和描述轴代表的单位的典型的用法).
Rotation	设定Ｔｅｘｔ字符串的方向

12.1 轴

ChartArea 对象的Ｘ，Ｙ和Ｙ２属性返回 Axis 对象，这样你可以定制图表轴的外观．ChartArea 属性的二级属性代表了轴：ChartArea.AxisX, ChartArea.AxisY，和 ChartArea.AxisY2．这些属性中的每一个都返回一个 Axis 对象，Axis 对象有下面一些主要的属性：

属性	描述
Alignment	获得或设定轴显示时文字的对齐方式，从 ChartArea 中继承

Compass	允许你设定轴的位置．例如，你也许希望在数据上面显示Ｘ轴而不是显示在下面
C1.Win.C1Chart.Style.Font	设定轴上显示值所用的字体
ForeColor	设定显示轴，刻度标记和值的前景色
OnTop	取得或设定是否轴和格线在图表图的上面显示.
Reversed	取得或设定轴是通常的或者反转的（升序或者降序).
Text	设定一个字符串显示在轴的旁边 (这是描述变量和描述轴代表的单位的典型的用法).
Rotation	设定Ｔｅｘｔ字符串的旋转

12.1.1 轴的位置

轴的注释一般来说显示在轴的旁边 .这对图表的起点不在轴的最小值或者最大值的图表会有一些问题．图表可以自动决定在不同的情况下根据图表的类型在哪里放置注释．Compass 属性也可以定义一个轴注释的位置．X 轴的 Compass 值可以设定为 North 或者 South，Y 轴可以设定为 East 或者 West．默认情况下，X 轴设定为 **South** 并且 Y 轴设定为 **West．**

12.1.2 轴的外观

Alignment 对齐方式

Alignment 属性有三种设定：Center, Near, 或者 Far。 设定对齐方式为中央，轴的题目在相对于 ChartArea 的中央位置。设定对齐方式为 Near，就会放置轴的题目在 ChartArea 的左边位置。设定对齐方式为 Far，就会放置轴的题目到 ChartArea 的右边。

Font 字体

字体的大小和类型都可以通过操作轴的 Font 对象来改变。要在设计时访问 Font 属性，需要点击 Font 节点旁边的省略号或者在 Visual Studio 属性窗口展开轴对象的 Font 节点。想要通过编程改变轴的 Font 属性，输入下面的代码。

- Visual Basic

```
Dim f As Font = New Font("Arial", 8, FontStyle.Bold)
C1Chart1.ChartArea.AxisX.Font = f
```

- C#

```
Font f = new Font("Arial", 8, FontStyle.Bold);
c1Chart1.ChartArea.AxisX.Font = f;
```

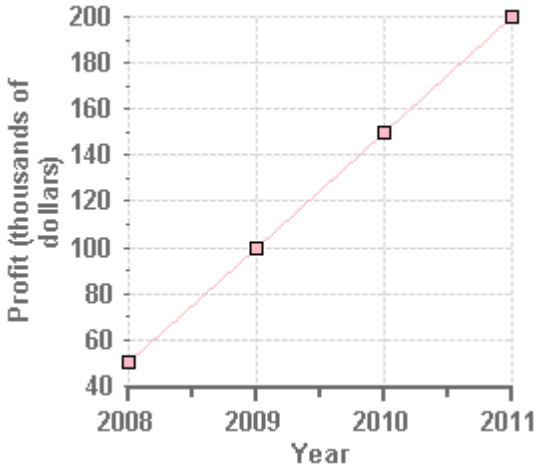
ForeColor 前景色

ForeColor 属性改变轴线，刻度标记和题目的颜色。改变 ForeColor，需要使用代码将 ForeColor 属性设定为一个正确的颜色 ,或者在设计时在轴对象的 Visual Studio 属性窗口也能访问到这个属性。

Thickness 厚度

轴的厚度由 Thickness 属性定义。这个值是用像素定义的。一个零厚度的轴也是完全可以画出来的就是没有轴线了。

下面的图表显示了 Y 轴的 Thickness 属性设定为 0 的效果。



12.1.3 轴标题和旋转

给一个轴添加标题能够阐明沿着轴画的是什么。轴的题目可以添加到 **Area, XY-Plot, Bar, HiLo, HiLoOpenClose** 或者 **Candle** 图表。标题或者注释也可以被旋转。

关于如何旋转轴注释的详细信息，参加轴注释的旋转（226 页）。

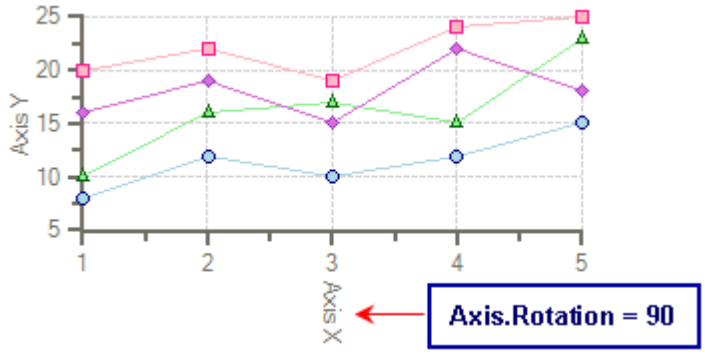
添加一个轴题目

使用轴的 Text 属性给一个轴添加题目。要删除题目，删除 AxisX 或者 AxisY 对象的默认文字。

注意：Y 轴的轴题目不能添加到极线图，雷达图或者填充雷达图

旋转轴题目

使用 Rotation 属性可以旋转轴的题目到 90,180 和 270 度。对应纵向的轴来说 ,90 度和 270 度的旋转是最有效的。



12.1.4 轴刻度标记

图表建立轴时就自动带有主要和副的刻度 ,定制刻度的间隔或者特性就如同设定一套属性那么容易。

TickMajor 和 TickMinor 属性设定轴刻度标记的状态。这个属性可以设定为

TickMarksEnum 的任何一个值。

刻度标记的位置

这些值设定哪里或者是否刻度标签显示

值	描述
TickMarksEnum.None	轴上没有刻度标记
TickMarksEnum.Cross	刻度标记和轴交叉
TickMarksEnum.Outside	刻度标记位于轴线上图表区域 的外面
TickMarksEnum.Inside	刻度标记位于轴线上图表区域的里面.

刻度标记的间隔

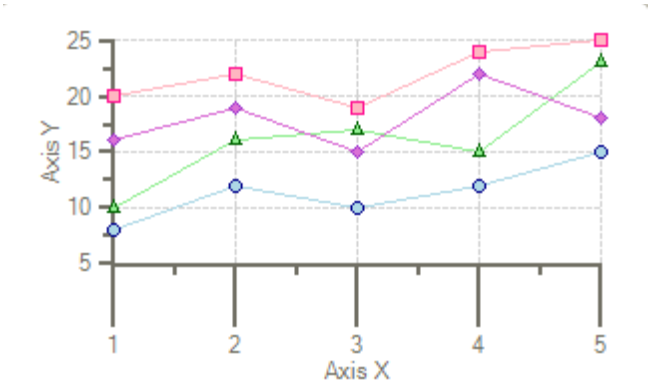
AutoMajor 和 AutoMinor 属性设定刻度标记是否由图表自动创建。当这两个属性都设定为 True，图表根据现在的数据逻辑地放置主要的和副的刻度标记。当 AutoMajor 是真，那就不必要使用轴注释了，容易重叠。

UnitMajor 和 UnitMinor 属性设定了刻度间隔的单位。当设定了 UnitMajor 属性，UnitMinor 属性自动被图表设定为 UnitMajor 一半的值。虽然图表自动设定 UnitMinor 属性，也是能够手动地改变为不同的值。

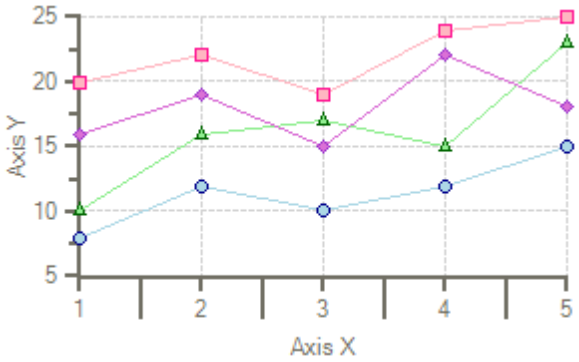
刻度标记的长度

通过使用 TickFactorMajor 和 TickFactorMinor 属性，你可以增加主要的和副的刻度标记的长度。刻度标记的大小需要根据轴线的厚度以及刻度因子。双倍的刻度因子就需要双倍的轴刻度标记的长度。值被设定为数字 1 到 20，超过这个范围的值会被忽略。

下面的图表的图将 TickFactorMajor 属性设定为 5.这就使主要刻度标记的刻度的长度是 5 倍的长度。



下面的图表的图将 TickFactorMinor 属性设定为 5.这就使副刻度标记的刻度的长度是 5 倍的长度。



12.1.5 轴格线

格线是指沿着主的 / 副的刻度标记垂直交叉的间隔主的 / 副的刻度宽度的线 .垂直于轴沿主间隔的线的显示 GridMajor 属性控制，垂直于轴沿副间隔的线的显示 GridMinor 属性控制．格线能够帮助提高图表的可读性，方便你看准确的值．主的和副的格线的外观由 ChartGridStyle 属性控制．

设定主格线的外观属性

使用 GridMajor 属性设定主格线的外观属性．GridMajor 属性取得 C1.Win.C1Chart.ChartGridStyle 对象，该对象控制主要格线的外观．

设定副格线的外观属性

使用 GridMajor 属性设定副格线的外观属性．GridMajor 属性取得 C1.Win.C1Chart.ChartGridStyle 对象，该对象控制副格线的外观．

OnTop

OnTop 属性定义是否轴的格线和刻度标记覆盖在图表的图之上或者在图表图的后面．设定这个属性为 True 会将格线和刻度标记放置图表图的前景的位置，设定为 False 将图表图放在前景．

12.1.6 轴边界

通常一个图显示它包含的所有数据．但是图表的一个特定部分能够显示固定的轴边界．

图表根据数据的最低和最高值以及编号增量定义每个轴的长度．设定 Min, Max, AutoMin, 和 AutoMax 属性可以定制这个定义轴长度的进程．

轴的最小和最大值

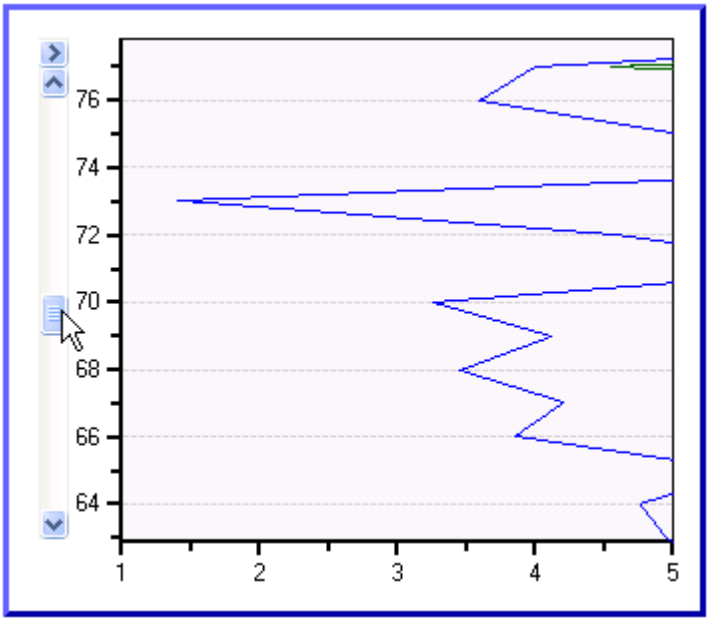
使用 Min 和 Max 属性定义轴的值能限制图表大小．如果图表的 X 轴的值从 0 到 １００，那么设定 Min 为 0 和 Max 为 10 只能显示最大到 10 的值．

图表也能够自动计算 Min 和 Max 值．如果 AutoMax 和 AutoMin 属性设定为 True，那么图表自动设定轴的编号方式以适应数据表．

注意：极线图不能设定 X 轴．

12.1.7 轴滚动和缩放

当你的图表数据中有相当数量的 X 值和 Y 值的情况下，你可以给轴添加滚动条．添加滚动条可以是图表的数据易读因为滚动滚动条你可以同时看到不同块的数据．下图展示了 AxisY 对象有 AxisScrollBar：



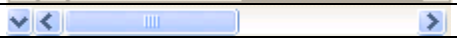
只需简单地将 ScrollBar 属性设定为 AxisX 或者 AxisY 对象就能设定 X 轴或者 Y 轴是否显示滚动条，然后设定图表数据系列的最大和最小值．设定最大和最小值可以避免当你滚动滚动条的时候改变轴的值．

12.1.7.1 滚动条的外观

C1Chart's AxisScrollbar 类提供了一些有用的属性来控制滚动条的外观，例如它的类型，大小和对齐方式．

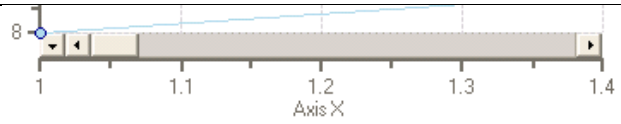
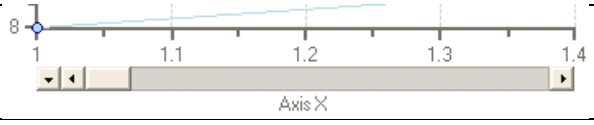
C1Chart 的滚动条外观能使用 Appearance 属性定义．

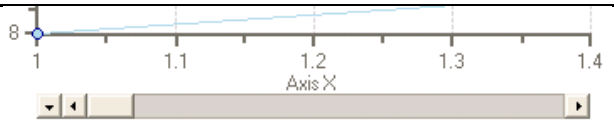
下面的表说明了你设定 Appearance 属性时可选择的不同的轴滚动条类型

Normal	
Flat	
XP	

滚动条的宽度可以用 Size 属性设定而对齐方式可以使用它的 Alignment 属性设定．

下表显示出了你设定 Alignment 属性时可选择的三种不同的对齐方式．

Near	
Center	

Far	

12.1.7.2 滚动条的缩放

当你使用滚动条在一大堆数据中滚动时，你可以使用缩放来放大重要的数据．

缩放可以在运行时通过点击滚动条上的按钮实现，或者通过代码，使用 ScaleMenuItems 属性．

当 Buttons 属性设定为 ScaleAndScrollButtons(默认)一个内置的缩放菜单就能显示出来．

下面的表是 Buttons 属性值的描述

值	描述
NoButtons	在滚动条上没有按钮显示
ScrollButtons	在滚动条上显示滚动按钮
ScaleButton	一个缩放的下拉按钮显示．当点击这个下拉按钮，一个内置的缩放菜单就会显示出来，显示的是默认的缩放项．
ScaleAndScrollButtons	默认选项．滚动按钮和缩放按钮都显示在滚动条上

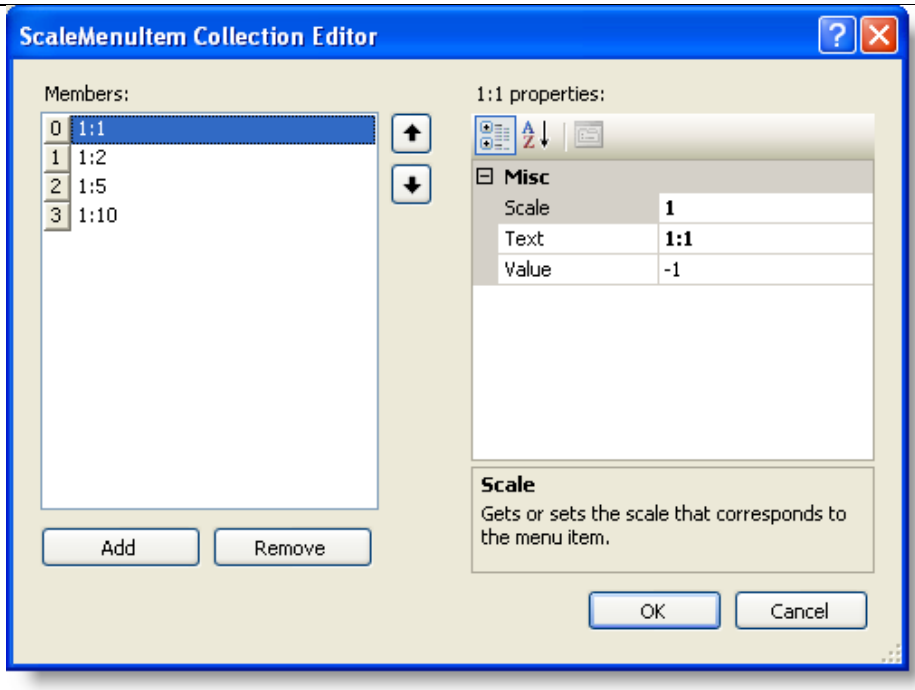
如果你想用自制的菜单代替缩放菜单，你可以创建该菜单然后把这个菜单分配到 ScaleMenu 属性．

ScaleMenuItem 集合的每个单独的项都可以在设计时使用 ScaleMenuItem 集合编辑器访问，或者使用 ScaleMenuItems 属性连接新的集合作为轴滚动条的缩放菜单的项目．

使用设计器

在设计时添加缩放项目到缩放菜单，需要完成下列步骤．

1. 右键点击 C1Chart 控件，从右键菜单中选择属性
2. 在属性窗口，展开 ChartArea 节点，展开 AxisX, AxisY,或者 AxisY2 节点，然后展开滚动条．
3. 点击 ScaleMenuItems 属性旁边的省略按钮 ，ScaleMenuItem 集合编辑器就出现了．



4. 点击新增（Ａｄｄ）给集合新增一个项目，然后设定它的 Scale 和 Text 属性为想要的值．
5. 结束时点击ＯＫ按钮．

使用代码

要增加一个定制的菜单项到缩放菜单 ,你可以清空已经存在的集合然后给集合天机缩放项目，如下所示．

- Visual Basic

```
With Me.C1Chart1.ChartArea.AxisX.ScrollBar
.ScaleMenuItems.Clear()
.ScaleMenuItems.Add(0.1, "1:10")
.ScaleMenuItems.Add(0.2, "2:10")
.ScaleMenuItems.Add(0.3, "3:10")
.ScaleMenuItems.Add(0.4, "4:10")
.ScaleMenuItems.Add(0.5, "5:10")
.ScaleMenuItems.Add(0.6, "6:10")
.ScaleMenuItems.Add(0.7, "7:10")
.ScaleMenuItems.Add(0.8, "8:10")
.ScaleMenuItems.Add(0.9, "9:10")
.ScaleMenuItems.Add(1.0, "10:10")
.Scale = 0.1       'initial value of scale
.Visible = True
End With
```

- C#

```
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Clear();
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.1, "1:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.2, "2:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.3, "3:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.4, "4:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.5, "5:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.6, "6:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.7, "7:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.8, "8:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(0.9, "9:10");
c1Chart1.ChartArea.AxisX.ScrollBar.ScaleMenuItems.Add(1.0, "10:10");
c1Chart1.ChartArea.AxisX.ScrollBar.Scale = 0.1 ;            //initial value of scale
c1Chart1.ChartArea.AxisX.ScrollBar.Visible = true;
```

上面的代码示例中清空已经存在的集合并且给 x 轴的滚动条添加了新的缩放菜单项． ScaleMenuItems 属性用来给图表的轴上的滚动条连接新的集合． Add 方法的第一个参数是缩放，必须是 0 和 1 之间（包含 0 和 1）的ｄｏｕｂｌｅ值．第二个参数，文本，是显示在菜单上的文字．

12.1.7.3 轴滚动的事件

当你需要给轴滚动定义更多的信息，C1Chart 和 AxisScrollBar 对象有 AxisScroll 事件提供．例如，使用 AxisScroll 事件你能够规定关于哪个轴被滚动的详细信息，在数据中滚动之前或之后最小值和最大值是什么的详细信息，滚动时表现为哪种类型以及滚动条的方向等详细信息．

AxisScroll 事件的发送对象可以是 AxisScrollBar 对象或者 c1Chart1 对象．如果 AxisScrollBar 对象和 c1Chart1 对象事件都被设定了，AxisScrollBar 事件先触发．但是，所有的轴都触发 AxisScroll 事件，设定 AxisScrollEventArgs 对象的 AxisId 属性来表示改变的轴．

AxisScrollEventArgs 类给 AxisScrollEvent 提供数据，AxisScrollEvent 在内置轴滚动条改变值的任何时候都触发事件．

下表表示出了 AxisScrollEventArgs 属性的信息

属性	描述
AxisID	表示哪个轴滚动的枚举值.
NewValue	取得和设定事件结束时轴滚动条的值．轴滚动条的值表示出了最小值和最大值之间的部分
OldValue	取得事件开始时轴滚动条的值．轴滚动条的值表示出了最小值和最大值之间的部分

ScrollEventType	取得象征滚动事件类型的值，例如 ThumbPosition, ThumbTrack, EndScroll, LargeIncrement, SmallIncrement, 等等.
ScrollOrientation	取得象征轴滚动条方向的值

下面的例子中显示了如何使用 AxisScroll 事件引用 AxisScrollEventArgs 类所有的属性．

- Visual Basic

```
Imports C1.Win.C1Chart
Public Sub New()
InitializeComponent()
c1Chart1.Dock = DockStyle.Fill

Dim scrollbar As C1.Win.C1Chart.AxisScrollBar =
c1Chart1.ChartArea.AxisX.ScrollBar
scrollbar.Scale  =          0.1

scrollbar.Visible           =          True
AddHandler     scrollbar.AxisScroll,  AddressOf          XAxis_ScrollEvent

AddHandler c1Chart1.AxisScroll, AddressOf XAxis_ScrollEvent
End Sub
Public Sub XAxis_ScrollEvent(ByVal sender As Object, ByVal e As
C1.Win.C1Chart.AxisScrollEventArgs) Handles C1Chart1.AxisScroll

Dim sb As New StringBuilder()
sb.AppendLine("" & Chr(10) & "AxisScroll Event Data")
sb.AppendLine(" Sender is: " + sender.ToString())
sb.AppendLine(" ID: " + e.AxisId.ToString())

sb.AppendLine(" OldVal: " + e.OldValue.ToString())
sb.AppendLine(" NewVal: " + e.NewValue.ToString())

sb.AppendLine(" EventType: " + e.ScrollEventType.ToString())
sb.AppendLine(" Orientation: " + e.ScrollOrientation.ToString())
System.Diagnostics.Debug.WriteLine(sb.ToString())
End Sub
```

- C#

```
using C1.Win.C1Chart;
public Form1()
{
InitializeComponent();
c1Chart1.Dock = DockStyle.Fill;

C1.Win.C1Chart.AxisScrollBar scrollbar =
c1Chart1.ChartArea.AxisX.ScrollBar;
scrollbar.Scale = 0.1; // set the scale to 1/10th
of the full axis width.
scrollbar.Visible = true; // make the scrollbar
visible
scrollbar.AxisScroll += new
C1.Win.C1Chart.AxisScrollEventHandler(XAxis_ScrollEvent);
c1Chart1.AxisScroll += new
C1.Win.C1Chart.AxisScrollEventHandler(XAxis_ScrollEvent);
}
public void XAxis_ScrollEvent(object sender,
C1.Win.C1Chart.AxisScrollEventArgs e)
{
StringBuilder sb = new StringBuilder();
sb.AppendLine("\nAxisScroll Event Data");
sb.AppendLine(" Sender is: " + sender.ToString());
sb.AppendLine(" ID: " + e.AxisId.ToString());

sb.AppendLine(" OldVal: " + e.OldValue.ToString());
sb.AppendLine(" NewVal: " + e.NewValue.ToString());

sb.AppendLine(" EventType: " +
e.ScrollEventType.ToString());
sb.AppendLine(" Orientation: " +
e.ScrollOrientation.ToString());
System.Diagnostics.Debug.WriteLine(sb.ToString());
}
```

代码输出 AxisScroll 事件数据如下．

```
AxisScroll Event Data
 Sender is: C1.Win.C1Chart.C1Chart
 ID: X
 OldVal: 0.251141552511416
 NewVal: 0.253678335870117
 EventType: ThumbTrack
 Orientation: HorizontalScroll


AxisScroll Event Data
 Sender is: C1.Win.C1Chart.AxisScrollBar
 ID: X
 OldVal: 0.253678335870117
 NewVal: 0.256215119228818
 EventType: ThumbTrack
 Orientation: HorizontalScroll


AxisScroll Event Data
 Sender is: C1.Win.C1Chart.C1Chart
 ID: X
 OldVal: 0.253678335870117
 NewVal: 0.256215119228818
 EventType: ThumbTrack
 Orientation: HorizontalScroll
```

12.1.8 轴对数比例

当显示有很大规模差异的数据或者希望在同样表格上的数据有指数的变化，通常在一个或多个轴上使用对数比例就能非常方便地实现．在一个对数的轴上，同等距离意味着同等百分数的变化．每个轴都可以设定为指数比例通过设定 IsLogarithmic 属性为 True．如果对数比例运用于一个或两个轴，图表就被称为 Log Scale（对数比例）图表．

使用对数比例，值被根据值的对数物理性地分隔而不是根据值本身．这对给很大范围的大数据量的数据做图表，以及希望描述几何和／或者指数的关系的图表时非常有用的．

不像算术图表改变是以直接的单位为单位计算的，log scale（对数比例）图表以百分数的变化为单位显示的．例如，在一个对数比例图表测量美元，从$1 到 $2 的变化是一个百分率１００的变化因此图表轴上$1 到 $2 的距离和$50 到$100 的距离相同．然而在一个算术的图表中，$50 到$100 的变化从轴上$50 移到$100 的距离会大得多因为$50 的变化和$1 变化是完全不同的．

通常使用的对数

对数可以使用任何基本值表达，包括数字和浮点值．两个最常使用对数的类型包括：

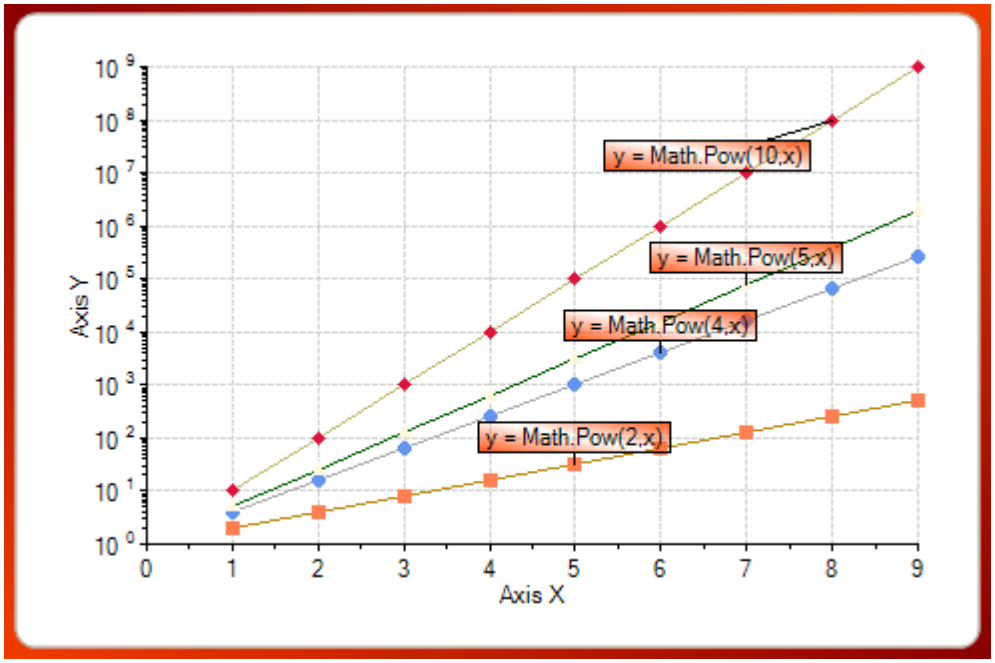
- 通常对数 - 以１０为基准写为对数１００＝２
- 自然对数 - 使用数学常数 e 为基准

对数底

当 IsLogarithmic 数学设定为真，你可以使用 LogarithmicBase 属性定义对数底数．默认值是１０．如果希望自然对数比例，设定 LogarithmicBase 为小于或等于１的值．如果值是１那么

1 的自然对数是 0 ，因为 e0 = 1. . 一个自然对数是底数是 e 的对数．注意对数比例当值等于或小于零的时候是没有数学意义的．因此当轴的 IsLogarithmic 属性设定为真，负值和零值是不画到轴上的．

下图表示出了 LogarithmicBase 设定为默认值１０也就是通常的对数时图表是如何表示的



可用示例

关于如何在 C1Chart 中使用对数比例的完整示例，参照实例 LogPlots， 位置在 http://helpcentral.componentone.com/Samples.aspx.

12.1.8.1 对数比例对轴上注释的输出

如果有下面的情况下，任何 C1Chart 的对数底都需要做格式化工作：

- AnnoFormat 属性= NumericManual
- AnnoFormatString 属性包含"**"
- 如果以上两个都是真，那么会输出字符串，这种对数底的对数输出为{0}.
- 如果存在下面的值
- LogarithmicBase = 10,
- AnnoFormat = NumericManual,
- AnnoFormatString = "10**{0}"

那么轴上的注释表示为 power10，“power” 是上标．

12.1.8.2 对数比例使用的准则

下面的补充准则是对数轴必须遵守的：

- 任何小于或者等于零的数据是不会画的（该数据当做数据漏洞处理），这是由于对数轴只处理大于 0 的值．也是因为同样的原因，轴和数据的最大 / 最小边界以及起

点属性不能设为零或者比零小。

- 轴的编号增量，标记增量和精度属性等轴是对数的时候是无效的
- 对于对数的 X 轴，图表类型必须是 plot, bubble, area, HiLo, HiLoOpenClose 或者 candle. 对于 Y 轴，图表的类型必须是 plot, bubble, area, polar, HiLo, HiLoOpenClose, candle, radar 或者 illed radar.
- X 轴的注释方法不能是 TimeLabels.

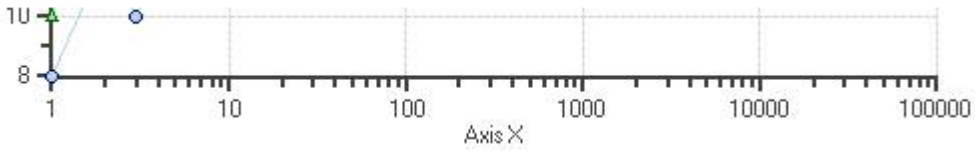
12.1.8.3 UnitMajor 和对数轴

对于对数轴比例，UnitMajor 被作为每个循环基础值的乘数，并提供一个提示作为对数底每个循环中注释的间隔。那就是（UnitMajor*基础循环的值）是大约每个循环注释的值的增量。对于数字对数基础值，这个结果通常是准确的。对于浮点值，对线性比例近似到比较准确的数字。

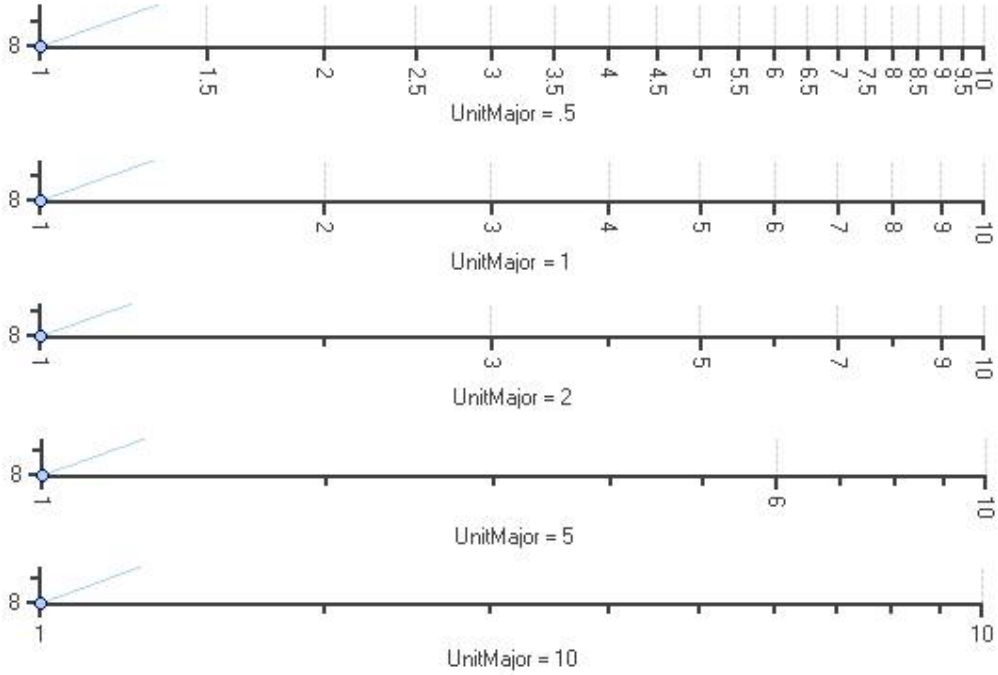
UnitMajor 和对数轴的详细说明

通常，当使用对数比例，一个图表轴的边界会跨越几个对数底循环。在这种情况下，通常对 UnitMajor 的线性规范就不再有意义了，因为对一个循环合适的值对上一个或下一个循环是没有意义的。UnitMajor 要设定为的值必须适合对数底的每个循环。

如果你不明白，想一下下面的轴你要用什么单个的，固定的，增量的值。



基于上面的原因，对于对数轴，图表假设 UnitMajor 定义了每个循环基础值的小数部分。如下例：



在每个例子中，基础循环的值是１，每个循环的下一个注释值＝前面的数＋（循环的基础值 * UnitMajor). UnitMajor 的最大值是 LogarithmicBase。UnitMajor 自动的值永远是 LogarithmicBase.

当所有的注释的值计算出来后，一个好的舍入算法就提供出来了，数据很易读。行为也许看起来有些古怪，但是它的结果适合任何的对数底同时给注释获取数字，这样阅读就合理了。

例如，上面的图示对数底为１０，但是也有自然对数需要考虑，例如对数底２，对数底Ｘ等等。

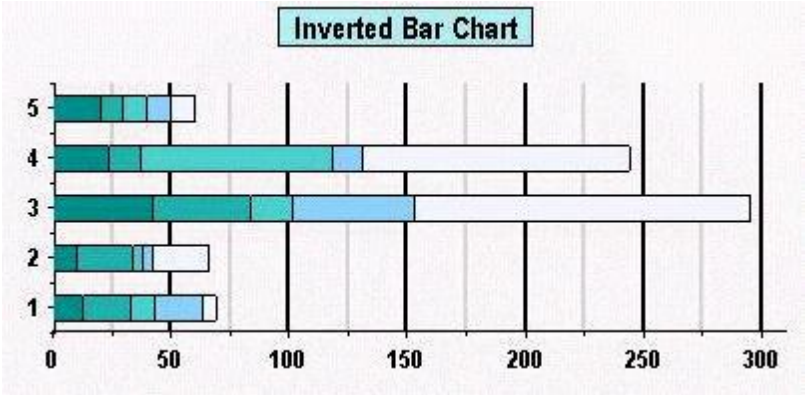
12.1.9 反向和翻转图表轴

当数据表包含很大范围的Ｘ或者Ｙ值，有时候一般的图表的结构就不能最有效地显示信息了 .把一个有纵向的Ｙ轴并且轴的注释从最小值开始的图表变为图表反向或者轴翻转会在视觉上更能满足要求 .因此 ,C1Chart 给 ChartArea 提供了 Inverted 属性 ,给轴提供了 Reversed 属性．

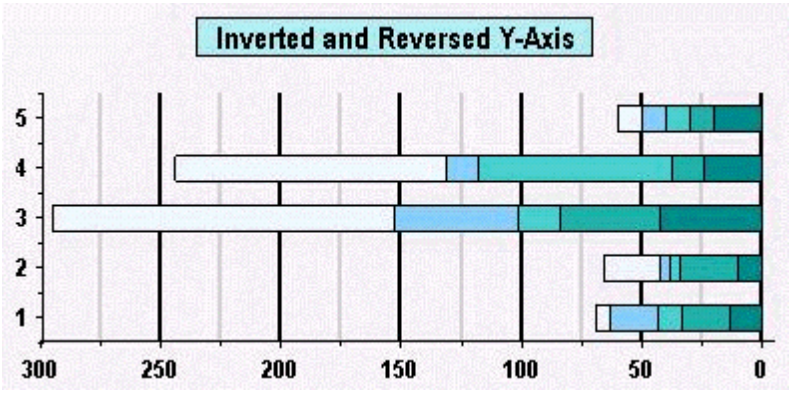
设定 ChartArea 的 Inverted 为 True 将翻转轴．就是Ｘ轴代替Ｙ轴，Ｙ轴代替Ｘ轴．最初图表建立的时候Ｘ轴水平显示，Ｙ轴纵向显示．这个功能通常用于条线图．

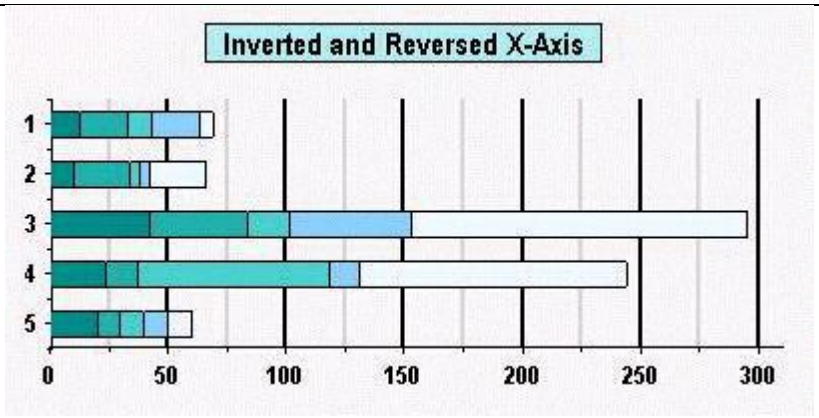
注意：

一个反向的条线图是 Inverted 属性设为 True 时的标准条线图．因此，反向条线图没有特殊的 ChartType．



设定 ChartArea 的 Reversed 属性为 True 将翻转轴．这就是说轴的最大的一边代替轴的最小的一边，最小的一边显示在原来最大的一边．最初图表在Ｘ轴的左边显示最小值，Ｙ轴的底边显示最小轴．设定轴的 Reversed 属性，会把这些最大值和最小值放在一起（例如Ｘ轴的最小值和Ｙ轴的最大值放在一起）





12.2 轴注释

沿着轴的注释是任何图表的重要部分．2 D图表用数字给轴做注释，这些数字是基于输入到 ChartDataArray 对象的数据．

图表自动创建最适合的注释，即使图表的数据变化了．Annotation 属性能够被改变来完善这个过程．

下面的属性列表列出了 C1Chart 给轴注释提供的格式和外观属性．

属性	描述
AnnoFormat	一套事先定义的格式用于格式化轴旁边的值
AnnoFormatString	当 AnnoFormat 设定为"NumericManual" 或者 "DateManual" 时使用的.NET 格式化字符串用于格式化轴旁边的值 .如果 AnnoFormat 设定为"NumericManual" 或者 "DateManual"并且 AnnoFormatString 是空，那么图表使用一个计算法则找到最好的可用的格式化方法．
AnnoMethod	定义轴旁边显示什么值．选项是"Values",指的是显示系列实际的值．如果是"ValueLabels",指的是显示在 ValueLabels 集合中的元素
ValueLabels	当 AnnoMethod 设定为 "ValueLabels"时，显示在轴旁边的成对的文字／值的集合.当你想在轴上显示字符串而不是数字值的时候这个属性非常有用（例如，你也许要画产品价格图并希望在 X 轴显示产品名）
AnnotationRotation	允许你旋转注释以减少所占的空间
NoAnnoOverlap	能让你定义是否轴注释是否允许重叠．

C1Chart 包含下面的轴注释种类

- 值注释
- ValueLabels 注释

- 混合注释

下面的主题介绍了不同种类的轴注释并且介绍了轴注释的位置和旋转．

12.2.1 值注释

值注释是图表根据数据本身自动产生数字型注释时使用的．轴注释可以用于任何的轴，任何的图表类型，任何的数据布局．它是由轴的下面的属性控制的．

属性	描述
AnnoFormat	取得或设置轴注释的格式
AnnoFormatString	取得或设置用于手动格式化的注释格式字符串

设定 AnnoFormat 属性的值为 FormatEnum 值之一会将输入到数据组中的数据格式化为.NET 的二十二种格式之一．设定 AnnoFormat 为 FormatEnum.NumericManual 或者 FormatEnum.DateManual 允许输入一个格式化好的字符串到 AnnoFormatString 属性中．手动格式化的字符串被命名到.NET 框架，下面提供了更多的信息．

日期时间格式的字符串

日期时间格式的字符串分为两个类别

1. 标准的日期时间格式的字符串
2. 自定义的日期时间格式字符串

自定义数字格式字符串

你也能够使用自定义的数字格式字符串定制你的格式化字符串

如果 AnnoFormat 设定为 FormatEnum.NumericManual 并且 AnnoFormatString 为空，那么数字的值会被格式化为和 FormatEnum.NumericGeneral.相同．

如果 AnnoFormat 设定为 FormatEnum. Date Manual 并且 AnnoFormatString 为空，那么日期和时间的值会被格式化为根据轴的最大值和最小值定义的合适的时间间隔．

12.2.2 值标签注释

作为一个非常固定的注释类型． ValueLabels 在特定的轴坐标显示定义的文本，这种注释只在特定坐标放置标签或者当给图表不提供的表格制作标签时是非常有用的．ValueLabels 标签可以被用于任何轴，任何图表类型，以及任何的数据布局，除了饼图．

使用 X 轴的 ValueLabels，能给非堆积的饼图的每个饼／点显示标签．ValueLabels 按照 ValueLabel 集合中的顺序提供给饼或者点 标签只有 X 轴 AnnoMethod 属性设定为 ValueLabels 才会显示．标签的显示还由 X 轴的属性控制，包括 Alignment, AnnoMethod, Compass, Font, ForeColor, Rotation, ValueLabels, VerticalText 和 Visible 属性．不仅 X 轴的属性影响标签，Inverted 属性也影响标签的显示，这个属性决定了是否 X 轴的 Compass 属性的值是 North, South 或者 East, West

如果 AnnoMethod 属性设定为 AnnotationMethod.ValueLabels，图表将标签放置在沿着一个轴的明确的位置．ValueLabels 属性，是个 ValueLabels 的集合，提供了字符串的列表以及它们的位置．例如，下面的代码设定图表的标签在位置 1 ， 2 ，和 3 ．

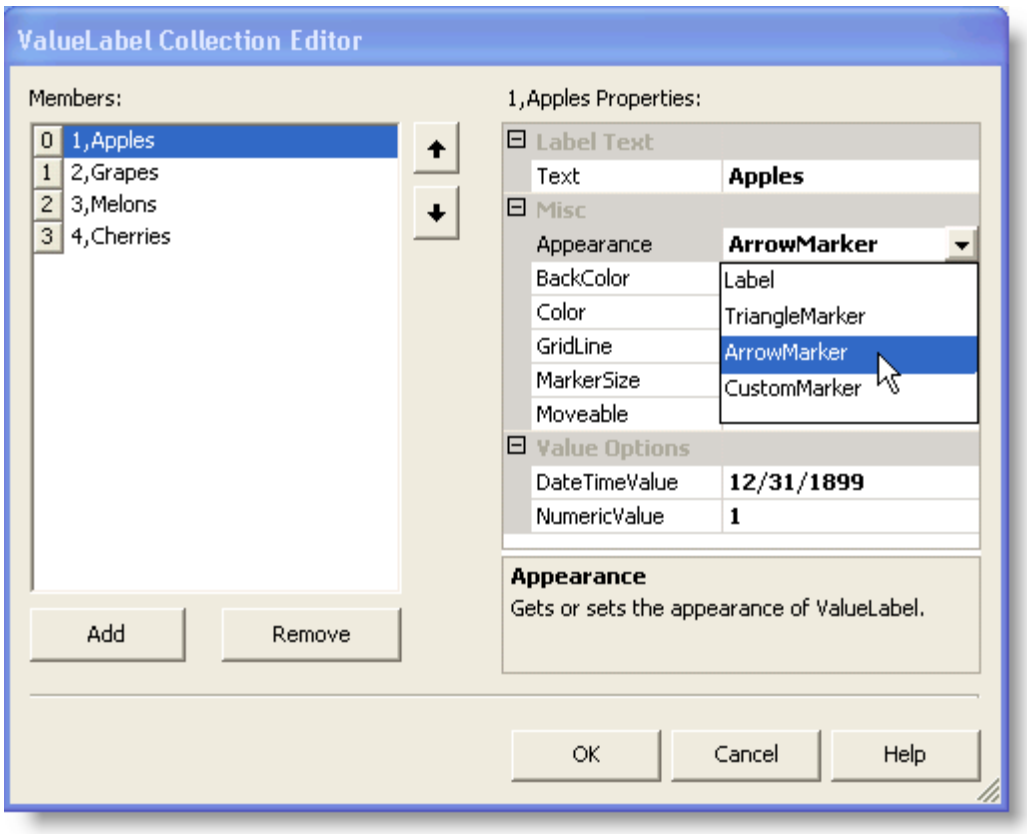
- Visual Basic

```
With C1Chart1.ChartArea.AxisX.ValueLabels
.Add(1,          "one")
.Add(2,          "two")
.Add(3,          "three")
End With
C1Chart1.ChartArea.AxisX.AnnoMethod = AnnotationMethodEnum.ValueLabels
```

- C#

```
ValueLabelsCollecton VlColl = c1Chart1.ChartArea.AxisX.ValueLabels;
VlColl.Add(1,    "one");
VlColl.Add(2,    "two");
VlColl.Add(3,    "three");
c1Chart1.ChartArea.AxisX.AnnoMethod = AnnotationMethodEnum.ValueLabels;
```

ValueLabels 在设计时也能通过 ValueLabel 集合编辑器修改 .关于 ValueLabel 集合编辑器的更多信息，参见 ValueLabel 集合编辑器（ 5 6 页）



下面的表说明了 ValueLabel 类的 Text, Appearance, BackColor, Color, MarkerSize, 和 Moveable 属性 .

注意数字的值的 1 和 2 的标签显示为一个箭头 .你能够在设计时和运行时设定你的标签的外观 .

设计时你能够在 ValueLabel 集合编辑器中选择下面的标签类型来设定标签类型：Label, Triangle Marker, ArrowMarker, 或者 CustomMarker．标签类的外观属性从 ValueLabelAppearanceEnum.中获得值．运行时可以使用下面的编码设定标签的外观．

- Visual Basic

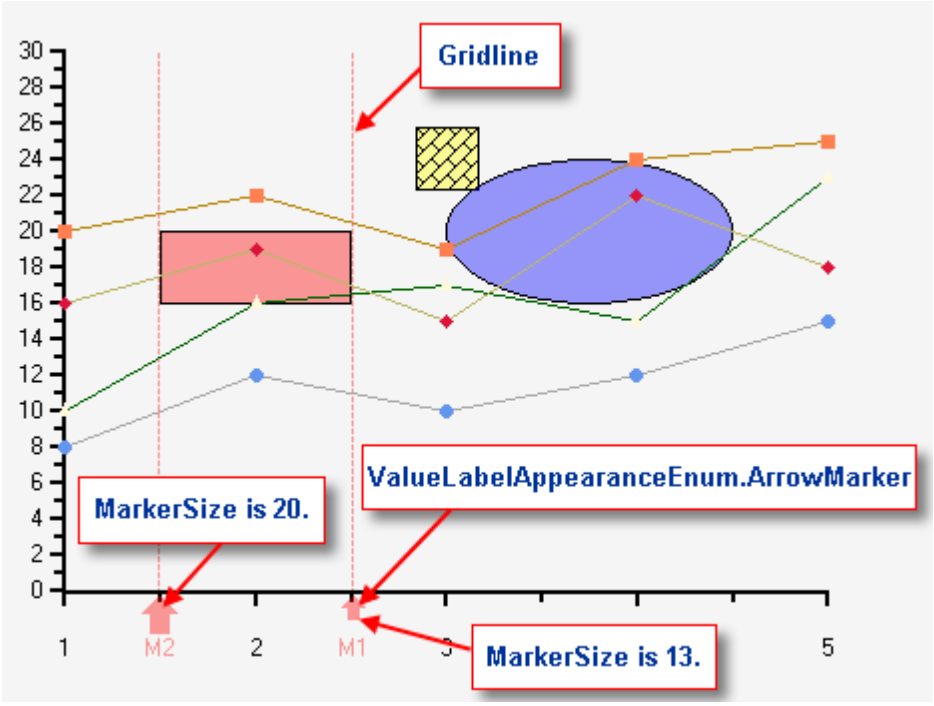
```
vl.Appearance = ValueLabelAppearanceEnum.ArrowMarker
```

- C#

```
vl.Appearance = ValueLabelAppearanceEnum.ArrowMarker;
```

注意：在上面的例子中，文字 vl,代表了值标签的变量的名字

如下图所示的 MarkerSize 属性说明了当设定为不同的大小的时候标记如何显示．通过设定每个标记标签的 Moveable 属性为 True，你可以沿着 X 轴移动M１和M２．这对一个用输入设备定义轴的值从而影响带有笛卡儿坐标的图表非常有用 .下图的和 ValueLabels 关联的两个格线帮助两个标记增加视觉效果．



ValueLabels 集合中每个单独的项目可以通过以下三种方法访问 给 ValueLabel 输入所有索引（从 0 开始），引用要被修改的 ValueLabel 的值．引用 ValueLabel 的文本的值．例如，下面三行代码全部都是设定 ValueLabels 集合的第一个项，把该项的值从 1 改为"one":

- Visual Basic

```
C1Chart1.ChartArea.AxisX.ValueLabels(0).Text = "one"
C1Chart1.ChartArea.AxisX.ValueLabels(1.0).Text = "one"
C1Chart1.ChartArea.AxisX.ValueLabels("one").Text = "one"
```

- C#

```
c1Chart1.ChartArea.AxisX.ValueLabels[0].Text =          "one";
c1Chart1.ChartArea.AxisX.ValueLabels[1.0].Text          =          "one";
c1Chart1.ChartArea.AxisX.ValueLabels["one"].Text        =          "one";
```

12.2.3 混合注释

作为一个非常灵活的注释种类，混合注释包括自动的注释和值标签．每种注释都会在定义的轴坐标上显示可见的文本．这个注释允许值和值标签重叠．

12.2.4 轴注释的位置

选择 TickLabelsEnum 枚举中的下列值之一就能使用设定轴注释的位置

值	描述
TickLabelsEnum.None	沿轴没有注释.
TickLabelsEnum.High	竖轴上最大值的附近画注释，并且在图形区之内．对于 X 轴的注释，如果 Group 0 的数据不可用而 Group1 的数据可用，Y2 轴定义位置，否则使用 Y 轴．与交叉轴重叠的注释会被消掉.
TickLabelsEnum.Low	竖轴上最小值的附近画注释，并且在图形区之内．对于 X 轴的注释，如果 Group 0 的数据不可用而 Group1 的数据可用，Y2 轴定义位置，否则使用 Y 轴．与交叉轴重叠的注释会被消掉.
TickLabelsEnum.NextToAxis	在轴附近画注释，这是默认值.

High 和 Low 定义交叉轴的最大和最小位置 .例如 ,如果给 X 轴 TickLabelEnum 定义了 High 那么注释就位于 Y 轴最大值附近，这对图表的顶部不是必须的(Reversed = true)

High 和 Low 的典型用法是如果你有小于 Y 轴原点的值．在下面的例子中，定义 AxisX.TickLabels = TickLabelsEnum.Low， 注释就为于 Y 轴 - 2 5 的值附近．

用程序设定 TickLabelsEnum 的值：

- Visual Basic

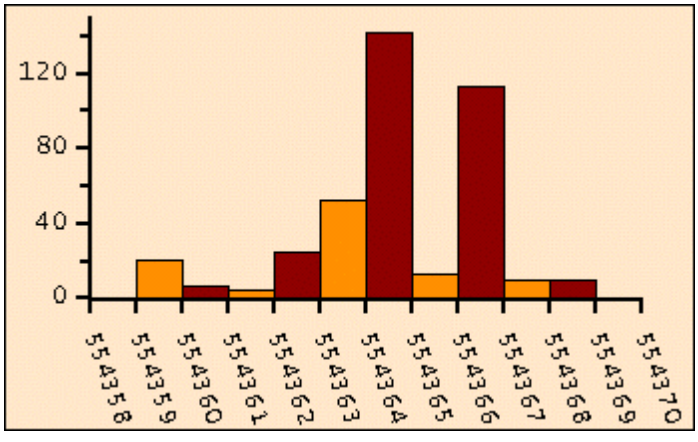
```
Dim ax As Axis =c1Chart1.ChartArea.AxisX
ax.TickLabels = TickLabelsEnum.Low
```

- C#

```
Axis ax = c1Chart1.ChartArea.AxisX;
Ax.TickLabels = TickLabelsEnum.Low;
```

12.2.5 轴注释的旋转

使用轴的 AnnotationRotation 属性可以逆时针方向按照定义的度数旋转轴注释 .这个属性在 X 轴有很多的注释的时候非常有用．旋转注释+/- 30 – 60 度就能够在水平轴上让很多的注释在有限的空间里表示出来．通过使用 AnnotationRotation 属性， X 轴的注释不会重叠，如下所示：



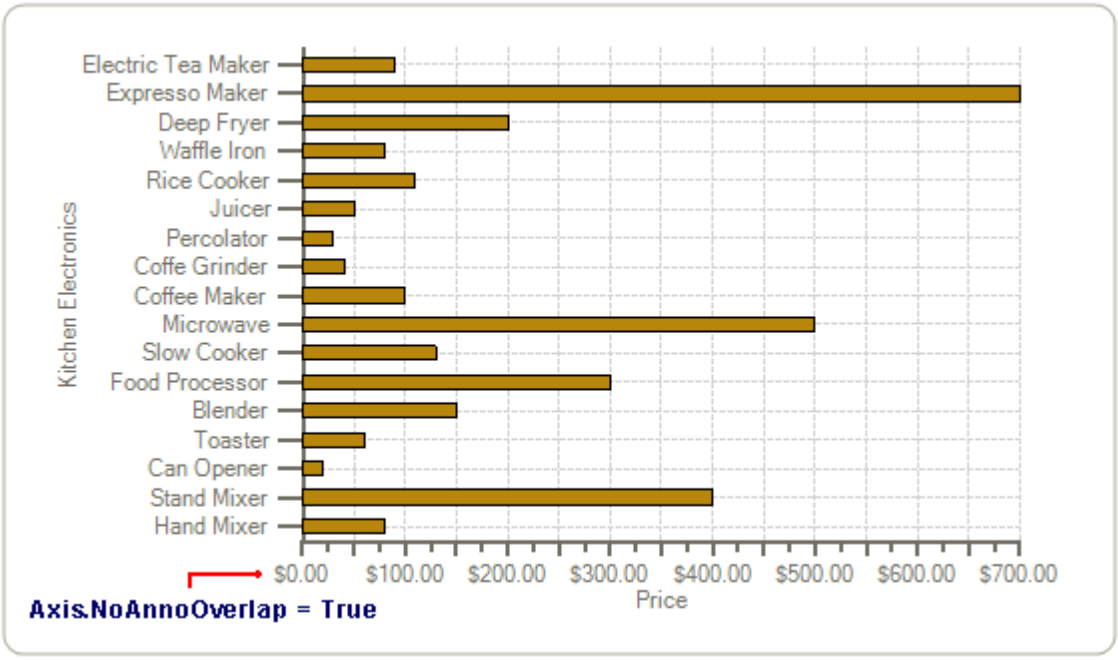
12.2.6 轴注释重叠

当你在轴上没有足够的空间显示你的注释的时候，你可以设定 NoAnnoOverlap 为 True，这

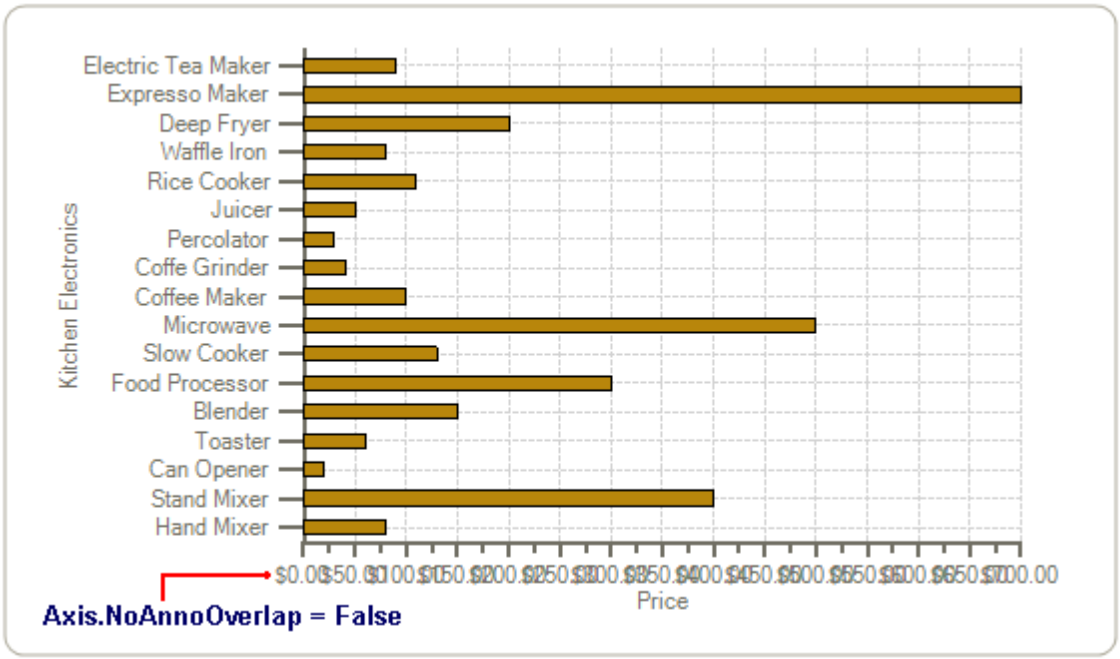
样就能够无条件地阻止画任何的轴注释重叠到前面画的注释上去．

如果你已经设定 AutoMajor 属性为真，那么你就不需要使用 NoAnnoOverlap 属性了，因为 AutoMajor 属性会自动计算主刻度线的最合适的值．

下面显示了 NoAnnoOverlap 属性设定为 True 时的图表



下面显示了 NoAnnoOverlap 属性设定为 False 时的图表



12.3 Plot Area

数据画在图表的图形区．在 PlotArea 对象中，你可以定制或者创建 PlotArea 的类型．例如你可以使用 BackColor 和 BackColor2 定义的图形区得背景色．你还可以通过 ForeColor 属性

设定 PlotArea 的前景色．

下表列出了图形区的外观属性及其功能

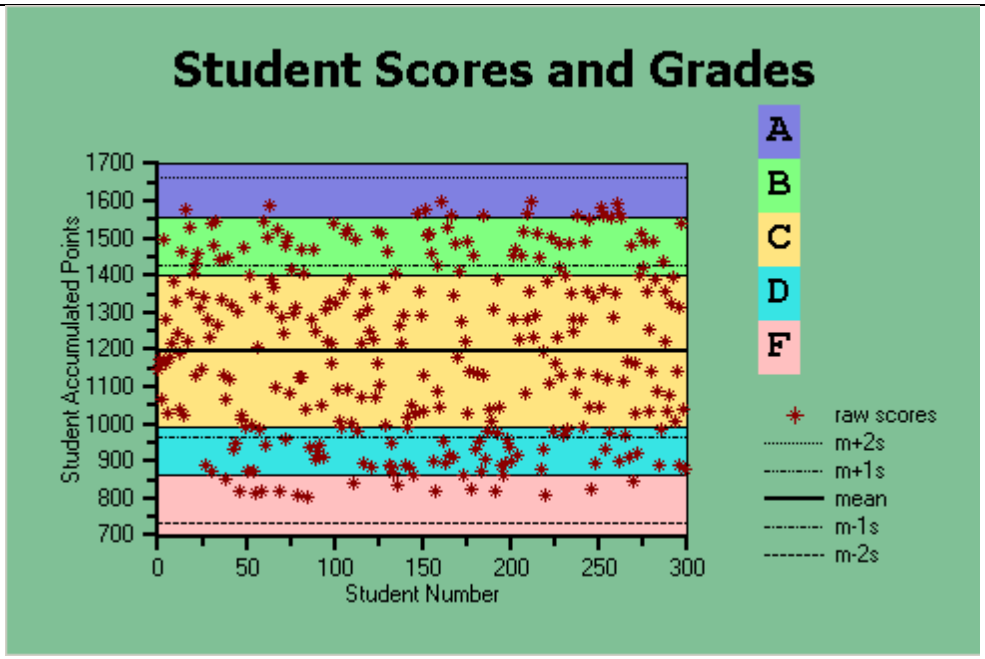
属性	描述
AlarmZones	AlarmZones 属性得到和现在的 PlotArea 关联的 AlarmZones 对象
BackColor	取得或设定 PlotArea 的背景色.从 ChartArea 中继承．
BackColor2	取得或设定第二个背景色.
Boxed	取得或设定 PlotArea 是否封装在盒子里
ForeColor	取得或设定 PlotArea 的前景色.从 ChartArea 中继承．
GradientStyle	取得和设定背景渐变填充的类型
HatchStyle	取得和设定背景影线填充的类型.
Opaque	取得和设定 PlotArea 的背景是否是不透明的
Size	取得 PlotArea 按图标控件的客户端坐标计算的大小
View3D	取得 View3D 对象.
Visible	取得和设定 PlotArea 是否可见

C1Chart 控件有一个 AlarmZone 集合编辑器，可以在设计时通过 C1Chart 的属性访问．AlarmZone 集合编辑器有一个窗口可以很方便地允许用户编辑／创建 AlarmZones．使用编辑器，用户可以新增／删除一个或多个报警区域，并且修改或者设定每个警报区域的属性．关于 AlarmZone 集合编辑器的更多信息，请参照 AlarmZone 集合编辑器（４０页）．

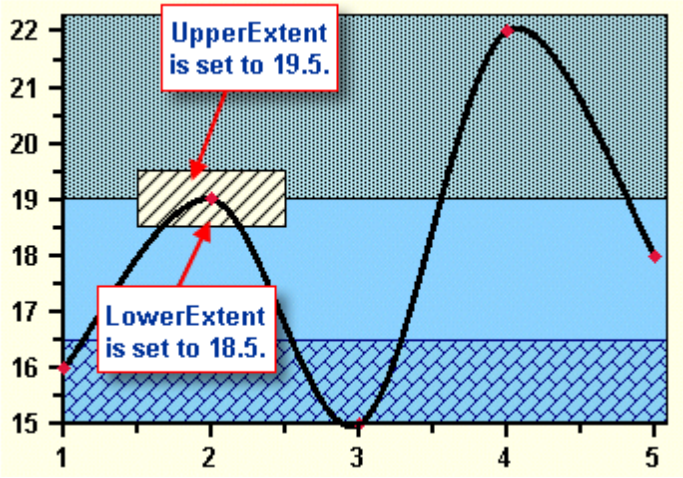
下面的章节解释什么是警报区域以及在图表中如何使用警报区域．

12.3.1 警报区域

警报区是一系列的能够放置在图形区后面的区段或者形状，但是在图表的背景之前．一般来说，警报区和格线有近似的用法，但是警报区可以被修改这样就能够让警报区更有用并且视觉上更抢眼．而且，警报区可以用来高亮图表中重要的Ｙ值．例如，下面的图表使用了五种不同的警报区（Ａ，Ｂ，Ｃ，Ｄ，Ｆ）代表了学生的年级．每个警报区以不同的颜色表示用于和其他的警报区区别开来．注意警报区是如何帮助显示重要的Ｙ值的（在例子中，是学生的累计的点数／年级）．



通过使用 UpperExtent 和 LowerExtent 属性，每个警报区的区段能够调整为特定的值．下图显示了警报区的区段通过使用 UpperExtent 和 LowerExtent 属性被调整为一个低的 Y 值１８．５和一个高的 Y 值１９．５的效果．



注意：警报区在 XY-Plot, Bar, Stacking Bar 和 Candle 图表中使用最为有效.

你可以通过使用 Shape 属性设定你的警报区的形状．你可以设定警报区为椭圆形的，长方形的，或者多边形的．Shape 属性可以在设计时通过 AlarmZone 集合编辑器设定，或者在运行时设定．使用编码，你需要使用 AlarmZoneShapeEnum 属性来获得圆形的，长方形的，或者多边形的形状．下面的编码给警报区创建了一个长方形的形状．

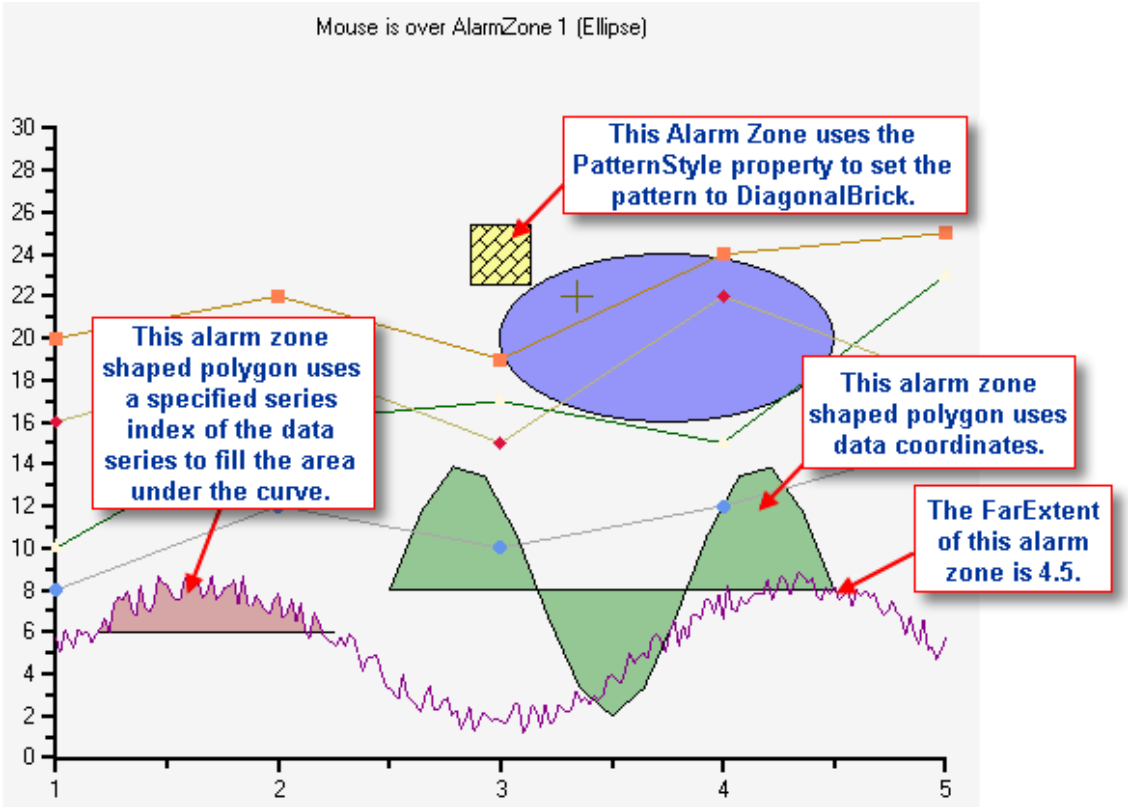
- Visual Basic

```
alarmzone.Shape = AlarmZoneShapeEnum.Rectangle
```

- C#

```
alarmzone.Shape = AlarmZoneShapeEnum.Rectangle;
```

下图列出了几个警报区的属性．注意光标指向警报区时下面的文本是如何描述的．这种技术可以使用 AlarmZoneAtCoord 方法做到．AlarmZoneAtCoord 方法根据指定的坐标取回最前面 AlarmZone 的参照．数据和 DataSeries 是多边形数据的两种不同的源类型．下面的图表中，绿色的多边形使用了用户填充的数组源来获得多边形的坐标．其他的多边形使用 DataSeries 来获得用户定义的系列的索引．



警报区提供下列形状：长方形，椭圆形和多边形．当使用多边形，多边形的数据可以明确的指定或者定义一个数据系类(只是ＸＹ　Ｐｌｏｔｓ）这可以很容易的填充曲线下的特定区域，像上面的图标的多边形一样．

12.3.1.1 增加警报区

AlarmZones 可以在设计时通过 Visual Studio 属性窗口中的 AlarmZone 集合编辑器来添加或者通过 AlarmZones 对象用程序添加．

用编程的方法给 AlarmZonesCollection 添加一个 AlarmZone

- Visual Basic

```
C1Chart1.ChartArea.PlotArea.AlarmZones.AddNewZone()
```

- C#

```
c1Chart1.ChartArea.PlotArea.AlarmZones.AddNewZone();
```

在设计时访问集合编辑器

1. 右键点击 C1Chart2D 控件并且从右键菜单中选择 Properties

2. 在属性窗口中，展开属性窗口中的 ChartArea 节点 ，然后展开 PlotArea 节点，点击 AlarmZones 属性旁边的省略号

关于 AlarmZone 集合编辑器的更多信息，请参照 AlarmZone 集合编辑器（４０页）．

12.3.1.2 定义警报区属性

12.3.1.3

警报区有两套基本的属性．第一套属性根据图表的数据值定义警报区的边界．设定 NearExtent 和 FarExtent 属性来使用 X 轴的范围内的值设定 AlarmZone 的 Left 和 Right 边界．相反，设定 UpperExtent 和 LowerExtent 属性来使用 Y 轴的范围内的值设定 AlarmZone 的 Upper 和 Lower 边界 .例如 ,设定上图中的绿色的 AlarmZone 的 NearExtent 为 2.5 ,FarExtent 为 4.5，LowerExtent 为 8.0，UpperExtent 为 14.0. 注意这些属性的名字是 UpperExtent, LowerExtent, NearExtent, 和 FarExtent 而不是 X, Y, Height, 和 Width．一个警报去重要的优点是它不是固定的，而是和实际图表数据关联的 . 这就意味着数据改变了，和数据关联的警报区也会改变因此图表有不止一个视觉效果．要防止 AlarmZones 因为数据改变而消失你可以设定 MinHeight 和 MinWidth 属性．并且，MinHeight 和 MinWidth 属性也允许警报区按固定尺寸放置．

第二套属性定义警报区的类型特性 . 可以给警报区设定 Shape, BackColor 和 PatternStyle 属性 . 你可以给警报区选择三种不同的形状类型：长方形的，椭圆形的或者多边形 . 如果你创建了一个多边形形状的警报区你需要定义 PolygonSource 属性，这个可以在设计时使用 AlarmZone 集合编辑器定义或者在运行时创建 .设定 ForeColor 属性设定 PatternStyle 属性定义的模式的颜色．

13. 自定义图表元素

当图表的数据和轴被正确的格式化时 ,我们可以自定义它的元素来让图表看起来更加的清晰和更加的专业化。以下是一些关于自定义一个图表的外观的主题。

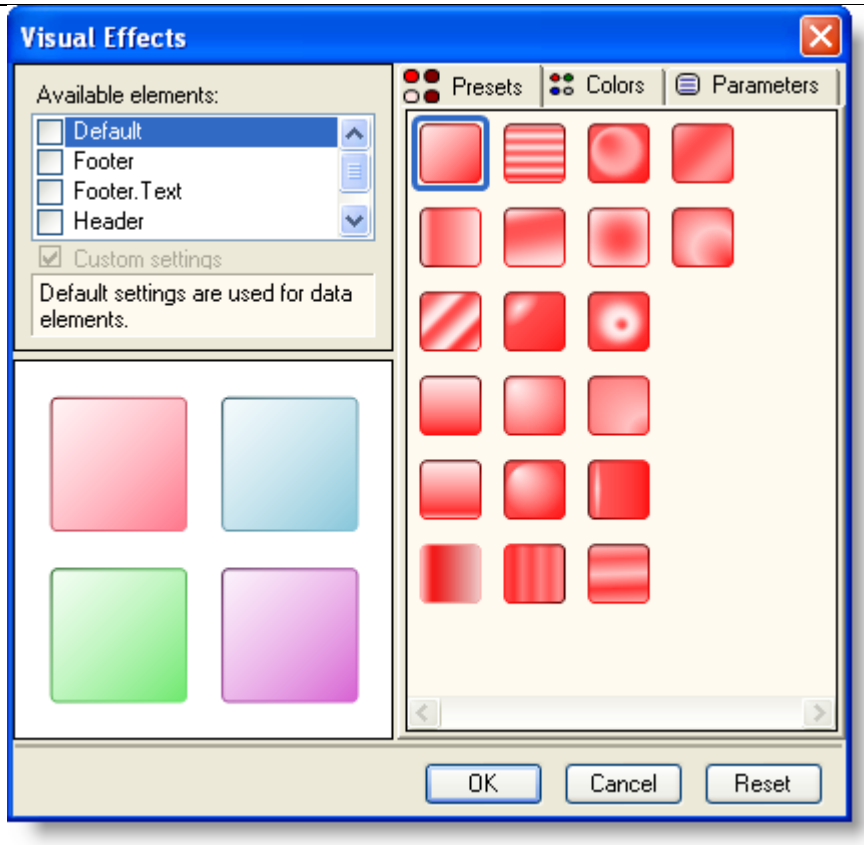
13.1 视觉效果设计器

视觉效果是一个可以用来直观地提升 Chart2D 元素(诸如数据序列，表头，页尾等)外观效果的工具。所有既存的工程都能够使用该工具的最新功能。通过**视觉效果**设计器中的少数几个简单的步骤就能够明显地提升图表的外观。

局限性

视觉效果渲染不是很适合在非常大且复杂的数据数组或者在有很高的性能要求的场景下使用。

当你运行图形化的视觉效果设计器时，它的外观如下:



关于如何访问视觉效果设计器的更多信息，请参见访问视觉效果设计器(307 页)。

13.1.1 视觉效果设计器导航

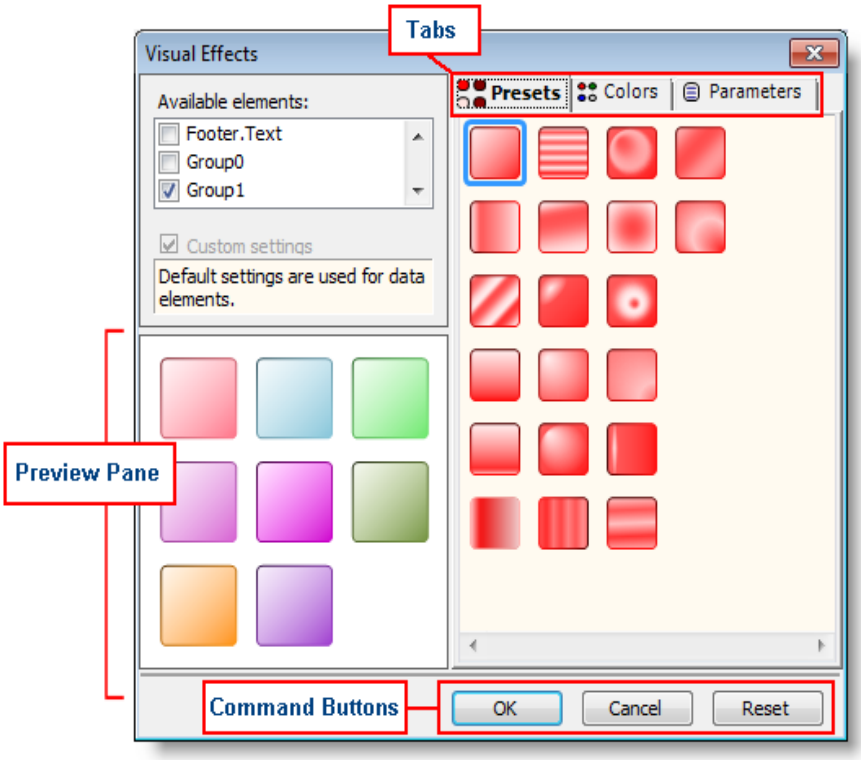
视觉效果设计器有非常直观的接口外观。

设计器由两部分组成。第一部分位于视觉效果设计器的左边。它显示了支持视觉效果渲染的元素的列表，和一个显示了当前元素外观的预览面板。

第二部分位于**视觉**效果设计器的右边，该部分含有 3 个选项卡:

预设，颜色，和参数。

除了图表元素，预览面板，选项卡页外，设计器的底部还有一些命令按钮。命令按钮包括 OK，Cancel，和 Reset 按钮。OK 按钮的功能是关闭视觉效果设计器并保存所有的修改。Cancel 按钮的功能是取消视觉效果设计器并回滚在编辑模式下的任何修改。Reset 按钮的功能是重置选中的属性值。



13.1.2 视觉效果元素

视觉效果元素可以独立地应用到不同的图表元素上。每一个支持视觉效果的元素都可以拥有自己的或者继承自父元素的自定义设置。

下表描述了在视觉效果设计器中的所有可用元素

元素名称	父元素	描述
Default		用来绘制所有的数据元素
Header	Default	表头背景
Header。Text	Header	表头文字
Footer	Default	页尾背景
Footer。Text	Footer	页尾文字

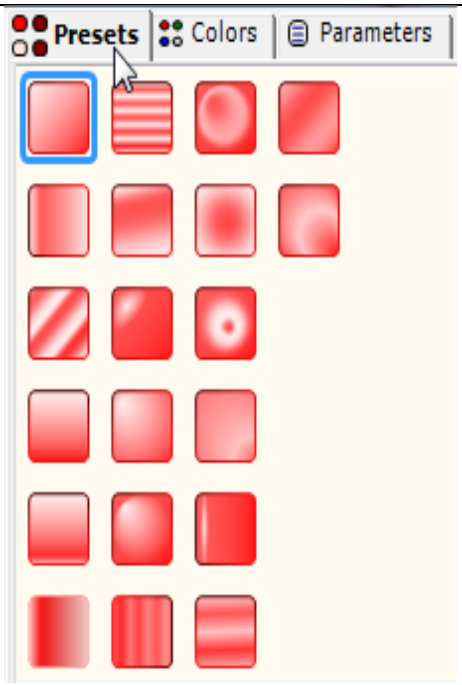
关于如何将视觉效果应用到图表元素上的更多信息，请参见添加视觉效果到图表元素(307 页)

13.1.3 视觉效果设计器选项卡

视觉效果设计器含有**预设**，**颜色**，和**参数** 3 个选项卡。

预设

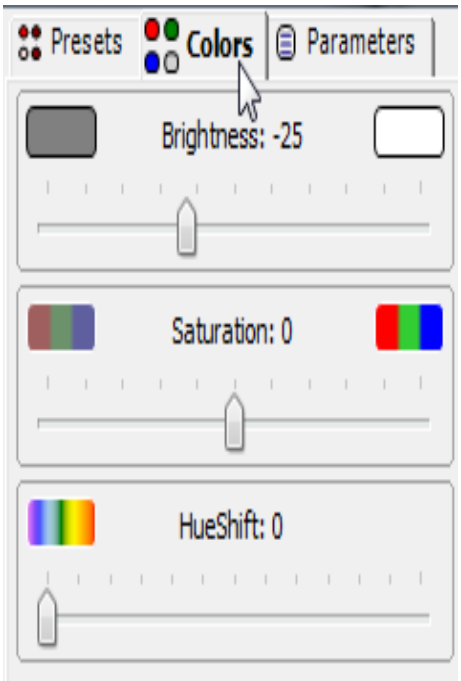
预设选项卡的外观如下:



预设选项卡含有一系列可用的 **Chart2D** 元素的内置设置。它能够通过一个简单的点击来非常直观地添加一个或者所有的可用的 **Chart2D** 元素。

颜色

颜色选项卡的外观如下:



颜色选项卡页含有颜色的亮度，饱和度，色调转换的游标，亮度，饱和度，色调转换代表了红，蓝，绿的配色方案。你可以通过调整亮度和饱和度来得到任何你想要的图表元素上的色调，而不是把你限制在几种特定的颜色之内。你甚至可以通过色调转换游标来改变所有的颜色。关于图表元素的颜色游标的更多信息，请参见**使用颜色游标扩展既存的图表数据序列**(319 页)**或者图表表头和页尾**@使用颜色游标来为图表表头和页尾扩展既存的颜色。

亮度游标

亮度是一个色调的亮度或者暗度。当要增加亮度的等级时，向右滑动亮度游标。相反地，当要减少亮度的等级时，向左滑动亮度游标。

饱和度游标

饱和度是一个色调从灰色基调到纯色的色调的强度。饱和度游标的取值的范围从-100 到 100。其中，-100 代表没有饱和度(灰色基调)，100 代表高饱和度(纯色调)。

向右滑动饱和度游标能使颜色看其来更鲜艳，更强烈。随着你向右移动游标，颜色的强度将随之进一步的增加。

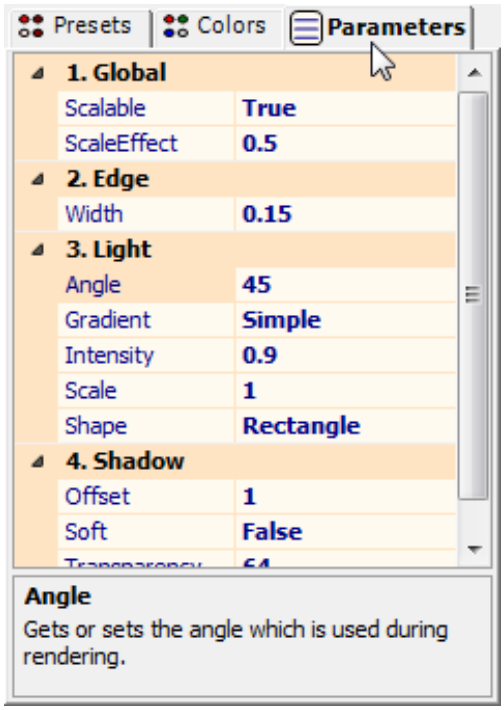
为了让颜色变淡，向左移动游标。当你向左移动游标时，颜色的强度将进一步的减少。

色调转换游标

色调代表色轮中的一个特定的颜色。色调的值取决于亮度和饱和度的值。当高饱和度但没有亮度(值是 0)时，色调看起来是灰黑色的。当高亮度但是没有饱和度时，色调看起来是无光泽的/白色。

参数

参数选项卡的外观如下图:



参数选项卡包含 Chart2D 元素的缩放，边框，灯光和阴影效果的属性。使用这些属性你可以在自定义图表元素上用于无限的可能性。关于这些属性的更多信息，请参见视觉效果参数(236 页)。

13.1.4 视觉效果参数

本节描述了所有的视觉效果参数，以及它们之间的相互关系。

13.1.4.1 缩放和边框效果

Scalable 属性的默认值是 True。当 **Scalable** 属性起效时你可以使用 ScaleEffects 属性来设置图像的缩放因子。你可以把缩放效果因子设置为从 0 到 1 的值。默认值是 **0.5**。当缩放因子大于 0.5 时，缩放将把图像变的更光亮。反之，当缩放因子小于 0.5 时，缩放将把图像变的更暗。

你可以通过使用 **ScaleEffects** 属性而非 **Offset** 属性来获得一个更加光亮或者更加暗的效果，因为它保留了相对更好的图像对比度。当 **Offset** 属性设置为 **0** 时，你感觉不到 ScaleEffects 属性的效果。

你可以通过设定 Width 属性来指定元素边框的宽度。当 **Scalable** 属性设置为 **True** 时 Width 属性的单位是相对单位，否则的话它以像素为单位。

关于在图表元素上应用缩放和边框效果的更多信息，请参考**增加数据序列中符号的大小** (320 页)

13.1.4.2 灯光效果

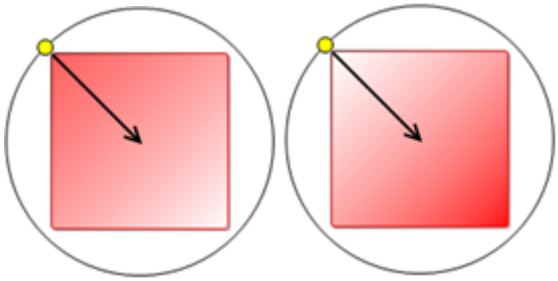
你甚至可以通过使用角度，渐变，明暗度，缩放，和形状来改变光源的效果从而达到自定义图表元素的属性的目的。

你可以通过更改 **Angle** 属性来改变一个视觉渲染的角度到一个不同的度上。 **Angle** 属性的默认值是 45 度，其取值范围是从-180 到 180 度。当你改变 **Angle** 属性的值的时候，光束指向颜色并且以反时钟方向在正方形中移动。

你可以通过使用 **Gradient** 属性来在图表元素上应用光渐变效果。该属性的默认值是 Simple。该属性含有 3 个不同的设置: **Simple**， **SigmeBell** 和 **Triangle**。当你在光渐变属性上选择 SigmeBell 或者 Triangle 值时，一个 **Focus** 属性就会出现。 **Focus** 属性的默认值是 0.1。当你增加 **Focus** 属性值时，光线逐渐地从以前的位置在相反的方向上移动。例如，当 **Focus** 属性从 0 变为 1 时，光线位置会为 SigmeBell 或者 Triangle 渐变移动到相反的方向上。

下图表示了光线位置在 **Focus** 属性为 0 和设置为 1 时的转变。

Focus = 0　　Focus = 1



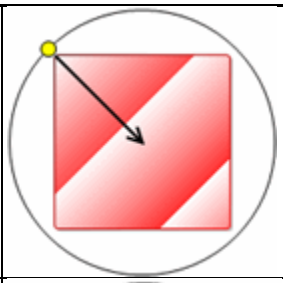
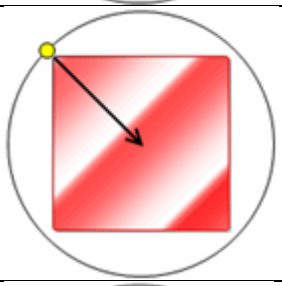
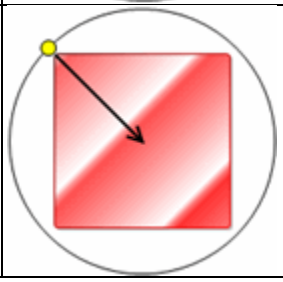
当光线明暗度设置为 0 时，渐变效果不是很明显。当你增加光线明暗度时光线的渐变效果会有更明显的效果。明暗度的默认值是 0.9。

你可以通过设置 **Scale** 属性值小于 1 来在一个图表元素上展示重复光图案。当你减少 Scale 属性的值时，光图案重复的更多。 **Scale** 属性的取值范围是从 0 到 1。

注意: **Scale** 属性仅在矩形光形状上才能使用。

下表展示了在光线明暗度设置为 1 且 **Scale** 属性设置为 0.4 时不同的光线渐变的效果。

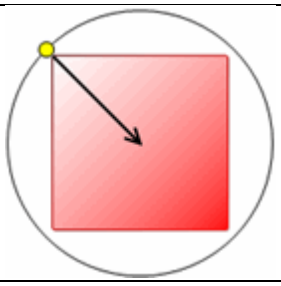
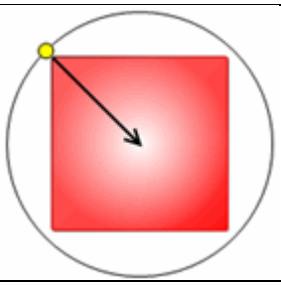
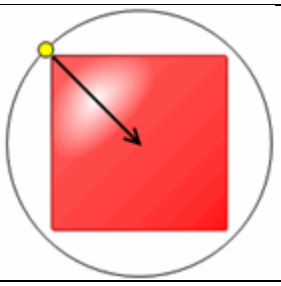
Light Gradient	Image

Simple	
SigmeBell	
Triangle	

你可以通过**视觉效果**设计器的 **Shape** 属性把光线的形状设置为 rectangle， ellipse， 或者 edge，默认的光线形状为 rectangle。

当你设置光线形状为 edge 时， Gradient 和 Scale 属性由于不会明显地影响 edge 光线形状而不可用。当你选择 Ellipse 形状时，两个额外的 Shift 和 Size 属性被添加以用来最更进一步的自定义。你可以通过滑动 Shift 游标来改变经过元素的 Ellipse 形状的光线。你可以通过从左到右地拖动 Size 游标来增加 ellipse 光线形状的大小。

下表展示了 rectangle， ellipse， and edge 形状如何表现在图表元素上:

Rectangle	Ellipse	Edge
		

关于如何添加光线效果到特定的图表元素上的更多信息，**请参见在图表表头和页尾添加一个光图案**(309 页)，在**图表表头和页尾添加一个光线形状**(310 页)，**或者调整图表表头和页尾的光线焦点**(313 页)。

13.1.4.3 阴影效果

你可以通过增加 **Offset** 属性的值来添加引用效果。当该属性的值在 3 到 5(最高值)时阴影的

效果最明显。你可以通过增加 transparency 的值来加深阴影的效果。 transparency 的默认值是 64。当你启用 soft 属性时元素上的阴影效果看起来会更加的柔和。

关于将阴影效果应用到特定元素上的更多信息，请参见在**图表的表头和页尾添加阴影效果** (312 页)。

13.2 图表标题

一个图表含有 **Header** 和 **Footer** 两个标题。一个标题由一行或者多行文字组成，并且可以自定义是否含有可选的边框。因为每个标题都可以置于图表的上部，底部，左边，右边，所以不能按照传统的观点说 Header 必须位于对象的上部，而 Footer 必须位于对象的下部。另外，Header 和 Footer 的文字对齐方式，位置，颜色和字体都是可以修改的。

13.2.1 标题，文字和对齐

可以通过使用名为图表属性设计器的智能设计器或者通过 Text 属性来自定义 Header 和 Footer 的文字。

使用上述设计器的 Header 和 Footer 工具栏或者通过 Text 属性可以添加，修改或者删除 Header 和 Footer 上的文字。Text 属性位于对应的标题节点(Header 或者 Footer)的属性窗口中。

使用 HorizontalAlignment 属性或者 VerticalAlignment 属性来指定文字是居中，左对齐，还是右对齐。

通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点可以在设计时找到这些属性。

13.2.2 标题位置和大小

使用标题的 **Compass** 属性来指定标题相对于图表区的位置。可以选择相对于图表区的 4 个方位。

使用 Location 对象的 X 和 Y 属性来自定义标题的位置。通过设置 X 或者 Y 为-1 来将位置置为自动适应位置。可以通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点下的 Location 节点来在设计时找到这些属性。

使用 Size 对象的 Width 和 Height 属性来自定义标题的大小。通过设置 Width 或者 Height 为-1 来将大小重置为自动适应大小。可以通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点下的 Size 节点来在设计时找到这些属性。

更多信息请参见**图表元素的位置和大小**(252 页)。

13.2.3 标题边框

使用边框的 Type 和 Width 属性来自定义标题的边框。可以通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见**图表边框**(242 页)。

13.2.4 标题颜色和渐变效果

可以使用 ForeColor 和 BackColor 属性来自定义标题的背景和文字颜色。

可以通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见**图表颜色**(243 页)。

13.2.5 标题字体

使用 Font 属性来自定义标题使用的字体。 可以通过 Visual Studio 的属性窗口中的 Header 或者 Footer 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见图表字体(242 页)。

13.3 图表图例

每次图表中出现数据时图表都会自动地生成一个图例。图表给序列分配位于 ChartDataSeries 对象中的名字来作为序列的标示。序列的 LineStyle 和 SymbolStyle 属性决定了随着图例的序列名字成对出现的符号。图例的位置，边框，颜色和字体都可以自定义。

13.3.1 图例位置

使用 Compass 属性来指定图例相对于图表区的位置。可以选择相对于图表区的 4 个方位。可以通过 Visual Studio 的属性窗口中的 Legend 节点在设计时找到 Compass 属性。

默认情况下图表自动选择图例的位置。使用 Location 的 X 和 Y 属性来精细地调整位置。 可以通过 Visual Studio 的属性窗口中的 Legend 节点下的 Location 节点在设计时找到这些属性。

默认情况下图表会自动计算图例的大小。使用 Size 的 Height 和 Width 属性来精细地调整大小。可以通过 Visual Studio 的属性窗口中的 Legend 节点下的 Size 节点在设计时找到这些属性。

更多信息请参见图表元素的位置和大小(252 页)

13.3.2 图例标题

使用 Text 属性来指定图例的标题。图例的标题出现在其上部居中的位置。可以通过 Visual Studio 的属性窗口中的 Legend 节点找到 Text 属性。

13.3.3 图例边框

使用边框的 Type 和 Width 属性来自定义图例的边框。可以通过 Visual Studio 的属性窗口中的 Legend 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见图表边框(242 页)。

13.3.4 图例颜色

可以使用 ForeColor 和 BackColor 属性来自定义图例的背景和文字颜色。可以通过 Visual Studio 的属性窗口中的 Legend 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见图表颜色(243 页)。

13.3.5 图例字体

使用 Font 属性来自定义图例使用的字体。可以通过 Visual Studio 的属性窗口中的 Legend 节点下的 Style 节点来在设计时找到这些属性。更多的信息请参见图表字体(242 页)。

13.4 序列的线和符号样式

ChartDataSeries 的 LineStyle and SymbolStyle 属性能够让每一个序列在展示方面更加的完善。不仅限于序列线的线和符号，这些属性还可以修改 **Pie** 扇形图表切片的颜色，改变 Radar 图表序列的颜色，或者设置 **HiLo** 图表序列的宽度。

LineStyle 属性含有 Color， Pattern， Thickness， LineJoin 和 MiterLimit 属性。通常这

些属性设置画线连接处的颜色，图案，连接样式，联接限制，和序列的线的厚度属性。

SymbolStyle 属性含有 Color， Pattern 和 Thickness 属性。通常这些属性设置序列中的符号的颜色，图案和厚度。

下表描述了这两个属性是如何影响每一种图表类型的。请注意这些属性对一些图表类型是无效的，而对于另外一些图表类型，这些属性设置的是线和符号之外的另一些属性。

值	LineStyle 属性的效果	SymbolStyle 属性的效果
Chart2DTypeEnum.XYPlot	改变连接数据点的线的颜色，厚度，图案。	改变序列的符号样式。
Chart2DTypeEnum.Pie	改变序列的背景色(段)	无效
Chart2DTypeEnum.Bar	改变序列的背景色	无效
Chart2DTypeEnum.Area	改变序列的背景色	无效
Chart2DTypeEnum.Polar	改变序列线的背景色，图案和厚度。	改变序列符号样式 ,背景色和厚度。
Chart2DTypeEnum.Radar	改变序列线的背景色，图案和厚度。	改变序列符号样式 ,背景色和厚度。
Chart2DTypeEnum.HiLo	改变序列线的背景色，图案和厚度。	无效
Chart2DTypeEnum.HiLoOpenClose	改变序列线的背景色，图案和厚度。	无效
Chart2DTypeEnum.Candle	改变下跌股票价格的背景色，和烛光形状的厚度。	改变上升股票价格的背景色，和烛光形状的厚度。

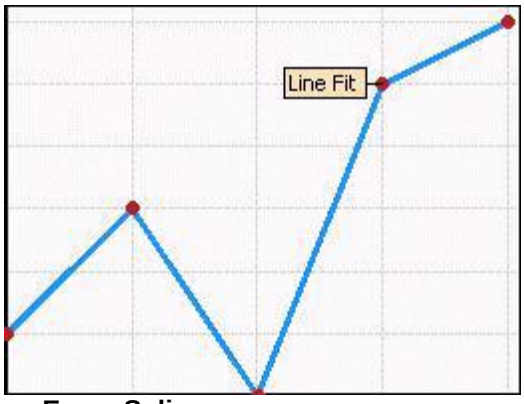
一个更精细的控制绘制的方法是使用画笔而非仅仅使用 **LineStyle** 和 **SymbolStyle** 属性来指定颜色。画笔可以提供比颜色更丰富的，更独特的外观—包括影线，渐变和纹理。关于使用画笔的更多信息，请参见**为绘制数据自定义画笔**(252 页)。

13.4.1 线的 FitType

对于有绘制线的图表(XY-Plot， Area， Polar 和 Radar)。FitType 属性设置线如何处理图表上的各个点。该属性使用一个 FitTypeEnum 枚举。

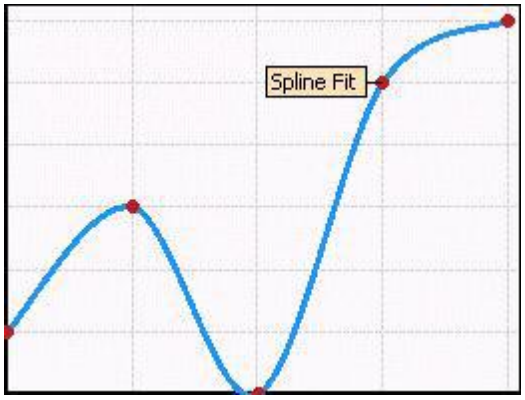
FitTypeEnum.Line

Line 适应是绘制线的默认方式，线直接穿过各个点。Line 的图示如下:



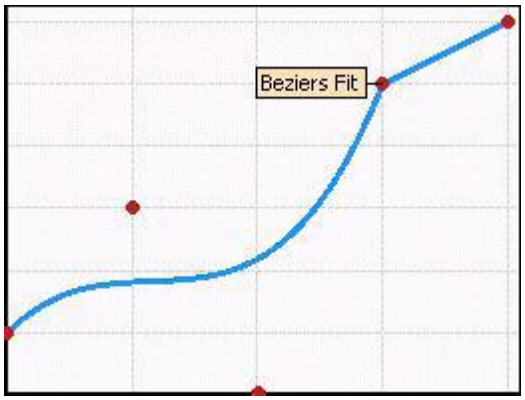
FitTypeEnum.Spline

Spline 方式是以曲线的形式来穿过各个点，不过是以平滑弯曲的方式来绘制线。熟悉如何在大多数的应用程序中创建曲线的很可能对 Spline 也会感到很熟悉。



FitTypeEnum.Beziers

Beziers 方式的做法是线不必穿过每一个点，但是点像磁铁一样，向特定方向牵引线的弯曲，而且影响曲线的弯曲方式。



13.5 图表边框

给图表的部分内容加上边框可以强调重要的信息，或者简单地使图表看起来更吸引人。以下图表元素的边框类型和宽度可以自定义:

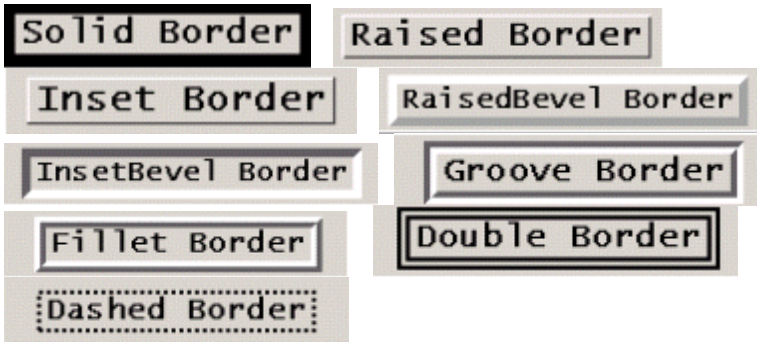
表头和页尾标题

图例

图表区

图表整体

下图展示了所有合法的边框类型，同时还可以指定没有边框。



改变边框

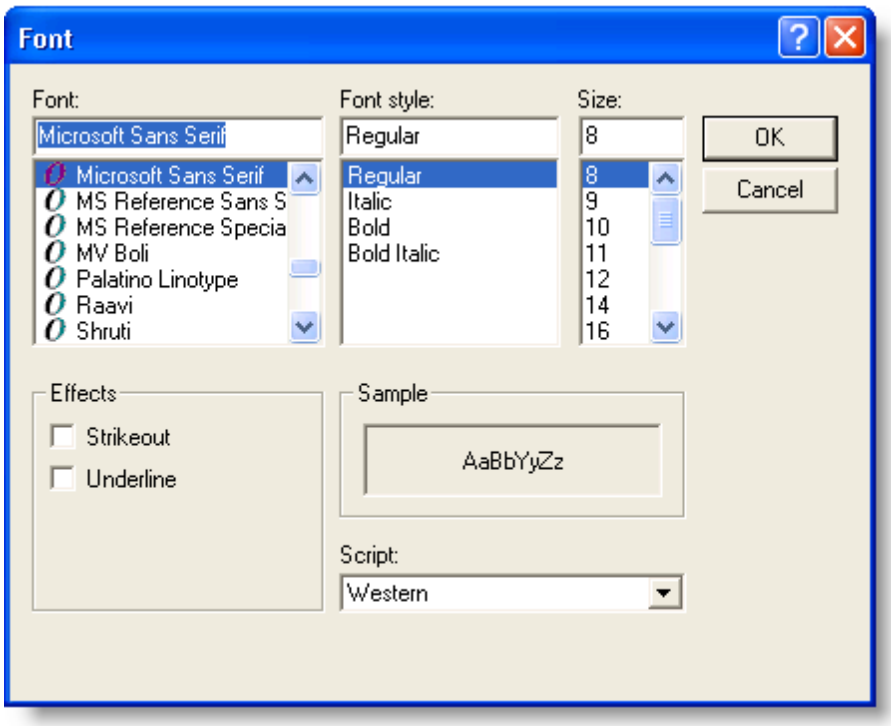
使用 Type 属性来改变边框的类型，使用 **Width** 属性来改变边框的厚度。 可以通过 Visual Studio 的属性窗口中 Control， ChartArea， Titles， Legend， 和 ChartLabels 对象中的 Style 节点中找到这些属性。

13.6 图表字体

通过自定义不同图表元素的字体可以使图表看起来更有冲击力。可以调整一个元素的字体大小来让它更加地适应图表的全局大小。

改变字体

使用.NET 标准的字体属性编辑器来设置字体 样式和大小属性。字体属性位于 Visual Studio 的属性窗口中的 Style 节点下。



13.7 图表颜色

颜色可以增加一个图表的视觉效果。颜色可以通过颜色名称，选择一个颜色主题，使用 RGB 值，或者使用交互式的颜色选择器来进行自定义。以下所列的图表中的可视元素都含有一个背景色和前景色可以来自定义设置。

- 图表整体
- 数据序列
- 表头和页尾标题
- 图例
- 图表区
- 每一个添加到图表的图表标签。

13.7.1 交互式地选择颜色

颜色可以使用交互式的.NET 颜色对话框来进行设置，该对话的功能与框标准 Windows 颜色对话框一样。可以从 Windows 基本色，自定义颜色，或者交互式地从全部颜色光谱中选择。

13.7.2 为数据序列设置颜色主题

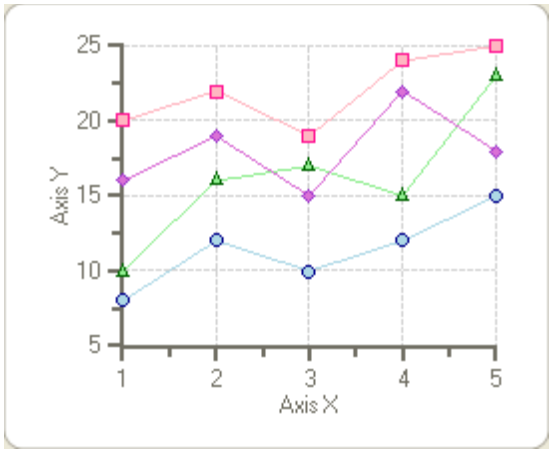
可以通过 ColorGeneration 属性来选择数据序列的颜色主题。默认情况下， C1Chart 使用 ColorGeneration.Custom 设置指定标准的颜色生成方式。其余的选项模拟 Microsoft Office 的颜色主题。

以下是可用的数据序列的颜色主题:

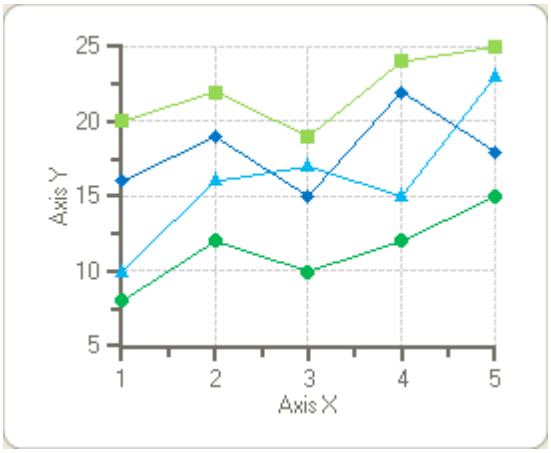
颜色生成设置 描述或者效果预览

CopyCurrentToCustom 拷贝现有的特定颜色组到自定义 group.*

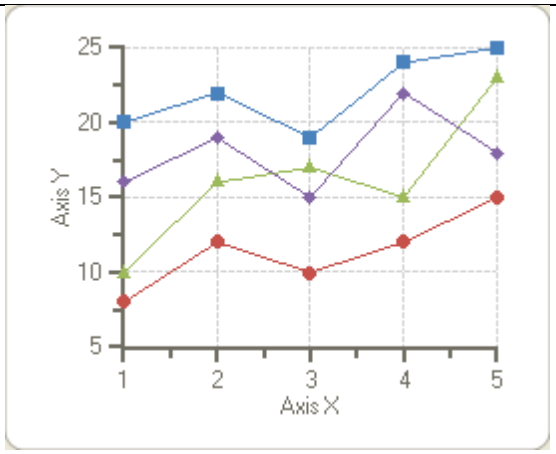
Custom (default)



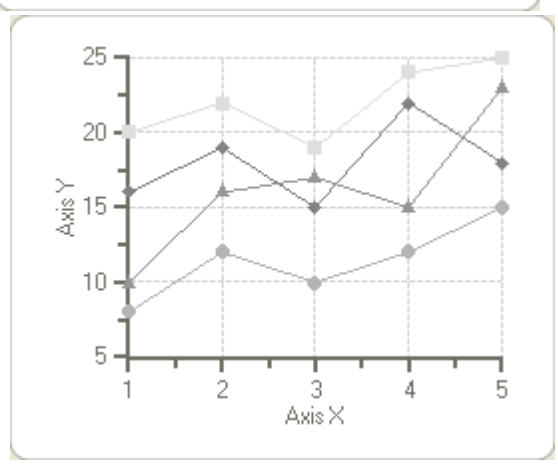
Standard



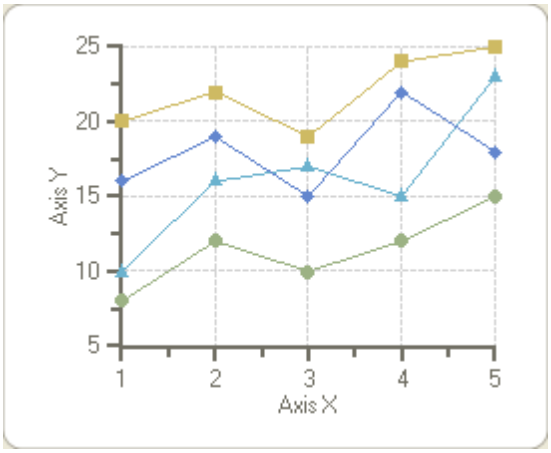
Office



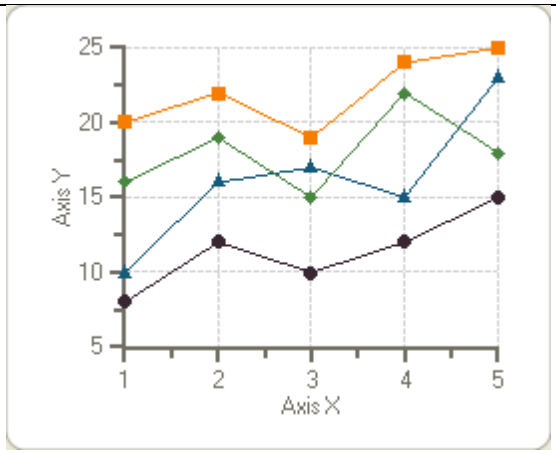
GrayScale



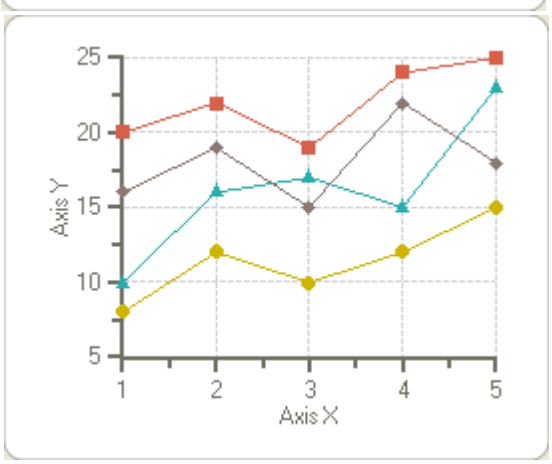
Apex



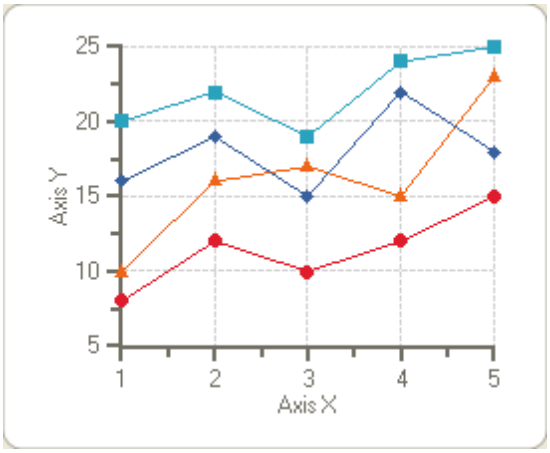
Aspect

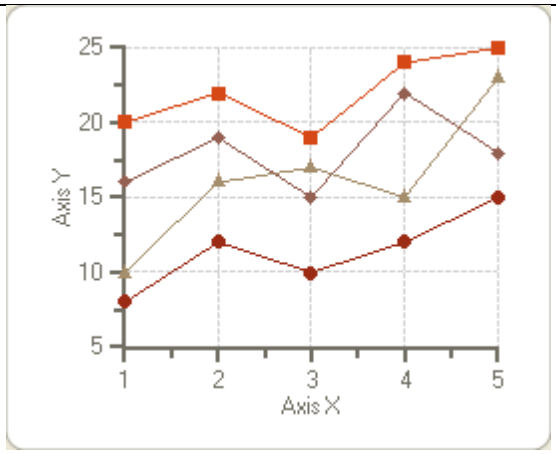


Civic

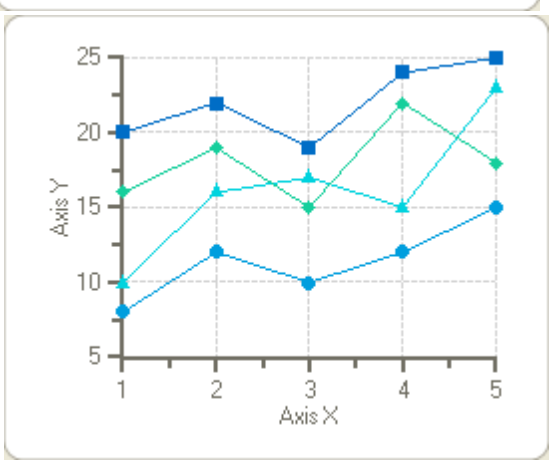


Concourse

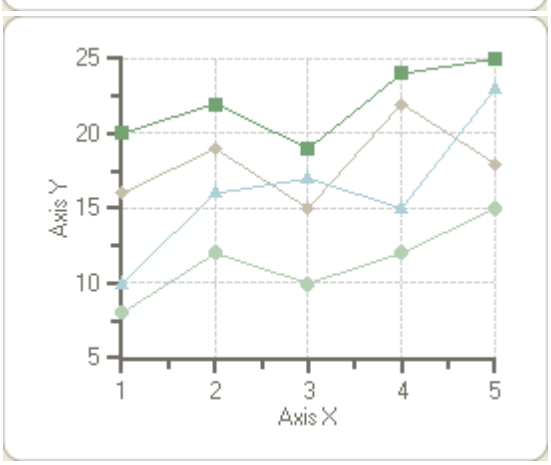




Equity

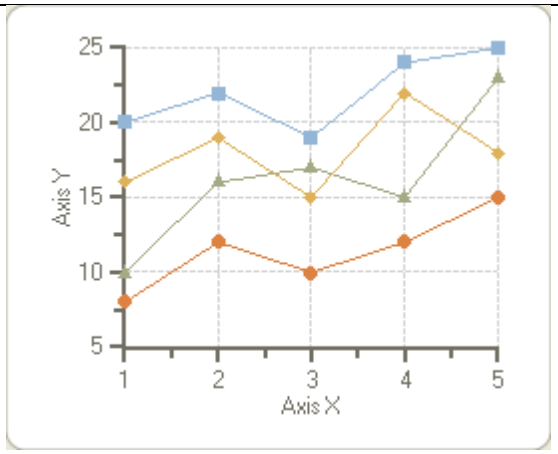


Flow

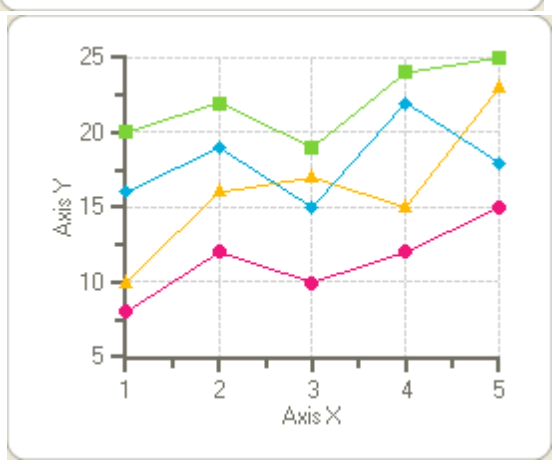


Foundry

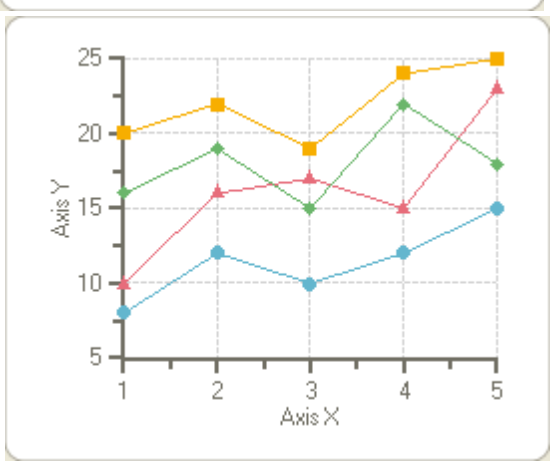
Median



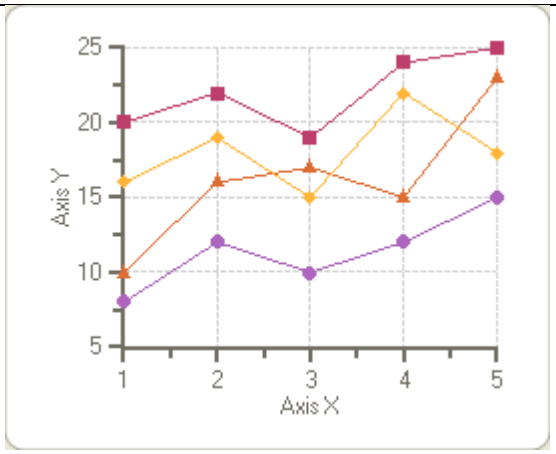
Metro



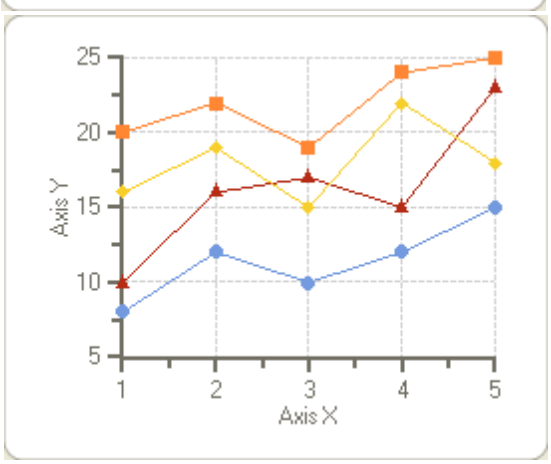
Module



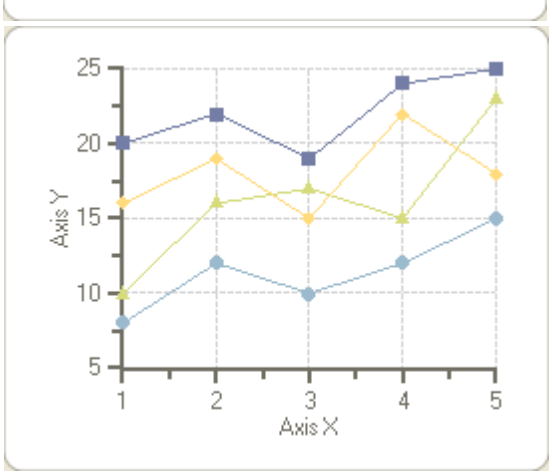
Opulent



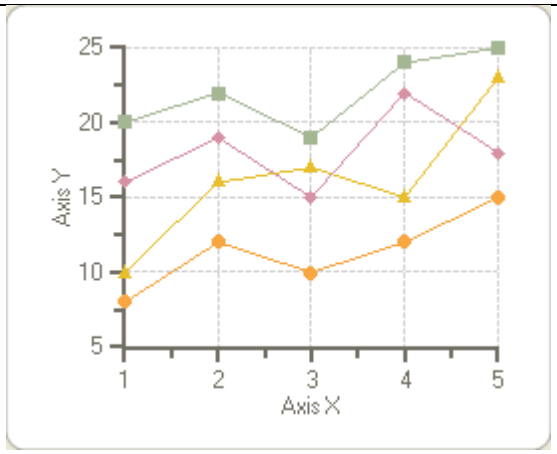
Oriel



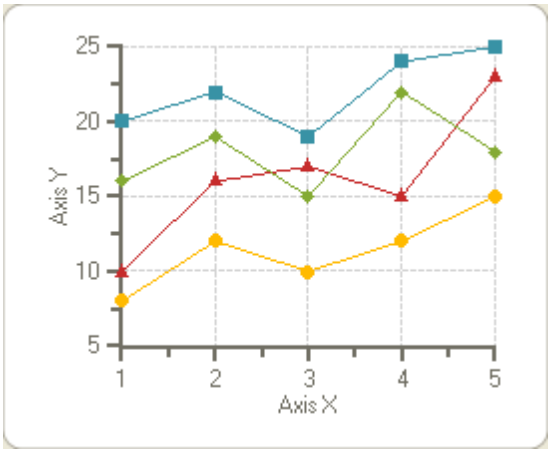
Origin



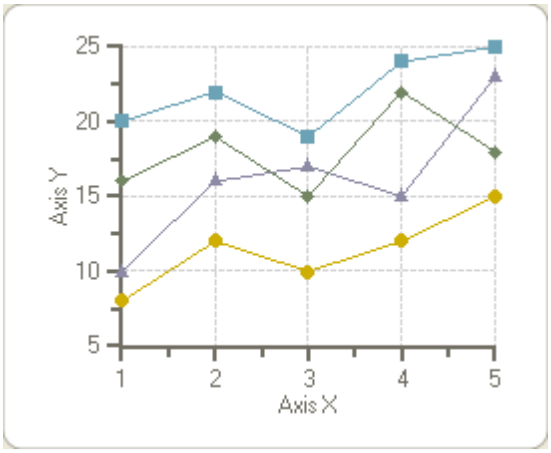
Paper

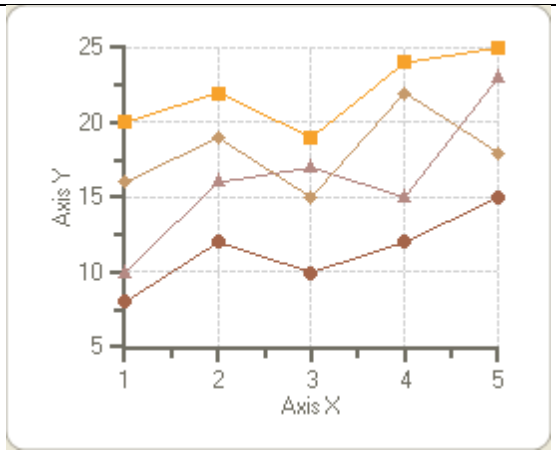


Solstice

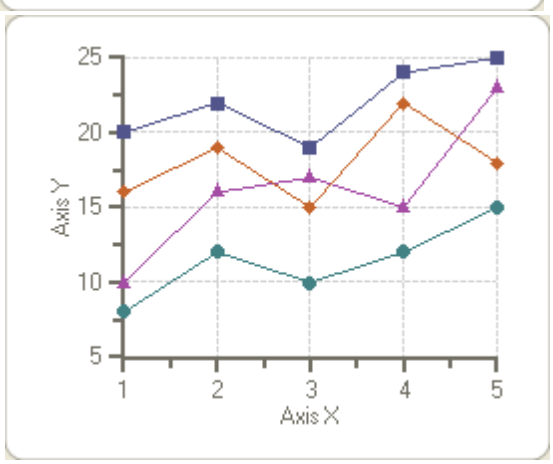


Technic

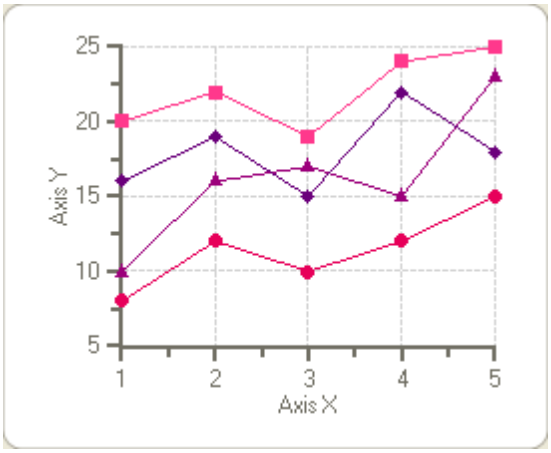




Trek



Urban



Verve

请注意当 ColorGeneration 属性值设置为 **CopyToCustom** 时，当前颜色被拷贝到自定义颜色生成上。而且该属性被自动的修改为 **ColorGeneration.Custom** 以进行进一步的自定义设置。并且请注意该操作无论图表数据是由字符串，文件，还是由设计器加载的，都会自动完成。由于颜色定制是用于每一个图表数据序列的 LineStyle 和 SymbolStyle 上的，所以这些定制会影响当前状态和自定义状态的值。

13.7.3 指定 RGB 颜色

或者，颜色也可以使用 RGB 组件来指定，当匹配另一种 RGB 颜色时这样做将非常有用。RGB 颜色的值是用 16 进制数字表示颜色的红，绿，蓝部分。”00”是一个颜色部分的最小值，”ff”

是一个颜色部分的最大值。例如，”#ff00ff”代表紫红色(红色和蓝色部分的最大值并且没有绿色部分)。

13.7.4 指定色调，饱和度和明暗度

颜色不仅由 RGB 部分组成，而且还可以使用色调，饱和度，明暗度来标示。色调，饱和度和明暗度对红，绿，蓝三种颜色部分都起效。色调是由红，绿，蓝色调组成的色轮中的一个特殊色调。饱和度是一个色调从灰色基调到纯色的色调的明暗度，亮度是一个色调的亮度或者暗度。

C1Chart 含有一个视觉效果设计器来让你指定数据序列，表头和页尾标题，和图例元素的色调，饱和度和明暗度。关于使用**视觉效果**设计器指定一种新颜色的色调，饱和度，明暗度的更多信息，请参见**使用颜色游标来增加图表数据序列中的既存颜色**(319 页)或者**图表表头和页尾@使用颜色游标来增加图表表头和页尾的既存颜色**。

13.7.5 使用透明色

除了图表本身外的所有元素的背景色和前景色都可以被设置为”透明色”。

当背景色或者前景色被设置为透明色时，图表使用元素的容器的颜色作为元素的背景色。例如，当表头的背景色被设置为透明色时，其使用图表本身的颜色作为自己的背景色。

换句话说，当一个元素的背景色被设置为透明色时，其背景不会被绘制。当一个元素的前景色被设置为透明色时，前景(例如，标题的文字)不会被绘制出来。

可以通过 Visual Studio 的属性窗口中 Control， ChartArea， Titles， Legend， 和 ChartLabels 对象中的 Style 节点中找到透明色属性。

13.8 图表元素的位置和大小

每一个主要的图表元素(Header， Footer， Legend， ChartArea， 和 ChartLabels)都含有能控制自身位置和大小的属性。另一方面，图表可以自动地控制除了图表标签之外的元素的位置，还可以自动控制任何元素的大小。这些属性都可以被自定义。

当图表控制位置时，会为表头，页尾和图例预留存在的空间(大小是由内容，边框和字体决定的)。 ChartArea 元素的大小和位置会被设置为占据最大剩余区域的矩形。当图表的其它属性被修改时，位置会被重新调整。

13.8.1 改变位置

使用 Location 的 X 属性来指定从图表的边界到图表元素的左边界的像素数。使用 Y 属性来指定从图表的边界到图表元素的顶部的像素数。设置 X 和 Y 属性值为-1 来让图表自动定位元素的位置。

可以通过 Visual Studio 的属性窗口中 ChartArea， Titles， Legend， 和 ChartLabels 节点中的 Location 节点找到这些属性。

13.8.2 改变宽度和高度

使用 Location 属性的 Width 和 Height 来指定图表元素的宽度和高度。设置这两个属性值为-1 可以让图表自动地给图表元素设置大小。

这些节点位于 Visual Studio 的属性窗口中的 Location 节点，并且在设计时可用。

13.9 为数据绘制自定义画笔

该部分描述了在绘制过程中使用 **DrawDataSeriesEventArgs** 事件来创建自定义画笔的信息。

每当一个数据序列被绘制时，都会有一个 C1Chart 的 DrawDataSeries 事件被触发。这个事件让我们在绘制时进行自定义画笔的选择。该事件的发送方对象是将要被绘制的 C1Chart。ChartDataSeries。事件数据(DrawDataSeriesEventArgs)含有以下属性:

- Brush 属性定义了绘制数据序列将使用的画笔。当你想自定义画笔时，你需要创建一个你自己的画笔，并将该画笔赋给 Brush 属性。
- 布尔类型的 DisposeBrush 属性值指定了当画笔使用完后是否需要被释放。当该属性被设置为 False 时，你必须自己负责释放画笔。
- GroupIndex 属性设置图表组的索引。
- SeriesIndex 属性设置图表组的索引。
- Bounds 属性代表数据将被绘制的矩形区域。当创建渐变的画笔时该矩形非常有用。

13.9.1 创建阴影画笔

下面的简单代码片段代表了创建阴影画笔的处理程序。

请注意以下给出的代码片段中一些对象的初始化声明中使用了全命名空间。在其后这些对象的使用过程中为了简洁而省略了全命名空间的写法。

- Visual Basic

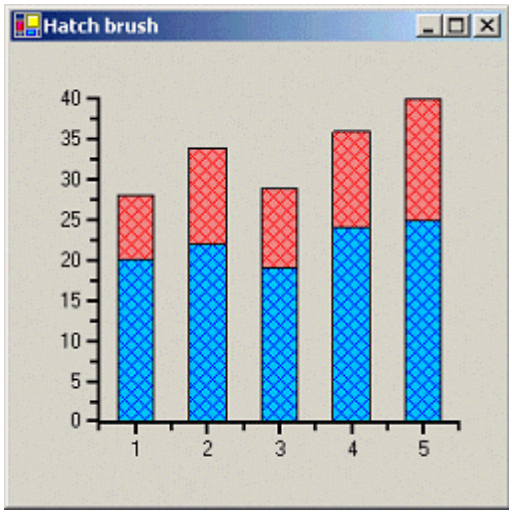
```
    Private Sub C1Chart1_DrawDataSeries(ByVal sender As Object, _ ByVal e As
C1.Win.C1Chart.DrawDataSeriesEventArgs) _ Handles      C1Chart1.DrawDataSeries
    Dim ds As C1.Win.C1Chart.ChartDataSeries = sender
    Dim foreclr As Color = ds.SymbolStyle.Color
    Dim backclr As Color = ds.LineStyle.Color
    Dim hb As System.Drawing.Drawing2D.HatchBrush
    hb = New HatchBrush(HatchStyle.OutlinedDiamond, foreclr, backclr)
    e.Brush = hb
    End Sub
```

- C#

```
    private void c1Chart1_DrawDataSeries(object sender, C1.Win.C1Chart.DrawDataSeriesEventArgs
e)
    {
    C1.Win.C1Chart.ChartDataSeries ds = (ChartDataSeries)sender;
    Color forecolor = ds.SymbolStyle.Color;
    Color backcolor = ds.LineStyle.Color;

    System.Drawing.Drawing2D.HatchBrush hb;
    hb = new HatchBrush(HatchStyle.OutlinedDiamond, forecolor, backcolor);
    e.Brush = hb;
    }
```

下图展示了阴影画笔的柱状图。



13.9.2 创建渐变画笔

下面的简单代码片段代表了创建线性渐变画笔的处理程序。

请注意以下给出的代码片段中一些对象的初始化声明中使用了全命名空间。在其后这些对象的使用过程中为了简洁而省略了全命名空间的写法。

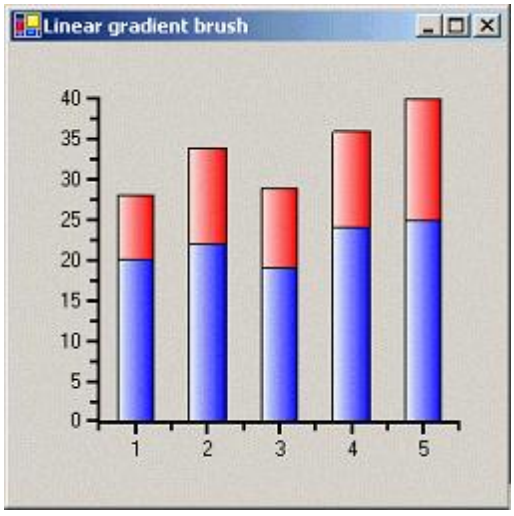
- Visual Basic

```
    Private Sub C1Chart1_DrawDataSeries(ByVal sender As Object, _ ByVal e As
C1.Win.C1Chart.DrawDataSeriesEventArgs) _ Handles C1Chart1.DrawDataSeries
    Dim ds As C1.Win.C1Chart.ChartDataSeries = sender
    Dim clr1 As Color = ds.LineStyle.Color
    Dim clr2 As Color = ds.SymbolStyle.Color
    If(e.Bounds.Height > 0 And e.Bounds.Width > 0) Then
    Dim lgb As System.Drawing.Drawing2D.LinearGradientBrush = _ New
LinearGradientBrush(e.Bounds, clr1, clr2, LinearGradientMode.Horizontal)
    e.Brush = lgb
    End If
    End Sub
```

- C#

```
    private void c1Chart1_DrawDataSeries(object sender,
    C1.Win.C1Chart.DrawDataSeriesEventArgs e)
    {
    C1.Win.C1Chart.ChartDataSeries ds = (ChartDataSeries)sender;
    Color clr1 = ds.LineStyle.Color;
    Color clr2 = ds.SymbolStyle.Color;
    if(e.Bounds.Size.Height > 0 && e.Bounds.Size.Width > 0)
    {
    System.Drawing.Drawing2D.LinearGradientBrush lgb = new
LinearGradientBrush(e.Bounds,clr1,clr2,inearGradientMode.Horizontal);
    e.Brush = lgb;
    }
    }
```

下图展示了线性渐变画笔。



14. 加载和保存图表，数据，图片

在 C1Chart 里每一个图表都可以使用外部 XML 文件 或者一个字符串值来进行加载和保存。另外，图表可以被保存到任意数量的图片文件里，这使得在对图表进行提取，访问数据，创建图表操作时有更大的灵活性。

有两种不同的方式可以被用来加载/保存图表到 XML 文件或者字符串值。一种是将整个图表(数据和格式)进行加载/保存，另一种是仅加载/保存图表的数据。

加载/保存图表的样式和数据

当整个图表被加载/保存时，所有的轴线的格式，标题，和图表区域都会被保存到一个 XML 文件里。

以下的方法被用来保存图表的格式和数据:

- LoadChartFromFile
- SaveChartToFile
- LoadChartFromString
- SaveChartToString

当你需要保存或者加载诸如图表背景，表头和页尾图片等的图表元素的背景图片时，可以使用以下保存/加载方法:

- LoadChartAndImagesFromFile
- SaveChartAndImagesToFile
- LoadChartAndImagesFromString
- SaveChartAndImagesToString

保存/加载图片的数据

只有当图表的数据被保存/加载到尺寸较大的外部文件，在文件被加载回图表时，只有数据被重置。

以下方法被用来仅保存/加载图表的数据:

- SaveDataToFile
- LoadDataFromFile

14.1 保存和加载字符串

除了保存和加载到一个 XML 文件，图表还能保存到一个字符串值中。这个字符串能被保存到应用程序中并在将来需要时重新得到。

C1Chart 提供了 4 种将图表保存和加载到字符串的方法:SaveChartToString，LoadChartFromString，SaveChartAndImagesToString，和 LoadChartAndImagesFromString。

SaveChartToString 和 LoadChartFromString 方法让你能够将图表的一个备份保存到应用程序中。这个属性值含有图表的所有属性设置，方法，事件，一言以蔽之，是图表的格式。图表可以保存到一个文件中，然后如果结构不需要不变，从这个文件中加载一个图表。

SaveChartAndImagesToString 和 LoadChartAndImagesFromString 方法跟 SaveChartToString 和 LoadChartFromString 方法的功能一致 除了它们会记录图表元素(图表，表头，页尾，标签)的各种各样的背景图片外。在运行时需要保存图表到一个字符串值时，调用 SaveChartToString 方法，该方法没有参数且返回一个字符串值。

- Visual Basic

```
Dim ChartString As String
ChartString = C1Chart1.SaveChartToString()
```

- C#

```
string ChartString;
ChartString = c1Chart1.SaveChartToString();
```

在运行时调用 LoadChartFromString 方法来从字符串值中加载一个图表，该方法仅需要一个字符串参数。

- Visual Basic

```
C1Chart1.LoadChartFromString(ChartString)
```

- C#

```
c1Chart1.LoadChartFromString(ChartString);
```

14.2 加载和保存图到一个文件中

在 C1Chart 中，每一个图都可以被加载和保存到外部的 XML 文件中，可以通过在设计时使用 C1Chart 的上下文菜单中的 Save Chart/Load Chart 命令或者在以编程的方式使用以下方法:

- LoadChartFromFile
- SaveChartToFile
- LoadChartAndImagesFromFile
- SaveChartAndImagesToFile

SaveChartAndImagesToFile 和 LoadChartAndImagesFromFile 方法与 SaveChartToFile 和 LoadChartFromFile 的效果一样， 除了它们会记录图表元素(图表，表头，页尾，标签)的各种各样的背景图片外。

如果你仅想保存图的数据，使用 ChartData 对象的 SaveDataToFile 和 LoadDataFromFile 方法。

如果仅仅想保存图的数据到 XML 中，调用 SaveDataToFile 方法，该方法仅有一个文件路径参数。

- Visual Basic

```
C1Chart1.ChartGroups.Group0.ChartData_
.SaveDataToFile("C:\ComponentOneDocs\chartdata1.xml")
```

- C#

```
c1Chart1.ChartGroups.Group0.ChartData
.SaveDataToFile("C:\\ComponentOneDocs\\chartdata1.xml");
```

当需要从 XML 中加载图的数据时，调用 LoadDataFromFile，该方法仅有一个文件路径参数。

- Visual Basic

```
C1Chart1.ChartGroups.Group0.ChartData_
.LoadDataFromFile("C:\ComponentOneDocs\chartdata1.xml")
```

- C#

```
c1Chart1.ChartGroups.Group0.ChartData
.LoadDataFromFile("C:\\ComponentOneDocs\\chartdata1.xml");
```

关于XML文件结构的深层讨论不在本文档的讨论范围内，但是它可以作为文本文件打开并很容易分析。

14.3 保存图表图片

C1Chart 提供了为整体图表创建一个单独的图片然后绘制在屏幕或者打印机上的能力。调用 SaveImage 方法后，所有在图表范围内的元素都被保存到粘贴板，字节数组，流，或者图片文件中。

SaveImage 方法提供了八种不同的参数重载方式以能够保存到四种不同的输出类型中。对于每一种输出类型 ,有一个选项可以是指定按照图表在屏幕上的显示尺寸来保存图片还是保存为另一个特定的尺寸的图片。

当要保存图表图片到粘贴板时，需要指定图片格式，但是不用指定尺寸参数。

- Visual Basic

```
C1Chart1.SaveImage(System.Drawing.Imaging.ImageFormat.Bmp)
```

- C#

```
c1Chart1.SaveImage(System.Drawing.Imaging.ImageFormat.Bmp);
```

当要保存图表图片到一个图片文件时 ,指定新图片的路径和图片格式 ,但是不用指定尺寸参数。

- Visual Basic

```
C1Chart1.SaveImage("C:\temp\ChartImages\CandleChart.bmp",_
System.Drawing.Imaging.ImageFormat.Bmp)
```

- C#

```
c1Chart1.SaveImage("C:\\temp\\ChartImages\\CandleChart.bmp",
System.Drawing.Imaging.ImageFormat.Bmp);
```

当要保存图表图片到一个流中时，指定流对象和图片格式，但是不用指定尺寸参数。

- Visual Basic

```
Dim coutstream As New System.IO.MemoryStream()
C1Chart1.SaveImage(coutstream, System.Drawing.Imaging.ImageFormat.Bmp)
```

- C#

```
System.IO.MemoryStream coutstream = new System.IO.MemoryStream();
c1Chart1.SaveImage(coutstream, System.Drawing.Imaging.ImageFormat.Bmp);
```

当要保存图表图片当一个字节数组时，指定字节数组对象和图片格式，但是不用指定尺寸参数。

- Visual Basic

```
Dim bytes() As Byte
C1Chart1.SaveImage(bytes, System.Drawing.Imaging.ImageFormat.Bmp)
```

- C#

```
Byte[] bytes;
c1Chart1.SaveImage(bytes, System.Drawing.Imaging.ImageFormat.Bmp);
```

15. 终端用户交互

有很多本文档没有描述的但是可以执行的操作。每一个图表控件的用户都有自己不同的功能和特性的要求。一种能够适应多种功能需求的做法是提供一系列的能够创建控件中不存在的功能的工具。

在 C1Chart 中，有一系列的内置工具集，例如转换方法和诸如旋转，缩放，缩放图表等交互式的内置工具，这些工具能帮助我们进行定制并且更深一步的开发应用程序。

以下的章节描述了如何使用不同类型的坐标转换方法和内置工具集来进行运行时交互。

15.1 坐标转换方法

诸如 **MouseMove** 等的.Net 事件记录了图表区域的坐标值并提供了像素坐标，它在某个层上非常有用，但是它不会描述任何关于底层数据坐标，数据点，或者图表区域的任何信息。

图表的转换方法处理当终端用户的光标通过了图表并执行了某种操作的场景。

C1Chart 的坐标转换方法从像素坐标中提取了上述的信息。一旦你知道了像素坐标，你就可以调用任何下表所示的任何 C1Chart 中的坐标转换方法。来获取特定序列，数据坐标，数据索引的数据值。

像素到数据坐标转换方法	描述
SeriesFromCoord	获取离特定工作区坐标最近的图例项的组和系列指数。鼠标坐标在工作区坐标中指定。
CoordToDataCoord	计算代码所指定的图表工作区坐标内的一个点的数据坐标。
CoordToDataIndex	返回序列，点指数，到组指定的工作区坐标最近的数据点的距离。

如果你知道数据坐标或者数据指数，但是不知道像素坐标，你可以使用下述的方法来将数据坐标或者数据指数转换为像素坐标。

数据到像素坐标转换方法	描述
DataIndexToCoord	计算代码所指定的数据坐标内的一个点的工作区坐

	标。
DataCoordToCoord	返回指定数据点的工作区坐标

当和.Net 的 MouseMove 事件一起使用时 ,这些工具能够创建有趣的应用程序—处理双击图例和图表提示的特性。.Net 中的 MouseMove 事件提供了用户的光标 ,当光标拖到图表上并且不断地更新用户光标所在位置的数据。

本文当的下面几个章节简要地描述了如何使用每一种的坐标转换方法 ,并且为每一种方法提供了示例。

15.1.1 转换坐标到序列

C1Chart 图例中，序列使用序列名称和一个含有 SymbolStyle 或者 LineStyle 的样式的符号来代表。 图例的 SeriesFromCoord 方法返回到图例中的一个像素坐标值的最近的关联序列。例如，假设用户在图例上点击第一个序列的符号，此刻将用户鼠标的像素坐标作为参数传入 SeriesFromCoord 方法，将会从 Group 和 Series 两个 ref 参数中得到组和序列的值。以下是前述的示例:

- Visual Basic

```
Dim LegendGroup As Integer
Dim LegendSeries As Integer
C1Chart1.Legend.SeriesFromCoord(170, 275, LegendGroup, LegendSeries)
```

- C#

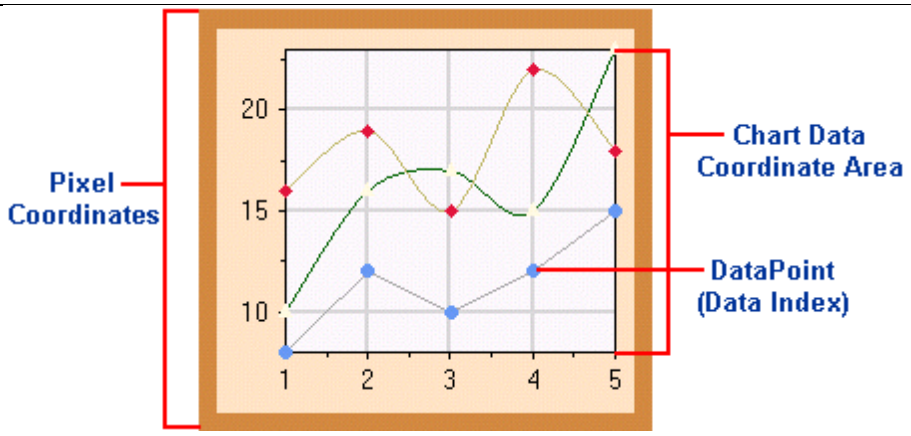
```
int LegendGroup = 0;
int LegendSeries = 0;
c1Chart1.Legend.SeriesFromCoord(170, 275, ref LegendGroup, ref LegendSeries);
```

已提供示例工程

如果想获取使用该方法的完整实例，参考 http://helpcentral.componentone.com/Samples.aspx 中的 DataStyl 或者 PieStuff 示例。

15.1.2 转换像素坐标到数据点和反向操作。

C1Chart 同样允许将像素坐标转化成它们关联的数据坐标或者数据索引 ,反向操作也是支持的。进行这些操作的方法有 CoordToDataCoord 方法，CoordToDataIndex 方法，DataCoordToCoord 方法和 DataIndexToCoord 方法。下图展示了在 **C1Chart** 中那些区域是使用像素坐标控制，那些区域是使用数据坐标控制(代码)



15.1.2.1 转换像素坐标成数据点

ChartGroup 对象的 CoordToDataCoord 和 CoordToDataIndex 两个方法用来把像素坐标转换成数据坐标或者数据点。这两个方法都有一个很可能来自于 MouseMove 事件的坐标值作为参数，然后返回代码坐标或者最近的数据点。

CoordToDataCoord 方法

CoordToDataCoord 方法有四个参数，前两个参数(示例程序中的 e.X 和 e.Y)是从 MouseMove 事件中拿到的像素坐标。后两个整数型参数是方法用来填充 X 和 Y 数据坐标的数据。下面的示例程序时在 MouseMove 事件中使用该方法的例子。

- Visual Basic

```
Dim CoordXOutput As Double
Dim CoordYOutput As Double
C1Chart1.ChartGroups.Group0.CoordToDataCoord(e.X, e.Y, CoordXOutput, CoordYOutput)
Debug.WriteLine("X Data Coordinate: " & CoordXOutput.ToString())
Debug.WriteLine("Y Data Coordinate: " & CoordYOutput.ToString())
```

- C#

```
double CoordXOutput = 0;
double CoordYOutput = 0;
c1Chart1.ChartGroups.Group0.CoordToDataCoord(e.X, e.Y, CoordXOutput, CoordYOutput);
c1Chart1.ChartGroups.Group0.
CoordToDataCoord(e.X, e.Y, ref CoordXOutput, ref CoordYOutput);
ConsoleDebug.Writeline("X Data Coordinate: " + CoordXOutput.ToString());
ConsoleDebug.Writeline("Y Data Coordinate: " + CoordYOutput.ToString());
```

已提供示例工程

如果想获取使用CoordToDataCoord方法的完整实例，参考 http://helpcentral.componentone.com/Samples.aspx中的StepChart或CoordToMapping3D示例。

CoordToDataIndex 方法

CoordToDataIndex 方法有六个参数，前两个参数(示例程序中的 e.X 和 e.Y)是从 MouseMove 事件中拿到的像素坐标。第三个参数是一个 CoordinateFocusEnum 枚举值。这个枚举指定了当要决定哪个点更近一些和到像素坐标的距离的时候以哪个轴线为焦点。例如，如果选择以 X 为焦点，那么会返回相对于 X 的最近点，而无视相对于 Y 的值。第四和第五两个整数型参数是方法用来填充对应的数据坐标的数据。第六个整数型参数是方法用来填充从指定的像素坐标到数据点的以像素为单位的距离值。下面的示例程序时在 MouseMove 事件中使用该方法的例子。

- Visual Basic

```
Dim SeriesOutput As Integer
Dim PointOutput As Integer
Dim DistanceOutput As Integer
C1Chart1.ChartGroups.Group0._
CoordToDataIndex(e.X, e.Y, CoordinateFocusEnum.XandYCoord,_ ref SeriesOutput, ref PointOutput, ref DistanceOutput)
Debug.WriteLine("Series Index: " & SeriesOutput.ToString())
Debug.WriteLine("Point Index: " & PointOutput.ToString())
Debug.WriteLine("Distance From Point: " & DistanceOutput.ToString())
```

- C#

```
int SeriesOutput = 0;
int PointOutput = 0;
int DistanceOutput = 0;
c1Chart1.ChartGroups.Group0.CoordToDataIndex(e.X, e.Y,
CoordinateFocusEnum.XandYCoord,
ref SeriesOutput, ref PointOutput, ref DistanceOutput);
ConsoleDebug.Writeline("Series Index: " + SeriesOutput.ToString());
ConsoleDebug.Writeline("Point Index: " + PointOutput.ToString());
ConsoleDebug.Writeline("Distance From Point: " + DistanceOutput.ToString());
```

已提供示例工程

如果想获取使用 CoordToDataIndex 方法的完整实例，参考 http://helpcentral.componentone.com/Samples.aspx 中的 DataStyl， PieStuff， StepChart 或者 Scatter 示例。

15.1.2.2 转换数据点到像素坐标

ChartGroup 对象的 DataIndexToCoord 和 DataCoordToCoord 方法的作用是转换数据坐标或者数据索引到像素坐标。DataIndexToCoord 方法以一个序列和点索引为参数并且返回指定数据点的工作区坐标。 DataCoordToCoord 方法以一个数据坐标集为参数并返回一个像素坐标。这两个方法与其它的转换方法非常相似。

DataIndexToCoord 方法

DataIndexToCoord 方法有四个参数。并且返回一个像素坐标。请看下面的例子:

- Visual Basic

```
Dim CoordX, CoordY As Integer
C1Chart1.ChartGroups.Group0.
DataIndexToCoord(ChartSeries, ChartPoint, CoordX, CoordY)
Debug.WriteLine("X Chart Coordinate: " & CoordX.ToString())
Debug.WriteLine("Y Chart Coordinate: " & CoordY.ToString())
```

- C#

```
int CoordX=0, CoordY=0;
c1Chart1.ChartGroups.Group0.
DataIndexToCoord(ChartSeries, ChartPoint, ref CoordX, ref CoordY);
Debug.Writeline("X Chart Coordinate: " + CoordX.ToString());
Debug.Writeline("Y Chart Coordinate: " + CoordY.ToString());
```

DataCoordToCoord 方法

DataCoordToCoord 方法有四个参数。并且返回一个像素坐标。请看下面的例子:

- Visual Basic

```
Dim CoordX, CoordY As Integer
C1Chart1.ChartGroups.Group0._
DataCoordToCoord(DataCordX, DataCoordY, CoordX, CoordY)
Debug.WriteLine("X Chart Coordinate: " & CoordX.ToString())
Debug.WriteLine("Y Chart Coordinate: " & CoordY.ToString())
```

- C#

```
int CoordX=0, CoordY=0;
c1Chart1.ChartGroups.Group0.
DataCoordToCoord(DataCordX, DataCoordY, ref CoordX, ref CoordY);
ConsoleDebug.Writeline("X Chart Coordinate: " + CoordX.ToString());
ConsoleDebug.Writeline("Y Chart Coordinate: " + CoordY.ToString());
```

已提供示例工程

如果想获取使用 DataCoordToCoord 方法的完整实例，参考 http://helpcentral.componentone.com/Samples.aspx 中的 PropGrid 或者 FExplorer 示例。

15.2 旋转，缩放(Scale)，平移和缩放(Zoom)

C1Chart 含有一些能够让与终端用户的交互变得更容易实现的内置工具。终端用户可以结合鼠标和 Shift 按键来对图表进行浏览，旋转和缩放操作。除了提供给终端用户的内置工具外，在 C1Chart 里对于终端用户来说修改任何诸如轴标签等的图表元素也是可行的。可以通过设置 C1Chart 对象的 Microsoft 属性格或者使用 ShowProperties 方法来实现上述功能。

终端用户交互

C1Chart 含有一些能够让与终端用户的交互变得更容易实现的内置工具。终端用户可以结合鼠标和 Shift 按键来对图表进行浏览，旋转和缩放操作。

C1Chart 里的用户交互特性的控制中心是 Interaction 对象。 Interaction 对象含有很多能够自定义界面的属性。所有这些属性都可以在设计时通过属性窗口或者 **Action Collection Editor** 来进行设置和修改，或者在代码中使用 Interaction 类来进行设置和修改动作。关于 **Action Collection Editor** 的更多信息，请参见**动作集合编辑器**(39 页)

Interaction 类含有以下的属性:

- IsDefault 是一个用来指示动作设定是否是默认的布尔类型属性。如果要恢复默认，将 IsDefault 属性设置为 True。
- Enabled 属性用来开启/关闭交互。请注意默认交互是禁用的。
- Actions 包含一系列图表可用的动作。
- 下面展示了所有支持的动作:
- 旋转动作可以改变视角，这个动作仅当图表有 3D 效果时才可用。
- 缩放(Scale)动作根据选中的横坐标或者纵坐标来增加或者减少图表的尺寸。
- 平移动作提供了在绘制区域内滚动的机会。
- 缩放(Zoom)动作允许用户选中矩形区域来查看。
- 缩放(Scale)，平移，缩放(Zoom)仅对笛卡尔坐标轴的图表才可用。
- 每一个动作对象都提供了一系列进行动作的自定义行为的属性:
- Name 属性获取动作的名称。
- Axis 属性指定动作涉及的轴线。可能对一个动作只使用一个轴线。例如，缩放(Scale)或者平移可以被应用在横向(x-轴)或者纵向(y-轴)方向。
- MouseButton 和 Modifier 属性指定可以出发动作执行的鼠标按钮和按键(Alt，Control 或者 Shift)的组合。

15.2.1 控制转换

应用到图表上的转换都可以通过 C1Chart 的 Transform 事件来监听到。当缩放(Scale)，平移，缩放(Zoom)发生时， Transform 时间被触发。对该事件进行处理可以限制轴线的边界，或者取消动作。

下面的示例通过当轴线超出最大或者最小范围时取消动作来限制 x-轴线的边界为从-10 到 10。

- Visual Basic

```
    Private Sub C1Chart1_Transform(ByVal sender As Object, _ ByVal e As
C1.Win.C1Chart.TransformEventArgs) Handles C1Chart1.Transform
    If e.MinX < -10 Or e.MaxX > 10 Then
    e.Cancel = True
    End If
    End Sub
```

- C#

```
private void c1Chart1_Transform(object sender,C1.Win.C1Chart.TransformEventArgs e)
{
if( e.MinX < -10 || e.MaxX > 10)
e.Cancel = true;
}
```

15.2.2 创建一个缩放(Zoom)效果

通过简单地调整轴线，可以为图表创建一个缩放(Zoom)的效果。例如，你可以在你的应用程序中添加两个按钮(**Zoom in** 和 **Zoom out** 按钮)来在运行时根据按钮的点击来调整你的轴线。这里是一份演示了如何处理 **Button_Click** 事件来控制缩放(Zoom)的代码。

- Visual Basic

```
' Controls zooming in
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim xMin As Double = Me.C1Chart1.ChartArea.AxisX.Min
Dim xMax As Double = Me.C1Chart1.ChartArea.AxisX.Max
Dim xChange As Double = (xMax - xMin) * 0.05
Me.C1Chart1.ChartArea.AxisX.Min = xMin + xChange
Me.C1Chart1.ChartArea.AxisX.Max = xMax - xChange
Dim yMin As Double = Me.C1Chart1.ChartArea.AxisY.Min()
Dim yMax As Double = Me.C1Chart1.ChartArea.AxisY.Max
Dim yChange As Double = (yMax - yMin) * 0.05
Me.C1Chart1.ChartArea.AxisY.Min = yMin + yChange
Me.C1Chart1.ChartArea.AxisY.Max = yMax - yChange
End Sub
' Controls zooming out
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
Dim xMin As Double = Me.C1Chart1.ChartArea.AxisX.Min
Dim xMax As Double = Me.C1Chart1.ChartArea.AxisX.Max
Dim xChange As Double = (xMax - xMin) * 0.05
Me.C1Chart1.ChartArea.AxisX.Min = xMin - xChange
Me.C1Chart1.ChartArea.AxisX.Max = xMax + xChange
Dim yMin As Double = Me.C1Chart1.ChartArea.AxisY.Min()
Dim yMax As Double = Me.C1Chart1.ChartArea.AxisY.Max
Dim yChange As Double = (yMax - yMin) * 0.05
Me.C1Chart1.ChartArea.AxisY.Min = yMin - yChange
Me.C1Chart1.ChartArea.AxisY.Max = yMax + yChange
End Sub
```

- C#

```
// Controls zooming in
private void button1_Click(object sender, System.EventArgs e)
{
double xMin = this.c1Chart1.ChartArea.AxisX.Min;
double xMax = this.c1Chart1.ChartArea.AxisX.Max;
double xChange = (xMax - xMin) * 0.05;
this.c1Chart1.ChartArea.AxisX.Min = xMin + xChange;
this.c1Chart1.ChartArea.AxisX.Max = xMax - xChange;
double yMin = this.C1Chart1.ChartArea.AxisY.Min();
double yMax = this.C1Chart1.ChartArea.AxisY.Max;
double yChange = (yMax - yMin) * 0.05;
this.c1Chart1.ChartArea.AxisY.Min = yMin + yChange;
this.c1Chart1.ChartArea.AxisY.Max = yMax - yChange;
}
// Controls zooming out
private void button2_Click(object sender, System.EventArgs e)
{
double xMin = this.c1Chart1.ChartArea.AxisX.Min;
double xMax = this.c1Chart1.ChartArea.AxisX.Max;
double xChange = (xMax - xMin) * 0.05;
this.c1Chart1.ChartArea.AxisX.Min = xMin - xChange;
this.c1Chart1.ChartArea.AxisX.Max = xMax + xChange;
double yMin = this.c1Chart1.ChartArea.AxisY.Min();
double yMax = this.c1Chart1.ChartArea.AxisY.Max;
double yChange = (yMax - yMin) * 0.05;
this.c1Chart1.ChartArea.AxisY.Min = yMin - yChange;
this.c1Chart1.ChartArea.AxisY.Max = yMax + yChange;
}
```

15.2.3 用多 Y-轴线来缩放(Zoom)，平移和缩放(Scale)

为了在诸如缩放(Zoom)，平移和缩放(Scale)等的交互特性中使用第二个 y-轴线，必须设置对应的 Axis 属性。例如:

- Visual Basic

```
C1Chart1.Interaction.Actions("Zoom").Axis = AxisFlagEnum.AxesAll
```

- C#

```
c1Chart1.Interaction.Actions["Zoom"].Axis = AxisFlagEnum.AxesAll;
```

16. WinForms 中的图表示例

请注意 ComponentOne 软件工具中含有大量的实例工程或者演示程序，这些对于 ComponentOne Studios 中的工具开发很有用。

可以通过 **ComponentOne Sample Explorer** 来访问演示程序。在你的电脑桌面上点击 Start 按钮并依次点击 **ComponentOne | Studio for WinForms| Samples| Chart Samples** 菜单。下表是所有演示程序的一个描述。

点击以下连接中的一个来访问 **C1Chart** 或者 **C1Chart Mobile Components** 示例程序。

C1Chart Samples

C1Chart 含有以下的 C#和 VB 示例程序。

演示程序	描述
Alarmz	演示了一个使用 AlarmZones 的 2D 散点图表(散点图表用于显示一个资料片)，示例程序使用了 C1Chart 控件。
AlarmZoneShapes	演示了使用不同样式的 AlarmZones 的 XY 图表 ,示例程序使用了诸如矩形，椭圆形，多边形等的 AlarmZones 样式来创建默认的 XY 图表。同时也含有在混合模式下使用 ValueLabels，同时还有特征标记。
AutoArrangement	演示了一个无需控制的可以自动调整的 XY 绘制的标签。示例程序创建了一个 XY 绘制并且演示了能避免标签重叠的图表标签的自动定位。
Box	演示了一个盒须(用来表示一组数据，并且让你很容易看到那些数字出现的最多)图表，该示例使用 C1Chart 控件。
Bubbles	演示了一个数据流图表，该示例使用 C1Chart 控件。
CandyBox	演示了一个区域图表 ,并且可以玩游戏。该示例使用 C1Chart 控件。
Chlabels	演示了一个带有鼠标追踪的水平条形图表，该示例使用 C1Chart 控件。
ChartLoader	演示了使用 XML 来对各种 2D 和 3D 图表进行的加载和保存动作。并且运行使用图表属性对话框，图表向导对话框，或者基本的属性网格来进行编辑。
CpuUsage	演示了 CPU 的使用情况和历史数据，示例使用 C1Chart 控件。
CustomBrushes	演示了为线条和填充使用自定义的画刷。示例使用 C1Chart 控件。
CustomDraw	演示了一个使用螺旋形式的单序列创建的 2D XY 绘制。通过使用 ChartDataSeries 绘制事件，螺旋是使用双色绘制的。可以使用该事件进行更进一步的自定义设置。
DataBoundChart	演示了一个使用从一个数据库中获取的数据进行绘制的 2D 水平条形图表。示例使用 C1Chart 控件。

DataChart	演示了一个使用从一个数据库中获取的数据进行绘制的 2D 水平条形图表。示例使用 C1Chart 控件。
DataStyl	演示了多种允许交互式样式变化的 2D 图表。示例使用 C1Chart 控件。
Demo2D	探讨了所有的 2D 图表类型和它们的属性 ,示例使用 C1Chart 控件。
DiskSpace	使用一个饼形统计图和效果显示了磁盘空间使用情况。示例使用 C1Chart 控件。
Donut	演示了一个圆环形图表的各种功能。包括字母混合，提示信息(tooltips)，序列突出效果。示例创建了一个圆环形图表并演示了如何使用提示信息和序列突出效果。一个圆环形图表是一个含有指定大小内径的饼形图表。 示例使用 C1Chart 控件。
ErrorBar	演示了一个含有错误提示条的 XY 绘制。示例使用 C1Chart 控件。
FExplorer	创建并绘制显式的且参数化的函数的不同的组。示例使用 C1Chart 控件。
FloatBar	演示了一个浮动的条形图表，示例使用 C1Chart 控件。
Function2d	演示了指定函数的 XY 绘制和极坐标图表。示例使用 C1Chart 控件。
Gantt	演示了一个用来表示标号任务的开始和结束日期的基本类型的甘特图(主要用于表示项目进度的一种图类型)表。
HLCandle	演示了不同种类的价格/股票图表。示例使用 C1Chart 控件。
Histogram	演示了一个展示从一个特定点到一个 2D 散点图表上的各个点的距离的分布情况的矩形图表。示例使用随机数据来创建一个 2D 散点图表 ,然后在第二张图表中 ,基于各个数据点到一个十字准线的距离创建了一个矩形图表。十字准线是使用可移动的带有标记箭头和网格线的变量值标签生成的。示例使用 C1Chart 控件。
HistTemp	演示了如何使用 C1Chart 中的 GenerateHistogramData 方法来生成矩形图数据，并在一个条形图中加载并显示数据。示例创建了美国密苏里州的堪萨斯城的季节性平均气温的 XY 绘制。在第二个图表中，一个条形图表含有根据不同年份的平均气温差异的分布而生成的矩形图数据。统计信息由图表里相应的数据信息计算得出而且在相应的文本框中显示。
Interview	演示了一个含有多种效果的雷达图(用于多元数据显示)。示例使用 C1Chart 控件。
LogPlots	演示了一个使用幂级数来生成数据的日志扩展提示。它还演

	示了一种鲜为人知的专门为日志绘制提供的格式。
Multi	演示了多并发图表和打印技术，示例使用 C1Chart 控件。
Pie Grouping	演示了根据类型为扇形分组的堆积饼图的用法。示例创建了一个根据食品类型分类的堆积饼图。每一个数据项都被作为一个单独的序列添加到图表中。
PieStuff	演示了一个使用标签和效果的饼图。示例使用 C1Chart 控件。
PointStyles	使用不同的选项创建点样式，并应用到条形图和散点图数据点中，示例使用 C1Chart 控件。
PrintIt2D	加载 2D 和 3D 备份，并为它们生成图片或者打印，示例使用 C1Chart 和 C1Chart3D 控件。
Radar	演示了一个雷达图并展示了如何迭代和填充雷达数据。示例使用 C1Chart 控件。
RotatedLabels	演示了一个图表标签，并让标签根据一个固定点进行旋转。
RtfTitle	示例创建了一个 XY 绘制，并使用 rft 格式添加表头和页尾。示例中有一个简单的带有富文本框的 rft 编辑器来编辑表头和页尾。
Scatter	演示了一个散点图表，并用一个游戏展示了鼠标追踪。示例使用 C1Chart 控件。
SelectSeries	示例创建了一个带有多序列和一个在条形图表中有轴线滚动的表现和效果的倒置的 XY 绘制。单个序列可以使用鼠标点击进行选中，选中的序列会被高亮。
StepChart	示例创建了一个含有 2D 符号，和 3D 效果的步骤图表。示例创建了一个步骤图表，并且演示了提示消息的用法。还有使用 3D 效果来改变步骤图表的 3D 海拔和旋转。同样地也展示了步骤图表中的鼠标跟踪。
StockChart	演示了一个基本的带有成交量的开放的高低-收盘图。示例使用 C1Chart2D 控件。
ToolTip	示例创建了一个 2Dxy 绘制图，并且演示了图表提示消息功能，属性和事件。
TrendAndError	示例创建了两个图表，上面的图表表示了带有多项式趋势线的随机数据。下面的图表示了上面的图表上的每一个点的回归错误。示例使用 C1Chart 控件。
TrendLines	使用各种模型创建伪随机数据，并且使用不同类型的趋势线绘制它们。 示例使用 C1Chart 控件。

- 移动控件示例

C1Chart 中的移动控件含有以下的 VB 和 C#示例程序:

示例程序	**描述**
Alarmz	演示了一个使用 AlarmZones 的 2D 散点图表(用于显示一个资料片)，

	示例程序使用了 C1Chart 控件。
Box	演示了一个盒须(用来表示一组数据，并且让你很容易看到那些数字出现的最多)图表，该示例使用 C1Chart 控件。
Bubbles	演示了一个数据流图表，该示例使用 C1Chart 控件。
Chlabels	演示了一个带有鼠标跟着的水平条形图表，该示例使用 C1Chart 控件。
DataStyle	演示了多种允许交互式样式变化的 2D 图表。示例使用 C1Chart 控件。
FloatBar	演示了一个浮动的条形图表，示例使用 C1Chart 控件。
HLCandle	演示了不同种类的价格/股票图表。示例使用 C1Chart 控件。
MemStatus	示例使用 C1Chart 控件。
Pie	演示了一个使用标签和效果的饼图。示例使用 C1Chart 控件。
Radar	演示了一个雷达图并展示了如何迭代和填充雷达数据。示例使用 C1Chart 控件。
Scatter	演示了一个散点图表，并用一个游戏展示了鼠标跟踪。示例使用 C1Chart 控件。

17. WinForms 图表指南

以下步骤将引导您创建基本的图表。例如，线条，饼状图，x-y 绘制，烛台图表。

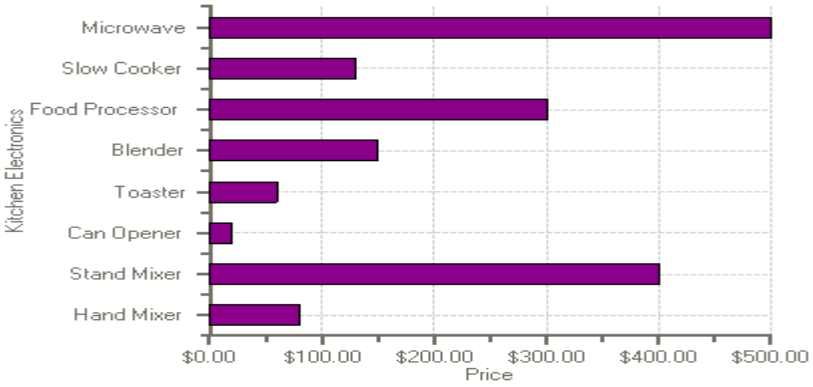
本章节含有以下指南:

指南	描述
Bar Chart Tutorial(268 页)	本指南演示了如何在运行时或者代码来创建一个简单的条形图。
Line Chart Tutorial (273 页)	本指南演示了如何在运行时或者通过代码来创建一个线形图。
Pie Chart Tutorial(278 页)	本指南演示了如何在运行时或者通过代码来创建一个饼状图。
Candle Chart Tutorial (281 页)	本指南演示了如何在运行时或者通过代码来创建一个烛台图。

条形图指南

本节将教您如何一步步地创建一个简单的条形图。该图形作为一个简单的条形图表用来显示信息，其中，y-轴表示每个产品价格的数值点，x-轴提供点之间的间距信息并为每一个数据点提供附属的标签信息。

在您创建该图表以前给您展示一下该图表大致的样子。



在设计时创建一个条形图

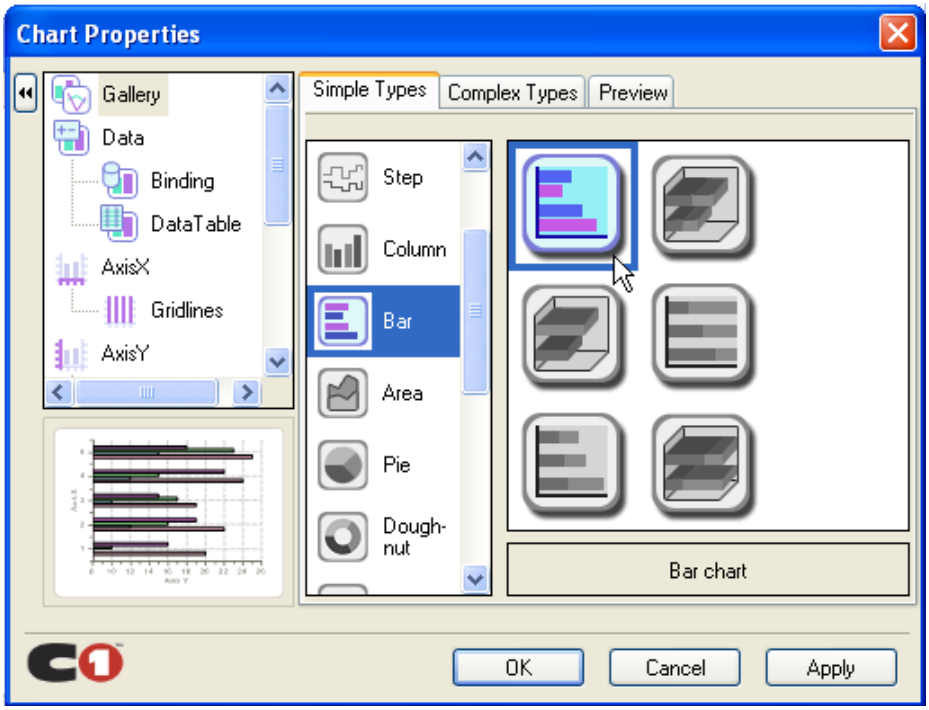
本主题引导您使用图**表属性**编辑器来创建一个简单的条形图。在本主题中您将学习到如何设置特殊的图表类型，为图表添加数据，给轴线做注解，所有的这些都是通过几个简单的步骤在设计时完成的。

这些任务都假定您已经将 C1Chart 控件添加到窗体中。

设置图表类型

使用一个智能标签向导来配置一个图表的第一个步骤是从可用图表类型中选择一个库类型。从像条形图和线性图一样的简单图表到像极地和表面一样的更加复杂的图表 ,C1Chart 允许您进行任何的数据可视化处理。

1. 在 C1Chart 控件上右键点击，然后在快捷菜单中点击 Chart Properties。
2. 在树视图面板上选择 Gallery。然后从 Simple Types 列表中选择 Bar。
3. 在面板上，选择左上的条目，然后点击 Apply 按钮。

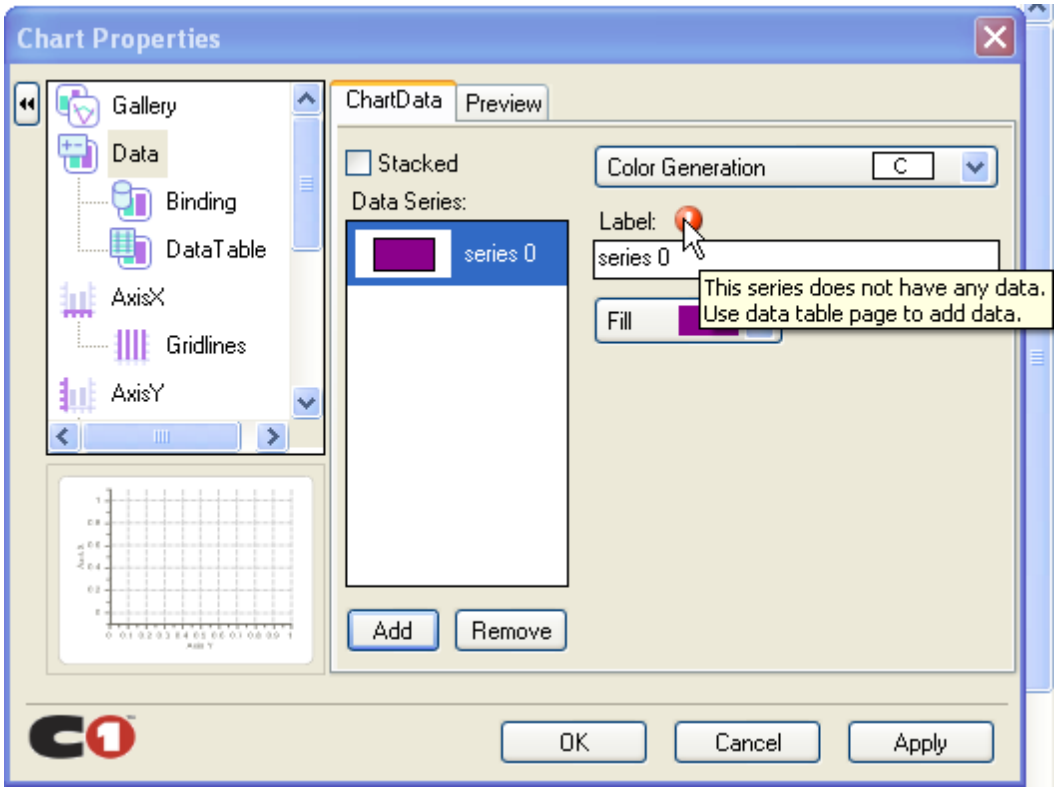


修改数据序列

4. 从树视图面试中选择 Data。

5. 通过点击 Remove 来删除 Series0， Series 1， Series 2， 和 Series 3。
6. 通过点击 Add 来添加一个新的用于显示商品价格的 Y-值的序列。

会出现一个提示信息来提醒您必须给新序列添加数据。



给数据序列添加数据

7. 从树视图面板中选择 **Data->DataTable。**然后向数据表中添加以下的数据。
8. 在 X-表上单击 ,并输入以下的 X-Y 数值对: (0,80) ,(1,400)，(2,20)，(3,60)，(4,150)，(5,300)， (6,130)， 和(7,500)。

注意:通过点击 **Tab** 来将光标移动到下一个 X-Y 值中。

9. 点击 **Apply** 来更新图表。

一个更新后的图表的预览图像会出现在图表属性编辑器的左下方区域中。

配置轴线

下面我们给 x-轴线和 y-轴线做注解。我们将使用值标签来给 x-轴线做注解。关于值标签注解的更多信息，请参见值标签注解(222 页)。

10. 从树视图面板中选择 **AxisX**，接着选择注解选项卡。
11. 从方法下拉列表框中选择 **ValueLabels。**

    一个 **Labels** 按钮将会出现。
12. 点击 **Labels** 来打开 **ValueLabel Collection Editor。**
13. 点击 **Add** 按钮八次来添加八个值标签。
14. 在 **Members** 列表框中选择第一个值标签。并设置 **NumericValue** 属性为 **0**，并且设置它的 **Text** 属性为**"Hand Mixer"**
15. 设置标签的剩余属性为以下的值:
    - **ValueLabel1 的 NumericValue** 属性设置为 1 并将 Text 属性设置为"**Stand**

Mixer"。

- **ValueLabel2 的 NumericValue** 属性设置为 2 并将 Text 属性设置为"**Can Opener"。**
- **ValueLabel3 的 NumericValue** 属性设置为 3 并将 Text 属性设置为"**Toaster"。**
- **ValueLabel4 的 NumericValue** 属性设置为 4 并将 Text 属性设置为"**Blender**"。
- **ValueLabel5 的 NumericValue** 属性设置为 5 并将 Text 属性设置为"**Food Processor"。**
- **ValueLabel6 的 NumericValue** 属性设置为 6 并将 Text 属性设置为**"Slow Cooker"。**
- **ValueLabel7 的 NumericValue** 属性设置为 7 并将 Text 属性设置为 "**Microwave"。**

16. 点击 **OK** 按钮来将值标签应用到 x-轴线注解上。
17. 从树视图面板上选择 **AxisY，**然后选择注解选项卡。
18. 从格式化下拉列表中选择 **NumericCurrency。**

给轴线设置标题

19. 设置 AxisX 标题，从树视图面板中选择 AxisX。并将其 Title 设置为”Kitchen Electronics”
20. 设置 AxisY 标题，从树视图面板中选择 AxisY。并将其 Title 设置为”Price”
21. 点击 OK 来关闭图表属性编辑器。

以编程方式创建一个条形图

一个简单的条形图可以按照以下步骤来以编程方式来创建:

1. 给窗体添加一个 **C1Chart** 控件。
2. 在窗体上右键点击，并选择查看代码来查看代码文件。然后添加以下的代码来声明 **C1.Win.C1Chart 命名空间。**

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

3. 3 ,双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来生成一个简单的条形图。

- Visual Basic

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Clear previous data
    C1Chart1.ChartGroups(0).ChartData.SeriesList.Clear()
    ' Add Data
    Dim ProductNames() As String = {"Hand Mixer", "Stand Mixer", "Can Opener",
"Toaster" ,"Blender" ,"Food Processor" ,"Slow Cooker" ,"Microwave"}
    Dim PriceX() As Integer = {80, 400, 20, 60, 150, 300, 130, 500}
    ' Create first series
    Dim ds1 As C1.Win.C1Chart.ChartDataSeries = _
    C1Chart1.ChartGroups(0).ChartData.SeriesList.AddNewSeries()
    ds1.Label = "PriceX"
    ds1.X.CopyDataIn(ProductNames)
    ds1.Y.CopyDataIn(PriceX)
    ' Set chart type
    c1Chart1.ChartArea.Inverted = True
    c1Chart1.ChartGroups(0).ChartType = _
    C1.Win.C1Chart.Chart2DTypeEnum.Bar
    ' Set axes titles
    c1Chart1.ChartArea.AxisX.Text = "Kitchen Electronics"
    c1Chart1.ChartArea.AxisY.Text = "Price"
      ‘ Set format for the Y-axis annotation
    c1Chart1.ChartArea.AxisY.AnnoFormat = FormatEnum.NumericCurrency
    End Sub
```

- C#

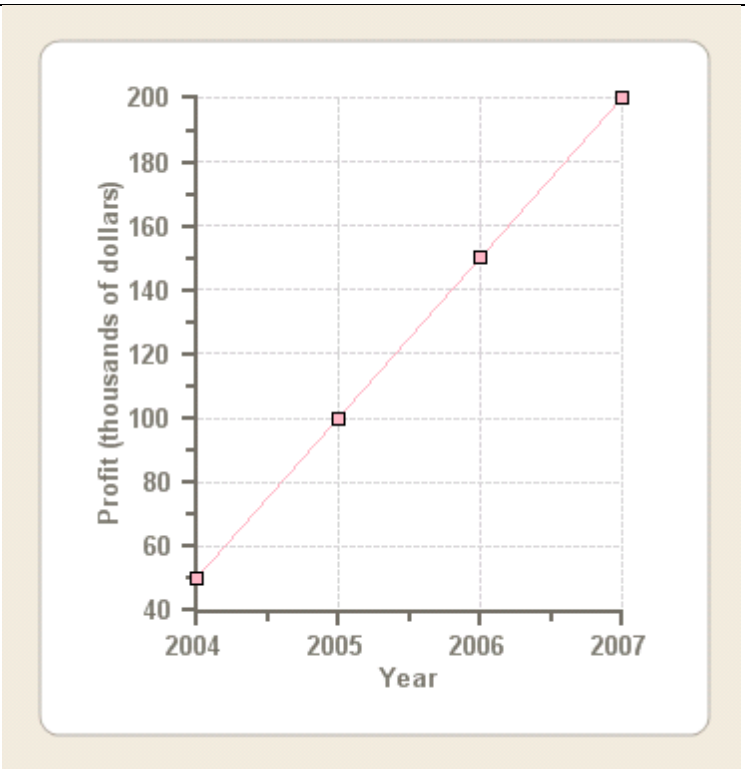
```
    private void Form1_Load(object sender, EventArgs e)
    {
    // Clear previous data
    c1Chart1.ChartGroups[0].ChartData.SeriesList.Clear();
    // Add Data
    string[] ProductNames = { "Hand Mixer", "Stand Mixer", "Can Opener",
"Toaster" ,"Blender" ,"Food Processor" ,"Slow Cooker" ,"Microwave"};
    int[] PriceX = { 80, 400, 20, 60, 150, 300, 130, 500 };
    // create single series for product price
    C1.Win.C1Chart.ChartDataSeries ds1 =
c1Chart1.ChartGroups[0].ChartData.SeriesList.AddNewSeries();
    ds1.Label = "Price X";
    //Use the CopyDataIn method of the ChartDataArray object to copy the X and Y value data
into the data series
    ds1.X.CopyDataIn(ProductNames);
    ds1.Y.CopyDataIn(PriceX);
    // Set chart type
    c1Chart1.ChartArea.Inverted = true;
    c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Bar;
    // Set axes titles
    c1Chart1.ChartArea.AxisX.Text = "Kitchen Electronics";
    c1Chart1.ChartArea.AxisY.Text = "Price";
    //Set format for the Y-axis annotation
    c1Chart1.ChartArea.AxisY.AnnoFormat = FormatEnum.NumericCurrency;
    }
```

17.1 线性图表指南

本章节介绍了如何创建带有符号的线性图表，它是默认的图表类型。一个线性图表是另外一种简单的用于展示数据关系的方式。

在这个示例里我们会创建一个带有符号的线性图表。我们只使用一个数据序列，因为只有一条线。这个线将用来表示一定时间范围内一个公司的利润的增长。在这个例子中的水平轴线，x 轴，代表年份值。例子中的垂直轴线，y 轴，代表以千美元为单位的利润值。

当你完成以下的步骤后，你将会创建如下图的出来。



在设计时创建带有符号的线性图表。

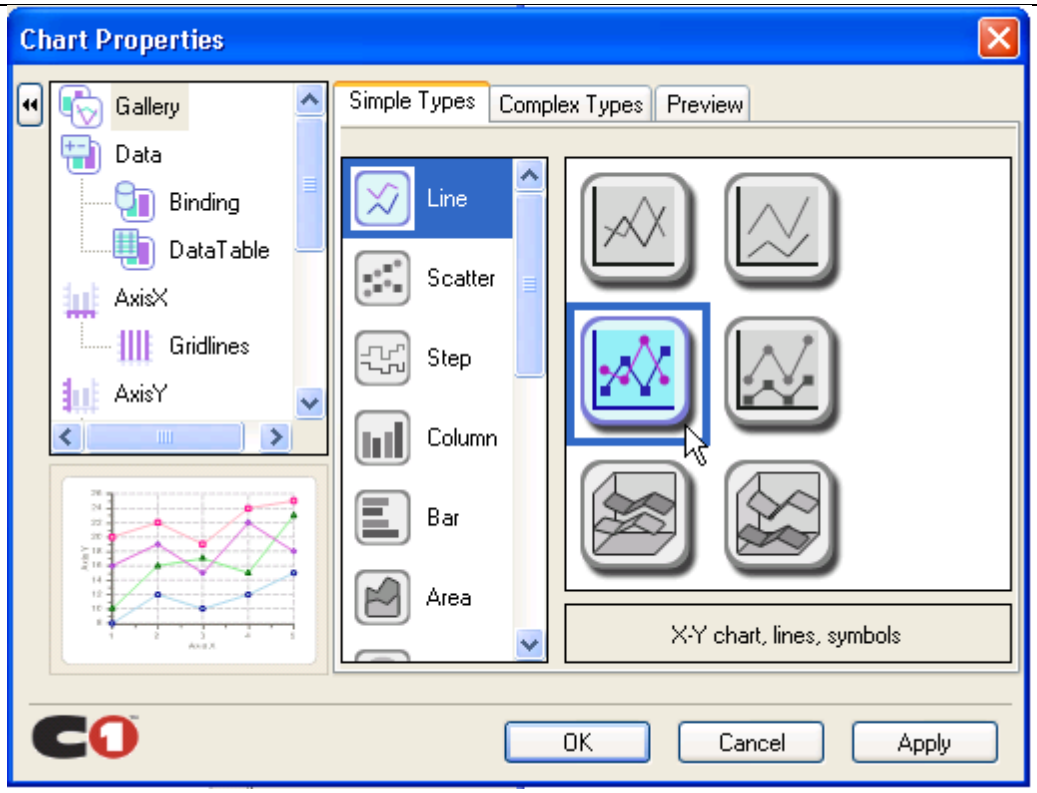
本任务假设您已经在您的窗体上添加了一个 C1Chart 控件。

设置图表类型

使用一个图表属性设计器来配置一个图表的第一个步骤是从可用图表类型中选择一个库类型。

1. 在 **C1Chart** 控件上右键点击，然后在快捷菜单中点击 **Chart Properties**。
2. 在树视图面板上选择 **Gallery**。然后从 **Simple Types** 列表中选择 **Line**。
3. 在紧挨着主要图表类型的面板上，选择 **X-Y chart**， **lines**， **symbols subtype**，然后点击 **Apply** 按钮。

默认的线性图表将会添加两个数据序列，并创建条线。



修改数据序列

4. 从树视图面试中选择 **Data**。
5. 通过点击 **Remove** 来删除 **Series0， Series 1， Series 2， 和 Series 3。**

给数据序列添加数据

6. 从树视图面板中选择 **Data->DataTable。** ，在 XY 数据表上单击，并输入以下的 X-Y 数值对:

   (2004，50)， (2005，100)， (2006， 150)， (2007， 100)。

注意:通过点击 Tab 来将光标移动到下一个 X-Y 值中。

7. 从默认的线性图表中删除最后一个 XY 值。
8. 点击 **Apply** 来更新图表。

一个更新后的图表的预览图像会出现在图表属性编辑器的左下方区域中。

修改轴线外观

下面我们修改 X 轴 Y 轴的默认标题，并且我们将使用图表属性编辑器来改变轴线的字体样式。从树视图面板中选择 **AxisX，**接着选择注解选项卡

9. 从树视图面板中选择 **AxisX。**
10. 在 AxisX 选项卡中，在标题中输入”**Year**”。然后点击 **Font** 按钮，然后出现字体对话框。
11. 将字体样式设置为粗体并将字体大小设置为 **10。**然后点击 **OK** 按钮。
12. 在图表属性编辑器中，点击 **Apply** 按钮。对 X-轴线的修改就会反映在图表中。
13. 从树视图中选择 **AxisY。**
14. 在 AxisY 选项卡中，在标题中输入**” Profit (thousands of dollars)** ”。然后点击 **Font** 按钮，然后出现字体对话框。

15. 将字体样式设置为粗体并将字体大小设置为 10。然后点击 **OK** 按钮。
16. 在图表属性编辑器中，点击 **Apply** 按钮。对 Y-轴线的修改就会反映在图表中。
17. 点击 OK 来关闭图表属性编辑器。

有成就感吧!您已经完成了使用图表属性编辑器创建一个线性图表的全过程。

以编程方式创建一个线性图表

一个简单的线性图可以按照以下步骤来以编程方式来创建:

1. 给窗体添加一个 C1Chart 控件。
2. 在窗体上右键点击，并选择查看代码来查看代码文件。然后添加以下的代码来声明 **C1.Win.C1Chart 命名空间。**

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

3. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来生成一个简单的条形图。

- Visual Basic

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    'create data for the chart
    Dim xdata() As Double = {2004, 2005, 2006, 2007}
    Dim ydata() As Double = {50, 100, 150, 200}
    'clear previous series
    C1Chart1.ChartGroups(0).ChartData.SeriesList.Clear()
    'add one series to the chart
    Dim ds As C1.Win.C1Chart.ChartDataSeries = _
C1Chart1.ChartGroups(0).ChartData.SeriesList.AddNewSeries()
    'copy the x and y data
    ds.X.CopyDataIn(xdata)
    ds.Y.CopyDataIn(ydata)
    'set the chart type
    C1Chart1.ChartGroups(0).ChartType = C1.Win.C1Chart.Chart2DTypeEnum.XYPlot
    'create new font for the X and Y axes
    Dim f As Font = New Font("Arial", 10, FontStyle.Bold)
    C1Chart1.ChartArea.Style.ForeColor = Color.DarkGray
    C1Chart1.ChartArea.AxisX.Font = f
    C1Chart1.ChartArea.AxisX.Text = "Year"
    C1Chart1.ChartArea.AxisX.GridMajor.Visible = True
    C1Chart1.ChartArea.AxisX.GridMajor.Color = Color.LightGray
    C1Chart1.ChartArea.AxisY.Font = f
    C1Chart1.ChartArea.AxisY.Text = "Profit (thousands of dollars)"
    C1Chart1.ChartArea.AxisY.GridMajor.Visible = True
    C1Chart1.ChartArea.AxisY.GridMajor.Color = Color.LightGray
    'change the default line style appearance
    ds.LineStyle.Color = Color.LightPink
    ds.LineStyle.Pattern = LinePatternEnum.Solid
    ds.LineStyle.Thickness = 1
    'change the default symbol style appearance
    ds.SymbolStyle.Shape = SymbolShapeEnum.Box
    ds.SymbolStyle.Color = Color.LightPink
    ds.SymbolStyle.OutlineColor = Color.Black
    ds.SymbolStyle.Size = 5
    ds.SymbolStyle.OutlineWidth = 1
    'set the backcolor for the plot area
    C1Chart1.ChartArea.PlotArea.BackColor = Color.White
    End Sub
```

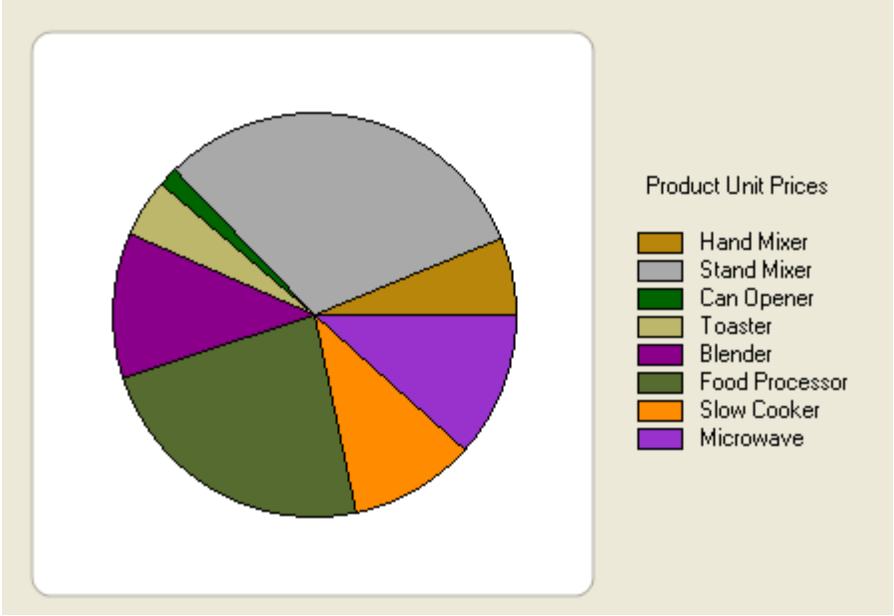
- C#

```
private void Form1_Load(object sender, EventArgs e)
{
//create data for the chart
double[] xdata = { 2004, 2005, 2006, 2007 };
double[] ydata = { 50, 100, 150, 200 };
//clear previous series
c1Chart1.ChartGroups[0].ChartData.SeriesList.Clear();
//add one series to the chart
C1.Win.C1Chart.ChartDataSeries ds =
c1Chart1.ChartGroups[0].ChartData.SeriesList.AddNewSeries();
//copy the x and y data
ds.X.CopyDataIn(xdata);
ds.Y.CopyDataIn(ydata);
//set the chart type
c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.XYPlot;
//create new font for the X and Y axes
Font f = new Font("Arial", 10, FontStyle.Bold);
c1Chart1.ChartArea.Style.ForeColor = Color.DarkGray;
c1Chart1.ChartArea.AxisX.Font = f;
c1Chart1.ChartArea.AxisX.Text = "Year";
c1Chart1.ChartArea.AxisX.GridMajor.Visible = true;
c1Chart1.ChartArea.AxisX.GridMajor.Color = Color.LightGray;
c1Chart1.ChartArea.AxisY.Font = f;
c1Chart1.ChartArea.AxisY.Text = "Profit (thousands of dollars)";
c1Chart1.ChartArea.AxisY.GridMajor.Visible = true;
c1Chart1.ChartArea.AxisY.GridMajor.Color = Color.LightGray;
//modify line style appearance
ds.LineStyle.Color = Color.LightPink;
ds.LineStyle.Pattern = LinePatternEnum.Solid;
ds.LineStyle.Thickness = 1;
//modify the symbol style appearance
ds.SymbolStyle.Shape = SymbolShapeEnum.Box;
ds.SymbolStyle.Color = Color.LightPink;
ds.SymbolStyle.OutlineColor = Color.Black;
ds.SymbolStyle.Size = 5;
ds.SymbolStyle.OutlineWidth = 1;
c1Chart1.ChartArea.PlotArea.BackColor = Color.White;
}
```

17.2 饼状图表指南

本节将教您如何一步步地创建一个饼状图表。饼状图表用来表示简单值。不同于其他的C1Chart 图表类型,饼状图表必须含有多于一个的序列以显示多个切片。如果仅有一个序列，那么饼状图表将会显示一个圆环而不会显示任何的切片。在本实例中，我们会创建有八个切片的饼状图表，所以我们会有八个序列，每个点对应一个序列，最终会有八个点。

下图展示了一个使用一个切片来代表一个商品种类的饼状图表。



以下将说明使用图表对象来在运行时和以编程方式来创建一个饼状图表的步骤。

在设计时创建一个饼状图表。

本任务假设您已经在您的窗体上添加了一个 C1Chart 控件。

设置图表类型

使用一个图表属性设计器来配置一个图表的第一个步骤是从可用图表类型中选择一个库类型。

1. 在 **C1Chart** 控件上右键点击，然后在快捷菜单中点击 **Chart Properties**。
2. 在树视图面板上选择 **Gallery**。然后从 Simple Types 列表中选择 **Pie**。
3. 在面板上，选择左上的饼状图，然后点击 **Apply** 按钮。

默认的饼状图表将添加五个饼状图并且每一个带有四个数据序列。

修改数据序列

4. 从树视图面试中选择 **Data**。
5. 通过点击 Remove 来删除 Series0， Series 1， Series 2， 和 Series 3。
6. 点击 **Add** 来为 y-值新增一个序列，它用来表示商品的价格。

会有一个警告信息来提醒您必须给新增的序列添加数据。

7. 点击 **Add** 按钮七次来添加剩余的七个数据序列到饼形图表中。这样，该饼形图表将含有八个切片。
8. 从树视图面板中选择 **Data->DataTable**。
9. 在数据表上点击第一个字段 ,并输入以下的(X,Y)数值对: (1,80)，(1,400)，(1,20)，

(1,60)， (1,150)， (1,300)， (1,130)， 和 (1,500)。

10. 从树视图面板中选择数据项目，然后在数据序列下拉框中选择数据序列。然后给每一个数据序列的 Label 文本框中输入以下的内容:
    - 设置 Series0 为"Hand Mixer ($80.00)"
    - 设置 Series1 为"Stand Mixer ($400.00)"
    - 设置 Series2 为"Can Opener ($20.00)"
    - 设置 Series3 为"Toaster ($60.00)"
    - 设置 Series4 为"Blender ($150.00)"
    - 设置 ries5 为 Food Processor ($300.00)"
    - 设置 ries6 为 Slow Cooker ($130.00)"
    - 设置 ries7 为 crowave ($500.00)"

给饼形图表添加图例

为了启用图例。在图表属性编辑器的树视图面板选择 Appearance，然后选择 Legend 复选框。

以编程方式来创建一个饼状图表

本任务假设您已经在您的窗体上添加了一个 C1Chart 控件。

按照以下的步骤可以以编程方式创建一个饼状图表:

1. 添加以下的代码来引入 C1.Win.C1Chart 命名空间。

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

2. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来生成一个简单的饼形图:

- Visual Basic

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' Set chart type
    C1Chart1.ChartArea.Inverted = True
    C1Chart1.ChartGroups(0).ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Pie
    ' Clear previous data
    C1Chart1.ChartGroups(0).ChartData.SeriesList.Clear()
    ' Add Data
    Dim ProductNames As String() = {"Hand Mixer", "Stand Mixer", "Can Opener", "Toaster",
"Blender", "Food Processor", "Slow Cooker", "Microwave"}
    Dim PriceX As Integer() = {80, 400, 20, 60, 150, 300,
    130, 500}
    'get series collection
    Dim dscoll As ChartDataSeriesCollection = C1Chart1.ChartGroups(0).ChartData.SeriesList
    For i As Integer = 0 To PriceX.Length - 1
    'populate the series
    Dim series As ChartDataSeries = dscoll.AddNewSeries()
    'Add one point to show one pie
    series.PointData.Length = 1
    'Assign the prices to the Y Data series
    series.Y(0) = PriceX(i)
    'format the product name and product price on the legend
    series.Label = String.Format("{0} ({1:c})", ProductNames(i), PriceX(i))
    Next
    ' show pie Legend
    C1Chart1.Legend.Visible = True
    'add a title to the chart legend
    C1Chart1.Legend.Text = "Product Unit Prices"
    End Sub
```

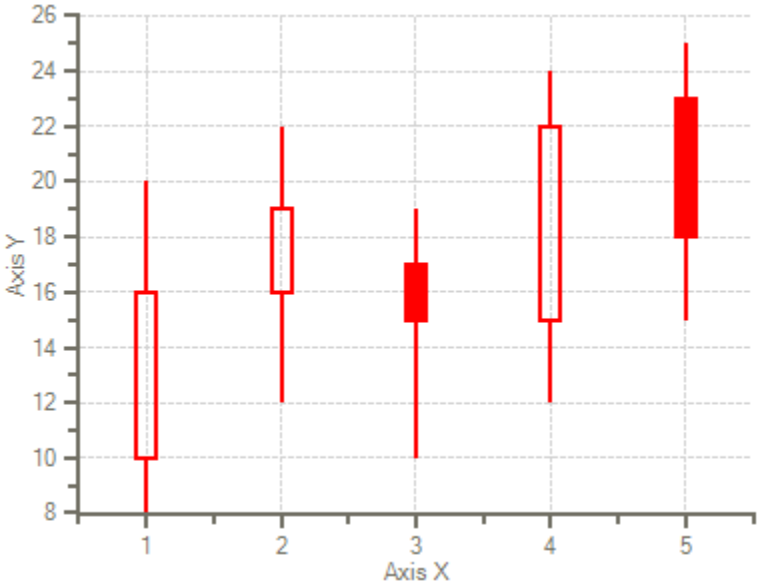
- C#

```
    private void Form1_Load(object sender, EventArgs e)
    {
    // Set chart type
    c1Chart1.ChartArea.Inverted = true;
    c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Pie;
    // Clear previous data
    c1Chart1.ChartGroups[0].ChartData.SeriesList.Clear();
    // Add Data
    string[] ProductNames = { "Hand Mixer", "Stand Mixer", "Can Opener",
"Toaster" ,"Blender" ,"Food Processor" ,"Slow Cooker" ,"Microwave"};
    int[] PriceX = { 80, 400, 20, 60, 150, 300, 130, 500 };
    //get series collection
    ChartDataSeriesCollection dscoll = c1Chart1.ChartGroups[0].ChartData.SeriesList;
    //populate the series
    for (int i=0; i < PriceX.Length; i++)
    {
    ChartDataSeries series = dscoll.AddNewSeries();
    //Add one point to show one pie
    series.PointData.Length = 1;
    //Assign the prices to the Y Data series
    series.Y[0] = PriceX[i];
    //format the product name and product price on the legend
    series.Label = string.Format("{0} ({1:c})", ProductNames[i], PriceX[i]);
    }
    // show pie Legend
    c1Chart1.Legend.Visible = true;
    //add a title to the chart legend
    c1Chart1.Legend.Text = "Product Unit Prices";
    }
```

17.3 烛台图指南

本节将教您如何一步步地创建一个烛台图表。图表使用烛台来表示信息，其中，y-轴线上的 Y 值代表股票的高，低，开，闭值。X-轴的值为每一个蜡烛提供位置。

在您创建图表之前下图显示了其大致的效果:



以下将说明使用图表对象来在运行时和以编程方式来创建一个烛台图的步骤。

在设计时创建一个烛台图

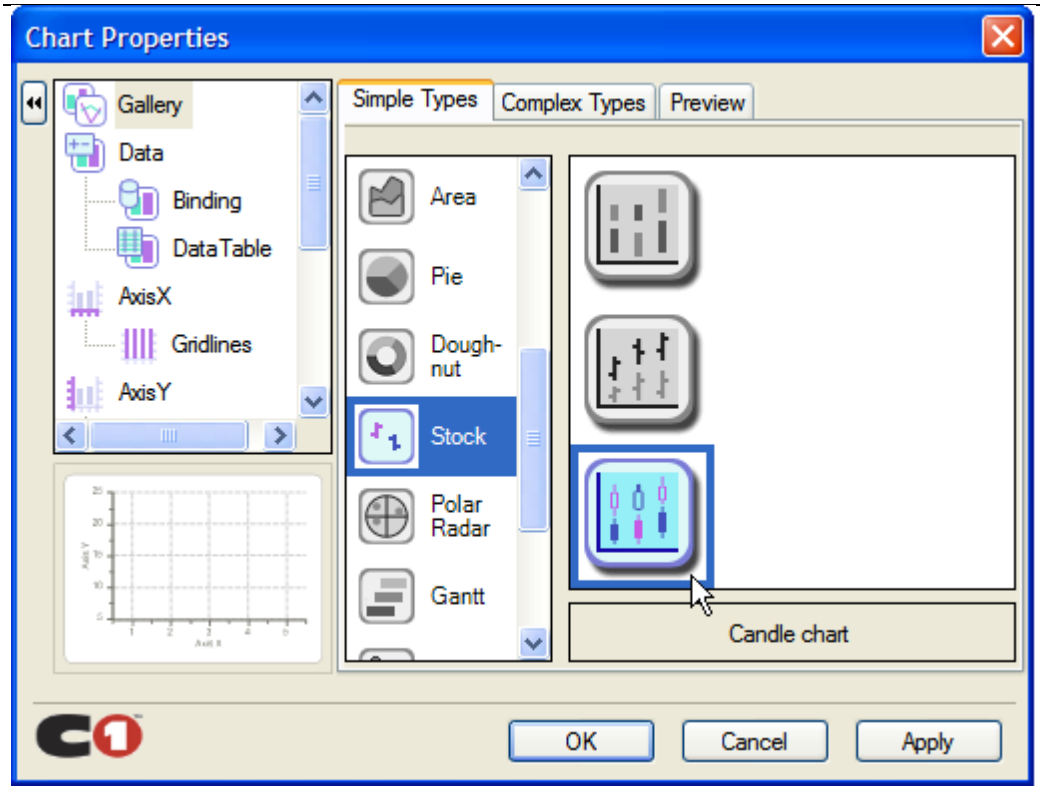
本主题将引导您使用图表属性编辑器来一步步地创建一个烛台图。在本主题中，您将学习到如何在设计时使用几个简单步骤来进行设置图表类型，修改数据序列，给图表添加数据，格式化外观等操作。

本任务假设您已经在您的窗体上添加了一个 C1Chart 控件。

设置图表类型

使用一个图表属性设计器来配置一个图表的第一个步骤是从可用图表类型中选择一个库类型。从像条形图和线性图一样的简单图表到像极地和表面一样的更加复杂的图表，C1Chart 允许您进行任何的数据可视化处理。

1. 在 **C1Chart** 控件上右键点击，然后在快捷菜单中点击 **Chart Properties**。
2. 在树视图面板上选择 **Gallery**。然后从 **Simple Types** 列表中选择 **Stock**。
3. 在面板上，选择左上的饼状图，然后点击 **Apply** 按钮。



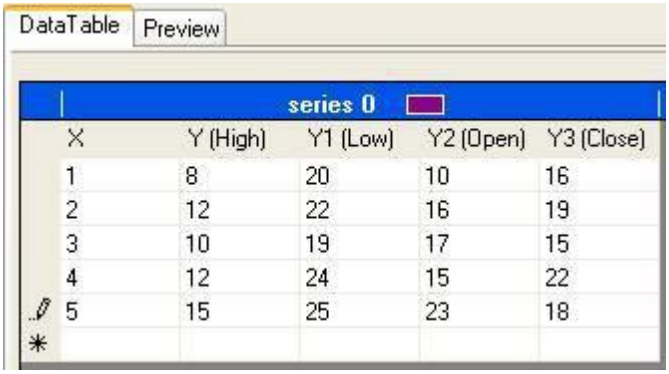
修改数据序列

4. 从树视图面试中选择 **Data**。
5. 通过点击 **Remove** 来删除 Series0， Series 1， Series 2， 和 Series 3。
6. 点击 **Add** 来为 y-值新增一个序列，它用来表示每一个股票的值。

会有一个警告信息来提醒您必须给新增的序列添加数据。

添加数据到数据序列中

7. 从树视图面板中选择 **Data->DataTable**。在数据表中添加以下的数据。
8. 像下述表格一样添加数据值。



X	Y (High)	Y1 (Low)	Y2 (Open)	Y3 (Close)
1	8	20	10	16
2	12	22	16	19
3	10	19	17	15
4	12	24	15	22
5	15	25	23	18

注意:X 代表在 x-轴上的位置。Y 代表高值，Y1 代表低值，Y2 代表开值，Y3 代表闭值。

9. 点击 **Apply** 按钮来更新图表。

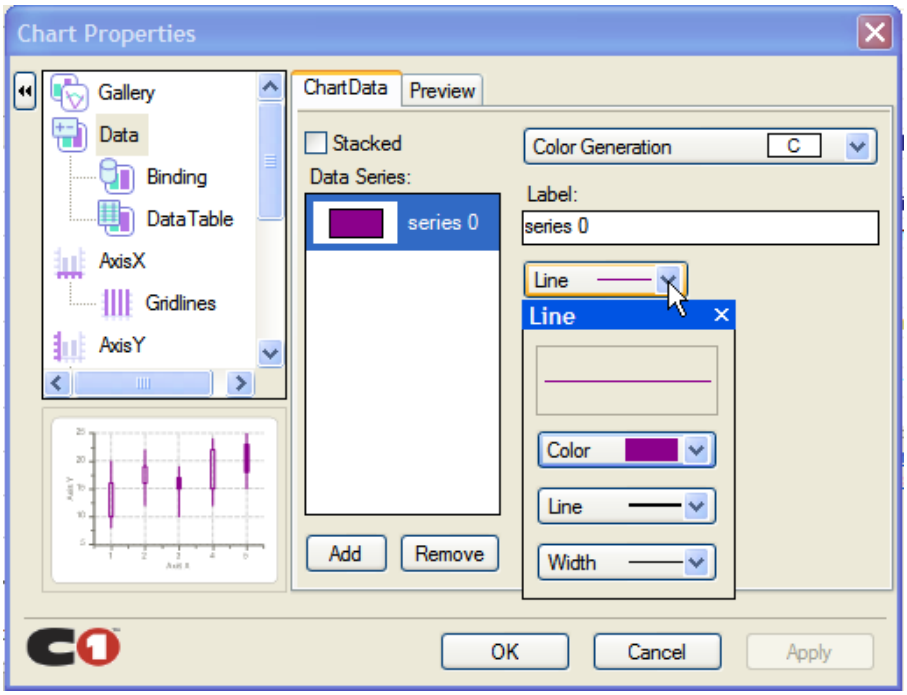
一个更新后的图表的预览图像会出现在图表属性编辑器的左下方区域中。

格式化烛台图

接下来我们将把线的颜色变为红色。增加蜡烛的柱体宽度，增加线的厚度。其中，我们使用 LineStyle 属性来把线的颜色变为红色，使用 SymbolStyle 属性来增加蜡烛的柱体宽度，使用

LineStyle 属性来增加线的厚度。

10. 在树视图面板中选择 **Data** 节点，然后选择 ChartData 选项卡。
11. 点击 **Line** 后面紧挨着的下拉箭头。



12. 在下拉列表中中选择 **Color**，并选择其中的红色。
13. 点击紧挨着 **Width** 的下拉箭头，并选择第二种线。
14. 点击 **OK** 来保存修改并关闭图表属性对话框。
15. 打开 C1Chart 属性窗口，并点击 **SeriesList** 后面的省略号按钮(…)。
    然后 **ChartDataSeries Collection Editor** 会出现。
16. 然后展开 **SymbolSize** 节点，并设置 **SymbolStyle**。Size 属性为 10。
17. 蜡烛的柱体大小会从默认的 5 变成 10。

以编程方式来创建一个烛台图

一个烛台图可以按照以下的步骤以编程方式创建。

1. 添加以下的代码来引入 **C1.Win.C1Chart** 命名空间。

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

2. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来生成一个简单的饼形图:

- Visual Basic

```
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As EventArgs)
    c1Chart1.ChartGroups(0).ChartData.SeriesList.AddNewSeries()
    'Declare the x and y variables as double to represent the x, y, y1, y2, and y3 values and assign
the values
    Dim x As Double() = New Double() {1, 2, 3, 4, 5}
    Dim y_hi As Double() = New Double() {8, 12, 10, 12, 15}
    Dim y_low As Double() = New Double() {20, 22, 19, 24, 25}
    Dim y_open As Double() = New Double() {10, 16, 17, 15, 23}
    Dim y_close As Double() = New Double() {16, 19, 15, 22, 18}
    'copy the x, y, y1, y2, and y3 data into the chart group
    c1Chart1.ChartGroups(0).ChartData(0).X.CopyDataIn(x)
    c1Chart1.ChartGroups(0).ChartData(0).Y.CopyDataIn(y_hi)
    c1Chart1.ChartGroups(0).ChartData(0).Y1.CopyDataIn(y_low)
    c1Chart1.ChartGroups(0).ChartData(0).Y2.CopyDataIn(y_open)
    c1Chart1.ChartGroups(0).ChartData(0).Y3.CopyDataIn(y_close)
    'assign the candle chart to the chart type
    c1Chart1.ChartGroups(0).ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Candle
    'Make the rising candles transparent to show rising prices
    c1Chart1.ChartGroups(0).HiLoData.FillTransparent = True
    'Declare the color for the lines
    c1Chart1.ChartGroups(0).ChartData.SeriesList(0).LineStyle.Color = System.Drawing.Color.Red
    'Increase the line thickness
    c1Chart1.ChartGroups(0).ChartData.SeriesList(0).LineStyle.Thickness = 2
    'Increase the body width of the candle
    c1Chart1.ChartGroups(0).ChartData.SeriesList(0).SymbolStyle.Size = 10
    End Sub
```

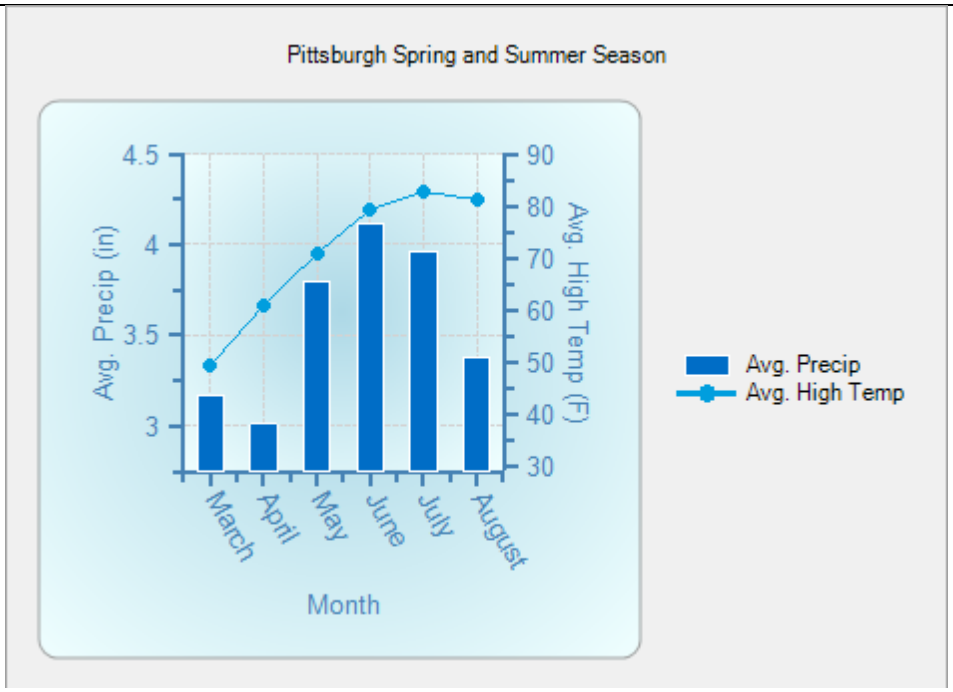
- C#

```
    private void Form1_Load(object sender, EventArgs e)
    {
    c1Chart1.ChartGroups[0].ChartData.SeriesList.AddNewSeries();
    //Declare the x and y variables as double to represent the x, y, y1, y2, and y3 values and
assign the values
    double[] x = new double[] { 1, 2, 3, 4, 5 };
    double[] y_hi = new double[] { 8, 12, 10, 12, 15 };
    double[] y_low = new double[] { 20, 22, 19, 24, 25 };
    double[] y_open = new double[] { 10, 16, 17, 15, 23 };
    double[] y_close = new double[] { 16, 19, 15, 22, 18 };
    //copy the x, y, y1, y2, and y3 data into the chart group
    c1Chart1.ChartGroups[0].ChartData[0].X.CopyDataIn(x);
    c1Chart1.ChartGroups[0].ChartData[0].Y.CopyDataIn(y_hi);
    c1Chart1.ChartGroups[0].ChartData[0].Y1.CopyDataIn(y_low);
    c1Chart1.ChartGroups[0].ChartData[0].Y2.CopyDataIn(y_open);
    c1Chart1.ChartGroups[0].ChartData[0].Y3.CopyDataIn(y_close);
    //assign the candle chart to the chart type
    c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Candle;
    //Make the rising candles transparent to show rising prices
    c1Chart1.ChartGroups[0].HiLoData.FillTransparent = true;
    //Declare the color for the lines
    c1Chart1.ChartGroups[0].ChartData.SeriesList[0].LineStyle.Color =
System.Drawing.Color.Red;
    //Increase the line thickness
    c1Chart1.ChartGroups[0].ChartData.SeriesList[0].LineStyle.Thickness = 2;
    c1Chart1.ChartGroups[0].ChartData.SeriesList[0].SymbolStyle.Size = 10;
    }
```

17.4 多图表指南

本主题将引导您以编程方式来一步步地在一个图表上添加一个 Bar 和 XYPlot 图表。图表使用条状图的 y-轴来表示平均的降雨量,X-轴线代表从五月到八月的月份字符串值。XYPlot 图表使用一个 y2-轴线来代表从五月到八月的每月的平均气温值，X-轴线代表从五月到八月的月份字符串值。

在您创建该图表之前下图用来展示该图表的效果。



多图表可以按照以下的步骤以编程方式创建。

1. 添加以下的代码来引入 **C1.Win.C1Chart** 命名空间。

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

2. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来生成一个简单的饼形图:

- Visual Basic

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
Dim cgroup As ChartGroup = c1Chart1.ChartGroups.Group0
cgroup.ChartType = Chart2DTypeEnum.Bar

'input the data through the series collection
Dim cdsc As ChartDataSeriesCollection = cgroup.ChartData.SeriesList
cdsc.Clear()
'remove default data
'create the series object from the collection and add data
Dim cds As ChartDataSeries = cdsc.AddNewSeries()
```

```
        ' Add Data for ChartGroup0, Bar chart
        Dim MonthNames As String() = {"March", "April", "May", "June", "July", "August"}
        Dim AvgPrecip As Double() = {3.17, 3.01, 3.8, 4.12, 3.96, 3.38}

        'create a label for the Bar chart data series
        cds.Label = "Avg. Precip"

        'Use the CopyDataIn method of the ChartDataArray object to copy the X and Y value data
into the data series
        cds.X.CopyDataIn(MonthNames)
        cds.Y.CopyDataIn(AvgPrecip)
        'create variable for chart area
        Dim carea As C1.Win.C1Chart.Area = c1Chart1.ChartArea
        'Set axes titles for the ChartGroup0 (Bar)
        carea.AxisX.Text = "Month"
        carea.AxisY.Text = "Avg. Precip (in)"

        'create and add the data for the XY chart in Group1
        Dim cgroup2 As ChartGroup = c1Chart1.ChartGroups.Group1
        cgroup2.ChartType = Chart2DTypeEnum.XYPlot

        'input the bar chart data of group1 through the series collection
        Dim cdsc2 As ChartDataSeriesCollection = cgroup2.ChartData.SeriesList
        'create the series object from the second collection and add data

        Dim cds2 As ChartDataSeries = cdsc2.AddNewSeries()
        cds2.X.CopyDataIn(MonthNames)
        cds2.Y.CopyDataIn(New Double() {49.5, 60.7, 70.8, 79.1, 82.7, 81.1})
        cds2.Label = "Avg. High Temp"

        'customize axes
        'create new font for the X, Y and Y2 axes
        Dim f As New Font("Arial", 10)
        carea.AxisX.Font = f
        carea.AxisY.Font = f
        Dim cdsc As ChartDataSeriesCollection = cgroup.ChartData.SeriesList
```

```
carea.AxisX.ForeColor = Color.SteelBlue
carea.AxisY.ForeColor = Color.SteelBlue
carea.AxisY2.ForeColor = Color.SteelBlue
carea.AxisY2.Font = f

'Set axes titles for the ChartGroup1 (XYPlot)
carea.AxisY2.Text = "Avg. High Temp (F)"

'set axis bounds
carea.AxisY.Min = 2.75
carea.AxisY2.Min = 30
carea.AxisY2.Max = 90
carea.AxisY.UnitMinor = 0.25

'rotate the axis X annotation
carea.AxisX.AnnotationRotation = 60

'add legend
c1Chart1.Legend.Visible = True

'add header
c1Chart1.Header.Visible = True
c1Chart1.Header.Text = "Pittsburgh Spring and Summer Season"
'add visual effects
Dim s As Style = carea.Style
s.ForeColor = Color.White
s.BackColor = Color.LightBlue
s.BackColor2 = Color.Azure
s.GradientStyle = GradientStyleEnum.Radial
c1Chart1.ColorGeneration = ColorGeneration.Flow
```

- C#

```
private void Form1_Load(object sender, EventArgs e)
{
ChartGroup cgroup = c1Chart1.ChartGroups.Group0;
cgroup.ChartType = Chart2DTypeEnum.Bar;
```

```
    //input the data through the series collection
    ChartDataSeriesCollection cdsc = cgroup.ChartData.SeriesList;
    cdsc.Clear(); //remove default data
    //create the series object from the collection and add data
    ChartDataSeries cds = cdsc.AddNewSeries();

    // Add Data for ChartGroup0, Bar chart
    string[] MonthNames = { "March", "April", "May", "June", "July", "August" };
    double[] AvgPrecip = { 3.17, 3.01, 3.80, 4.12, 3.96, 3.38};

    //create a label for the Bar chart data series
    cds.Label = "Avg. Precip";

    //Use the CopyDataIn method of the ChartDataArray object to copy the X and Y value data
into the data series
    cds.X.CopyDataIn(MonthNames);
    cds.Y.CopyDataIn(AvgPrecip);

    //create variable for chart area
    C1.Win.C1Chart.Area carea = c1Chart1.ChartArea;
    //Set axes titles for the ChartGroup0 (Bar)
    carea.AxisX.Text = "Month";
    carea.AxisY.Text = "Avg. Precip (in)";

    //create and add the data for the XY chart in Group1
    ChartGroup cgroup2 = c1Chart1.ChartGroups.Group1;
    cgroup2.ChartType = Chart2DTypeEnum.XYPlot;

    //input the bar chart data of group1 through the series collection
    ChartDataSeriesCollection cdsc2 = cgroup2.ChartData.SeriesList;

    //create the series object from the second collection and add data
    ChartDataSeries cds2 = cdsc2.AddNewSeries();
    cds2.X.CopyDataIn(MonthNames);
    cds2.Y.CopyDataIn(new double[] { 49.5, 60.7, 70.8, 79.1, 82.7, 81.1});
    cds2.Label = "Avg. High Temp";
```

```
//customize axes
//create new font for the X, Y and Y2 axes
Font f = new Font("Arial", 10);
carea.AxisX.Font = f;
carea.AxisY.Font = f;
carea.AxisX.ForeColor = Color.SteelBlue;
carea.AxisY.ForeColor = Color.SteelBlue;
carea.AxisY2.ForeColor = Color.SteelBlue;
carea.AxisY2.Font = f;

//Set axes titles for the ChartGroup1 (XYPlot)
carea.AxisY2.Text = "Avg. High Temp (F)";

//set axis bounds
carea.AxisY.Min = 2.75;
carea.AxisY2.Min = 30;
carea.AxisY2.Max = 90;
carea.AxisY.UnitMinor = .25;

//rotate the axis X annotation
carea.AxisX.AnnotationRotation = 60;
//add legend
c1Chart1.Legend.Visible = true;

//add header
c1Chart1.Header.Visible = true;
c1Chart1.Header.Text = "Pittsburgh Spring and Summer Season";

//add visual Effects
Style s = carea.Style;
s.ForeColor = Color.White;
s.BackColor = Color.LightBlue;
s.BackColor2 = Color.Azure;
s.GradientStyle = GradientStyleEnum.Radial;
c1Chart1.ColorGeneration = ColorGeneration.Flow;
}
```

18. 基于任务的 WinForms 图表帮助

基于任务的帮助假定您能够熟练地使用 Visual Studio .NET 进行编程。按照本帮助内容描述的步骤，您可以利用 C1Chart 属性来创建多种多样的图表类型。通过浏览 C1Chart 良好的接口定义和使用它的编辑器，您能够对 C1Chart 产品的能力有一个更好的印象。

18.1 旋转 Y-轴标题

如果您需要旋转 Y-轴标题，请像下述方式一样使用 **C1Chart.ChartArea.AxisY.Rotation** 枚举。

- Visual Basic

```
c1Chart2.ChartArea.AxisY.Rotation = RotationEnum.Rotate180
```

- C#

```
c1Chart2.ChartArea.AxisY.Rotation = RotationEnum.Rotate180;
```

18.2 旋转数据标签

如果您需要旋转一个数据标签 90 度。那么使用下述的代码:

- Visual Basic

```
dataSeries.DataLabel.Style.Rotation = RotationEnum.Rotate90
```

- C#

```
dataSeries。DataLabel。Style。Rotation = RotationEnum。Rotate90;
```

18.3 以一个百分率在饼状图表中显示数据标签

如果您需要在饼状图表中的一个给定点代表所有点的总计，那么像以下方式一样使用 {%YVAL}。

- Visual Basic

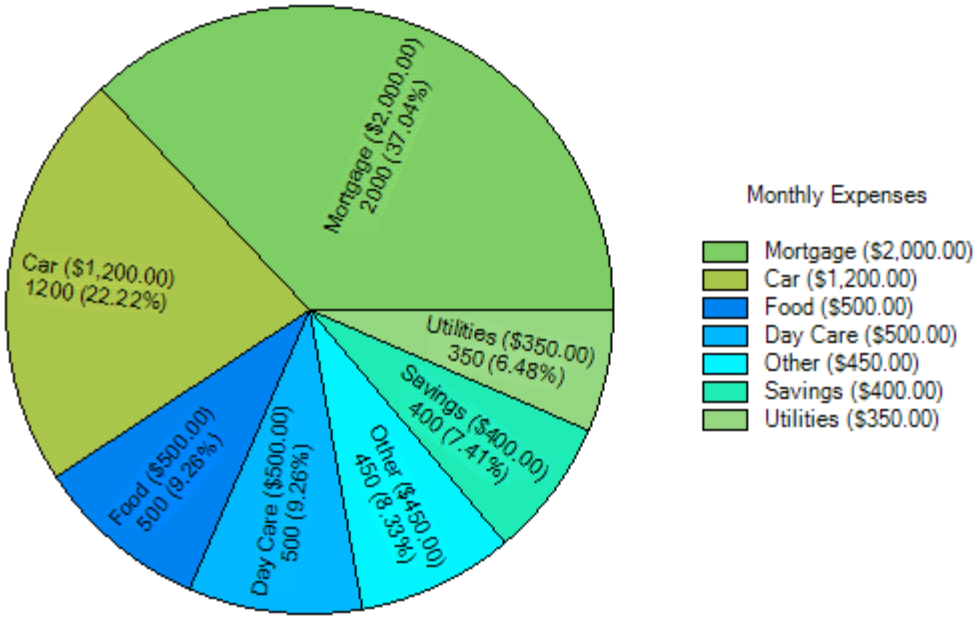
```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
    ' use light colors so all of the labels are easily read.
    C1Chart1.ColorGeneration = ColorGeneration.Flow
    ' maximize the ChartArea
    C1Chart1.ChartArea.Margins.SetMargins(0, 0, 0, 0)
    C1Chart1.ChartArea.Style.Border.BorderStyle = BorderStyleEnum.None
    ' Set chart type
    C1Chart1.ChartArea.Inverted = True
    C1Chart1.ChartGroups(0).ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Pie
    ' Clear previous data
    C1Chart1.ChartGroups(0).ChartData.SeriesList.Clear()
    ' Add Data
    Dim ProductNames As String() = {"Mortgage", "Car", "Food", "Day Care", "Other", "Savings",
"Utilities"}
    Dim PriceX As Integer() = {2000, 1200, 500, 500, 450, 400, 350}
    'get series collection
    Dim dscoll As ChartDataSeriesCollection = C1Chart1.ChartGroups(0).ChartData.SeriesList
    'populate the series
    For i As Integer = 0 To PriceX.Length - 1
    Dim series As ChartDataSeries = dscoll.AddNewSeries()
    'Add one point to show one pie
    series.PointData.Length = 1
    'Assign the prices to the Y Data series
    series.Y(0) = PriceX(i)
    'format the product name and product price on the legend
    series.Label = String.Format("{0} ({1:c})", ProductNames(i), PriceX(i))
    series.DataLabel.Text = "{#TEXT}" & vbCr & vbLf & "{#YVAL} ({%YVAL:0.00%})"
    series.DataLabel.Compass = LabelCompassEnum.RadialText
    series.DataLabel.Offset = -5
    series.DataLabel.Visible = True
    Next
    ' show pie Legend
    C1Chart1.Legend.Visible = True
    'add a title to the chart legend
    C1Chart1.Legend.Text = "Monthly Expenses"
    End Sub
```

- C#

```
    private void Form1_Load(object sender, EventArgs e)
    {
    // use light colors so all of the labels are easily read.
    c1Chart1.ColorGeneration = ColorGeneration.Flow;
    // maximize the ChartArea
    c1Chart1.ChartArea.Margins.SetMargins(0, 0, 0, 0);
    c1Chart1.ChartArea.Style.Border.BorderStyle = BorderStyleEnum.None;
    // Set chart type
    c1Chart1.ChartArea.Inverted = true;
    c1Chart1.ChartGroups[0].ChartType = C1.Win.C1Chart.Chart2DTypeEnum.Pie;
    // Clear previous data
    c1Chart1.ChartGroups[0].ChartData.SeriesList.Clear();
    // Add Data
    string[] ProductNames = { "Mortgage", "Car", "Food", "Day Care", "Other",
"Savings","Utilities" };
    int[] PriceX = {2000, 1200, 500, 500, 450, 400, 350 };
    //get series collection
    ChartDataSeriesCollection dscoll = c1Chart1.ChartGroups[0].ChartData.SeriesList;
    //populate the series
    for (int i = 0; i < PriceX.Length; i++)
    {
    ChartDataSeries series = dscoll.AddNewSeries();
    //Add one point to show one pie
    series.PointData.Length = 1;
    //Assign the prices to the Y Data series
    series.Y[0] = PriceX[i];
    //format the product name and product price on the legend
    series.Label = string.Format("{0} ({1:c})", ProductNames[i], PriceX[i]);
    series.DataLabel.Text = "{#TEXT}\r\n{#YVAL} ({%YVAL:0.00%})";
    series.DataLabel.Compass = LabelCompassEnum.RadialText;
    series.DataLabel.Offset = -5;
    series.DataLabel.Visible = true;
    }
    // show pie Legend
    c1Chart1.Legend.Visible = true;
    //add a title to the chart legend
    c1Chart1.Legend.Text = "Monthly Expenses";
    }
```

任务的图解说明

数据标签在饼状图表中代表点值和每一个切片的平均值。



18.4 给数据标签设置字体样式

如果您要设置数据标签的字体样式，例如设置为粗体，那么使用下述的代码:

- Visual Basic

```
label.Style.Font = New System.Drawing.Font("Arial", 10, System.Drawing.FontStyle.Bold)
```

- C#

```
label.Style.Font = new System.Drawing.Font("Arial", 10, System.Drawing.FontStyle.Bold);
```

18.5 给每一个条形图上方添加一个数据序列

如果您要给每一个条形图上方添加一个数据序列，那么使用下述的代码:

- Visual Basic

```
Dim sc As ChartDataSeriesCollection =
C1Chart1.ChartGroups(0).ChartData.SeriesList
Dim i As Integer
For i = 0 To sc.Count - 1
With sc(i).DataLabel
.Visible = True
.Compass = LabelCompassEnum.North
.Text = "{#YVAL}"
End With
Next
End Sub
```

- C#

```
{
ChartDataSeriesCollection sc = C1Chart1.ChartGroups(0).ChartData.SeriesList;
int i = 0;
for (i = 0; i <= sc.Count - 1; i++) {
{
sc(i).DataLabel.Visible = true;
sc(i).DataLabel.Compass = LabelCompassEnum.North;
sc(i).DataLabel.Text = "{#YVAL}";
}
}
}
```

18.6 标签折行

您可以使用折行字符(/n)来手动给标签中的文字折行。

作为使用折行字符(/n)来给标签中的文字折行的示例，请参见**以一个百分率在饼状图表中显示数据标签**(293 页)。

18.7 添加一个透明标签来调整值和 x-轴之间的间距

下述代码给图表添加一个箭头形状外观的透明值标签。因为它是透明的，所以它对最终用户是不可见的。通过改变 ValueLabel。MarkerSize 属性，您可以调整值和 x-轴之间的间距。

- Visual Basic

```
c1Chart1.ChartArea.AxisX.AnnoMethod = C1.Win.C1Chart.AnnotationMethodEnum.Mixed
Dim vl As ValueLabel = c1Chart1.ChartArea.AxisX.ValueLabels.AddNewLabel()
vl.Appearance = ValueLabelAppearanceEnum.TriangleMarker
vl.MarkerSize = 50
vl.Color = Color.Transparent
vl.NumericValue = 2
```

- C#

```
c1Chart1.ChartArea.AxisX.AnnoMethod = C1.Win.C1Chart.AnnotationMethodEnum.Mixed;
ValueLabel vl = c1Chart1.ChartArea.AxisX.ValueLabels.AddNewLabel();
vl.Appearance = ValueLabelAppearanceEnum.TriangleMarker;
vl.MarkerSize = 50;
vl.Color = Color.Transparent;
vl.NumericValue = 2;
```

18.8 显示图表图例和图表表头

表头和图例都含有 **Compass** 和 **Location** 属性，你可以使用其中一个或者全部两个属性来

调整位置。 **Compass** 属性可以通过 **ComapssEnum** 属性设置为东，南，西，北。默认地，表头和图例将位于罗盘区域的中间位置。 **Location** 属性可以被设置为一个 **System.Drawing.Point** 类型的值，它接受一个 x 坐标和 y 坐标。x 坐标和 y 坐标都可以被设置为-1，这时会进行自动的位置选择。

显示表头

下述的示例将图表表头放置在图表上部。点坐标可以被精确的校对(使用了自动位置选择) 。

- Visual Basic

```
c1Chart1.Header.Text = "My Chart Header"
c1Chart1.Header.Compass = C1.Win.C1Chart.CompassEnum.North
c1Chart1.Header.Location = New Point(-1, -1)
c1Chart1.Header.Visible = True
```

- C#

```
this.c1Chart1.Header.Text = "My Chart Header";
this.c1Chart1.Header.Compass = C1.Win.C1Chart.CompassEnum.East;
this.c1Chart1.Header.Location = new Point(-1, -1);
this.c1Chart1.Header.Visible = true;
```

显示图例

下述的示例将图表图例放置在图表上部。点坐标可以被精确的校对(使用了自动位置选择)。

注意:请记住-1 仅仅使用了罗盘设置的默认坐标，一个或者两个坐标都可以含有绝对位置。请确保点对位置是有意义的。例如，您可能不想给某个罗盘方向为北面的对象设置一个很大的 y 值，因为图表会自动压缩以适应表头。

- Visual Basic

```
c1Chart1.Legend.Text = "My Chart Legend"
c1Chart1.Legend.Compass = C1.Win.C1Chart.CompassEnum.West
c1Chart1.Legend.Location = new Point(-1, -1)
c1Chart1.Legend.Visible = True
```

- C#

```
this.c1Chart1.Legend.Text = "My Chart Legend";
this.c1Chart1.Legend.Compass = C1.Win.C1Chart.CompassEnum.West;
this.c1Chart1.Legend.Location = new Point(-1, -1);
this.c1Chart1.Legend.Visible = true;
```

18.9 垂直地显示图例

如果您需要这样做的话，使用 C1Chart 的 Style 属性。一个名为’Orientation’的属性可以被设置为 horizontal 或者 vertical.请确保您将其设置为’vertical’。

18.10 通过点击来获取一个饼状图的切片

您可以使用下述代码来通过点击来获取一个饼状图的切片:

- Visual Basic

```
Dim seriesIndex = 0
Dim pointIndex = 0
Private Sub C1Chart1_MouseMove(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles C1Chart1.MouseMove
Dim si, pi, d As Integer
If C1Chart1.ChartGroups(0).CoordToDataIndex(e.X, e.Y, _
C1.Win.C1Chart.CoordinateFocusEnum.XandYCoord, si, pi, d) Then
seriesIndex = si
pointIndex = pi
End If
End Sub
Private Sub C1Chart1_MouseClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles C1Chart1.MouseClick
MsgBox(C1Chart1.ChartGroups(0).ChartData(seriesIndex).Y(pointIndex))
End Sub
```

- C#

```
var seriesIndex = 0;
var pointIndex = 0;
private void // ERROR: Handles clauses are not supported in C#
C1Chart1_MouseMove(System.Object sender, System.Windows.Forms.MouseEventArgs e)
{
int si = 0;
int pi = 0;
int d = 0;
if (C1Chart1.ChartGroups(0).CoordToDataIndex(e.X, e.Y,
C1.Win.C1Chart.CoordinateFocusEnum.XandYCoord, si, pi, d)) {
seriesIndex = si;
pointIndex = pi;
}
}
private void // ERROR: Handles clauses are not supported in C#
C1Chart1_MouseClick(System.Object sender, System.Windows.Forms.MouseEventArgs e)
{
Interaction.MsgBox(C1Chart1.ChartGroups(0).ChartData(seriesIndex).Y(pointIndex));
}
```

18.11 创建标记

如果您需要给标签添加一个标记,那么请使用下述代码:

- Visual Basic

```
c1Chart1.ChartArea.AxisX.AnnoMethod = C1.Win.C1Chart.AnnotationMethodEnum.Mixed
Dim vl As ValueLabel = c1Chart1.ChartArea.AxisX.ValueLabels.AddNewLabel()
vl.Appearance = ValueLabelAppearanceEnum.TriangleMarker
vl.MarkerSize = 50
vl.Color = Color.Transparent
vl.NumericValue = 2
```

- C#

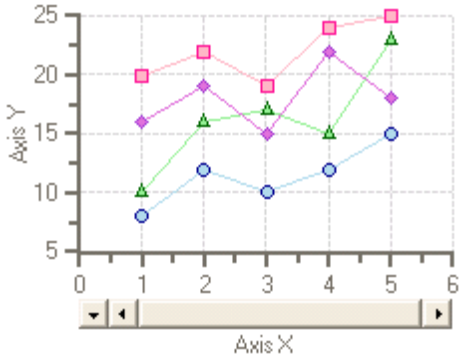
```
C1Chart1.ChartArea.AxisX.AnnoMethod = C1.Win.C1Chart.AnnotationMethodEnum.Mixed
Dim markerX As C1.Win.C1Chart.ValueLabel =
C1Chart1.ChartArea.AxisX.ValueLabels.AddNewLabel()
markerX.NumericValue = 3
markerX.Moveable = True
markerX.MarkerSize = 15
markerX.GridLine = True
markerX.Color = Color.Blue
markerX.Appearance = C1.Win.C1Chart.ValueLabelAppearanceEnum.ArrowMarker
```

18.12 给 X-轴和 Y-轴添加滚动条

下述的步骤介绍了如何在图表中给 X-轴和 Y-轴添加一个滚动条。

使用属性窗体:

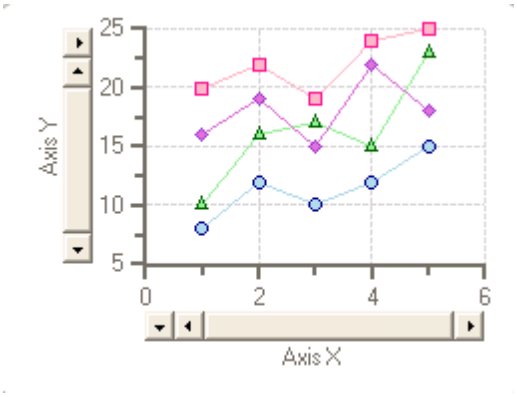
1. 在 **C1Chart** 属性窗体中，展开 ChartArea 节点，然后再展开 **AxisX** 节点。
2. 展开 **ScrollBar** 节点并设置其 Visible 属性为 **True**。
3. 设置 **AxisX.Min** 属性为 0，并将 AxisX.Max 属性设置为 6。滚动条将在最大值和最小值之间出现。
4. 默认的 **AxisX** 滚动条出现在 X 轴的中间，并出现在最大值和最小值之间。



5. 在 **C1Chart** 属性窗体中，展开 **AxisY** 节点。

6. 展开 **ScrollBar** 节点并设置其 Visible 属性为 **True**。
   设置 AxisY.Min 属性为 6，并将 AxisY.Max 属性设置为 25。滚动条将在最大值和最小值之间出现。

默认的 AxisY 滚动条出现在 Y 轴的中间，并出现在最大值和最小值之间。



使用代码:

使用下述代码在图表中给 X-轴和 Y-轴添加一个滚动条。

- Visual Basic

```
' Setup the AxisX scroll bar
'Note, that the cd variable represents the ChartData
c1Chart1.ChartArea.AxisX.ScrollBar.Min = cd.MinY
c1Chart1.ChartArea.AxisX.ScrollBar.Max = cd.MaxY
c1Chart1.ChartArea.AxisX.ScrollBar.Appearance = ScrollBarAppearanceEnum.Normal
c1Chart1.ChartArea.AxisX.ScrollBar.Visible = True
c1Chart1.ChartArea.AxisX.ScrollBar.Alignment = StringAlignment.Center
' Setup the AxisY scroll bar
c1Chart1.ChartArea.AxisY.ScrollBar.Min = cd.MinY
c1Chart1.ChartArea.AxisY.ScrollBar.Max = cd.MaxY
c1Chart1.ChartArea.AxisY.ScrollBar.Appearance = ScrollBarAppearanceEnum.Normal
c1Chart1.ChartArea.AxisY.ScrollBar.Visible = True
c1Chart1.ChartArea.AxisY.ScrollBar.Alignment = StringAlignment.Center
```

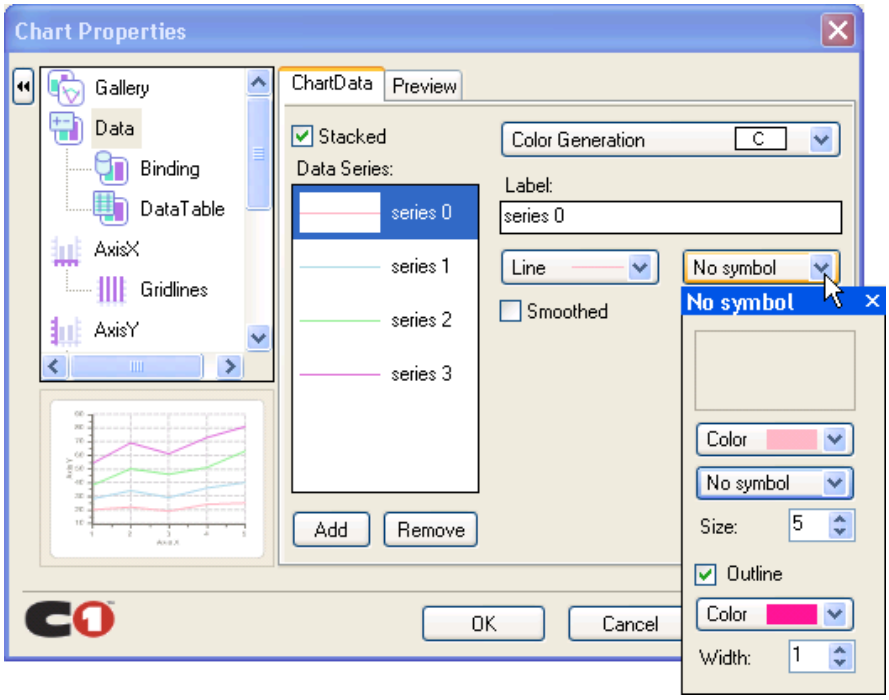
- C#

```
// Setup the AxisY scroll bar
//Note, that the cd variable represents the ChartData
c1Chart1.ChartArea.AxisX.ScrollBar.Min = cd.MinY;
c1Chart1.ChartArea.AxisX.ScrollBar.Max = cd.MaxY;
c1Chart1.ChartArea.AxisX.ScrollBar.Appearance = ScrollBarAppearanceEnum.Normal;
c1Chart1.ChartArea.AxisX.ScrollBar.Visible = true;
c1Chart1.ChartArea.AxisX.ScrollBar.Alignment = StringAlignment.Center;
// Setup the AxisY scroll bar
c1Chart1.ChartArea.AxisY.ScrollBar.Min = cd.MinY;
c1Chart1.ChartArea.AxisY.ScrollBar.Max = cd.MaxY;
c1Chart1.ChartArea.AxisY.ScrollBar.Appearance = ScrollBarAppearanceEnum.Normal;
c1Chart1.ChartArea.AxisY.ScrollBar.Visible = true;
c1Chart1.ChartArea.AxisY.ScrollBar.Alignment = StringAlignment.Center;
```

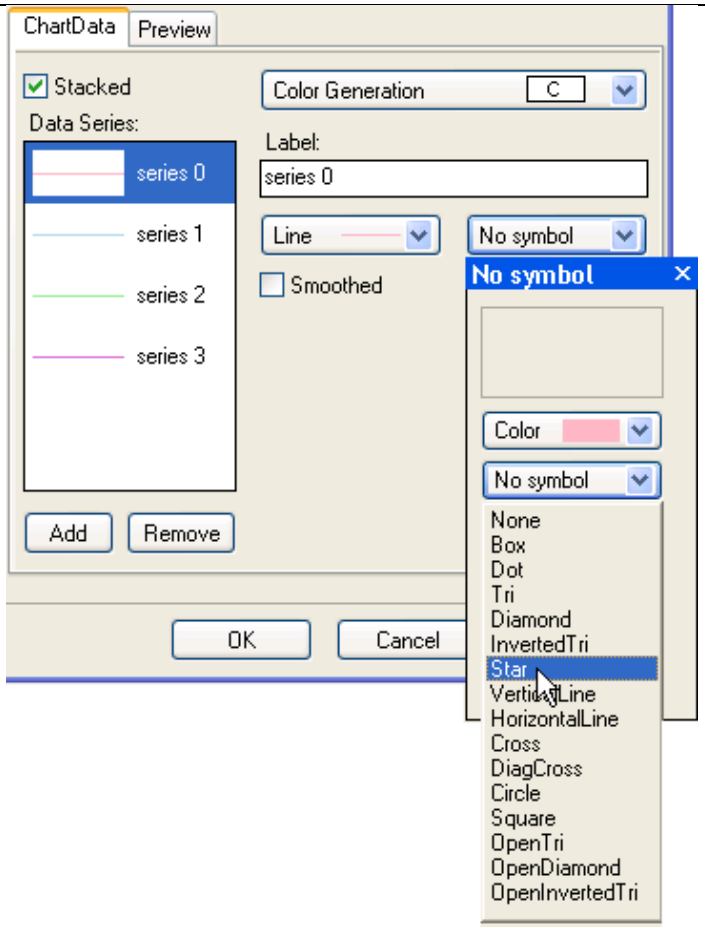
18.13 给数据序列添加符号

如果您需要使用 C1Chart 属性窗体给一个数据序列添加符号，那么按照以下步骤完成:

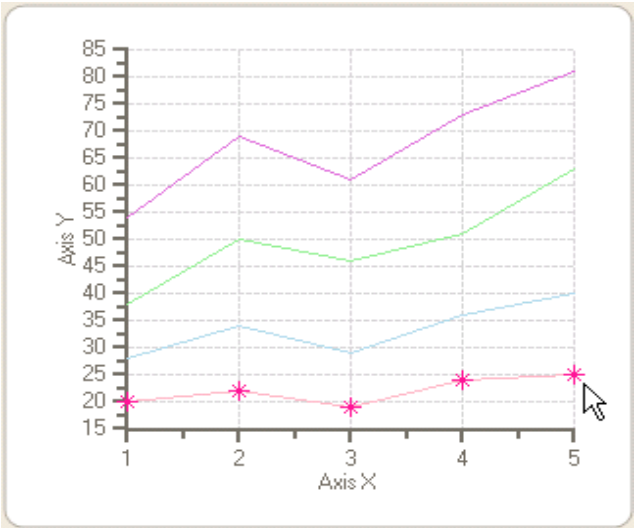
1. 在 **C1Chart** 控件上右键点击。并从其上下文菜单中选择 **Chart Properties**。
2. 在图表属性编辑器中选择 **Data** 元素。
3. 从 **Data Series** 下拉列表中选择一个数据序列。
4. 点击 **Symbol** 下拉箭头来打开可用符号的列表。



5. 选择符号形状。例如星形，并添加它到一个数据序列中。



6. 点击 **Color** 下拉箭头并选择 **Deep Pink**。
7. 反选 **Outline** 复选框，来去除星形的 **Deep Pink** 轮廓。
8. 从 5 到 8 增加星形的大小。
9. 点击 **OK**，您就完成了符号外观的修改。



18.14 给图表元素添加提示信息

以下的主题描述了如何给 C1Chart 的表头，页尾和数据序列元素添加提示信息。

18.14.1 给数据序列中的点添加提示信息

如果要给数据序列添加提示信息，那么按照以下的步骤完成:

1. 给窗体添加一个 **C1Chart** 控件。
2. 在窗体上右键点击，并选择查看代码来查看代码文件。然后添加以下的代码来声明 **C1.Win.C1Chart 命名空间。**

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

3. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来创建一个数据序列，并给它添加提示信息。

- Visual Basic

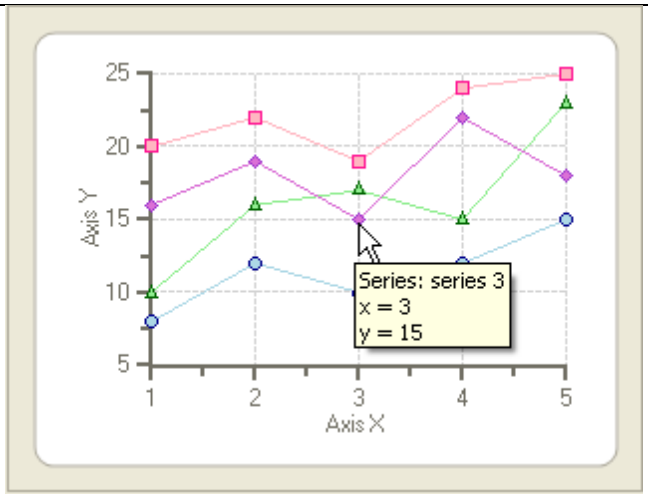
```
C1Chart1.ToolTip.Enabled = True
Dim sc As ChartDataSeriesCollection = C1Chart1.ChartGroups(0).ChartData.SeriesList
For Each ds As ChartDataSeries In sc
ds.TooltipText = "Series: {#TEXT}" + ControlChars.Cr + ControlChars.Lf + "x = {#XVAL}" +
ControlChars.Cr + ControlChars.Lf + "y = {#YVAL}"
Next ds
```

- C#

```
ChartDataSeriesCollection sc =
c1Chart1.ChartGroups[0].ChartData.SeriesList;
foreach (ChartDataSeries ds in sc)
ds.TooltipText = "Series: {#TEXT}" + '\r' + '\n' + "x = {#XVAL}" + '\r' + '\n' + "y = {#YVAL}";
// Enable tooltip
c1Chart1.ToolTip.Enabled = true;
```

以下是本主题的插图:

当您在运行时将鼠标停留在数据序列中的点上时，提示信息将会出现。如下图的效果:



18.14.2 给图表的表头和页尾添加提示信息

如果要给图表的表头和页尾添加提示信息，那么按照以下的步骤完成:

1. 给窗体添加一个 **C1Chart** 控件。
2. 在窗体上右键点击，并选择查看代码来查看代码文件。然后添加以下的代码来声明 **C1.Win.C1Chart 命名空间。**

- Visual Basic

```
Imports C1.Win.C1Chart;
```

- C#

```
using C1.Win.C1Chart;
```

3. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来给图表的表头和页尾添加提示信息。

- Visual Basic

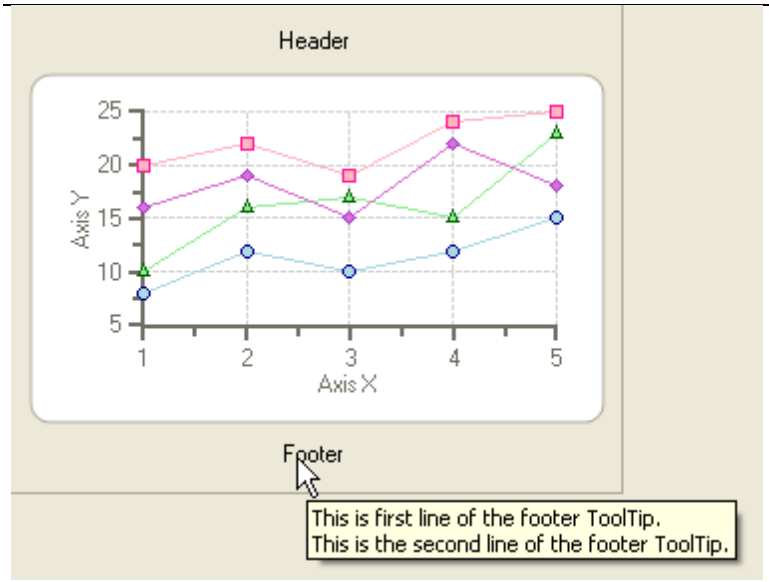
```
'Enable tooltip
c1Chart1.ToolTip.Enabled = True
c1Chart1.Header.TooltipText = "This is header tooltip." + ControlChars.Cr + ControlChars.Lf + "Second line."
c1Chart1.Footer.TooltipText = "This is footer tooltip." + ControlChars.Cr + ControlChars.Lf + "Second line."
```

- C#

```
//Enable tooltip
c1Chart1.ToolTip.Enabled = true;
c1Chart1.Header.TooltipText = "This is header tooltip.";
c1Chart1.Footer.TooltipText = "This is first line of the footer ToolTip.\nThis is the second line of the footer ToolTip.";
```

以下是本主题的插图:

当您在运行时将鼠标停留在图表的表头和页尾上时，提示信息将会出现。如下图的效果:



18.14.3 给图表的轴添加提示信息

如果要给图表的轴添加提示信息，那么按照以下的步骤完成:

1. 给窗体添加一个 **C1Chart** 控件。
2. 在窗体上右键点击，并选择查看代码来查看代码文件。然后添加以下的代码来声明 **C1.Win.C1Chart 命名空间。**
   - Visual Basic

```
Imports C1.Win.C1Chart;
```

   - C#

```
using C1.Win.C1Chart;
```

3. 双击窗体并在 **Form1_Load** 事件处理函数中添加以下的逻辑来添加轴。
   - Visual Basic

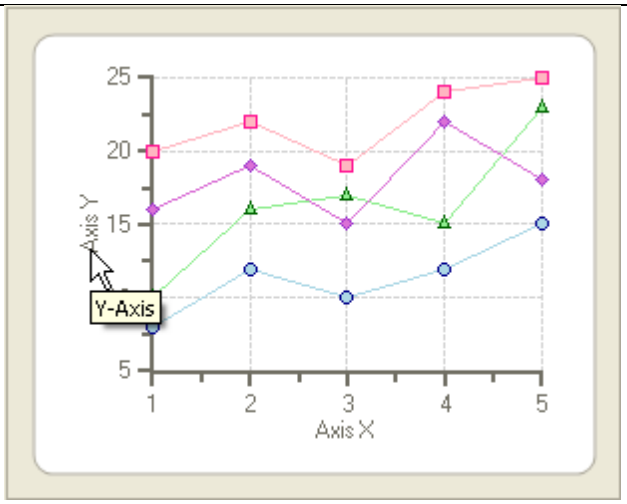
```
'Enable tooltip
C1Chart1.ToolTip.Enabled = True
C1Chart1.ChartArea.AxisX.TooltipText = "X-Axis"
C1Chart1.ChartArea.AxisY.TooltipText = "Y-Axis"
```

   - C#

```
//Enable tooltip
c1Chart1.ToolTip.Enabled = true;
c1Chart1.ChartArea.AxisX.TooltipText = "X-Axis";
c1Chart1.ChartArea.AxisY.TooltipText = "Y-Axis";
```

以下是本主题的插图:

当您在运行时将鼠标停留在图表的 X-轴或者 Y-轴上时，提示信息将会出现。如下图的效果:



18.15 给图表元素添加视觉效果

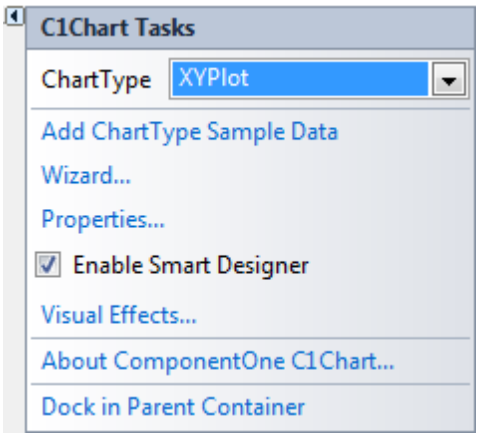
本主题提供了使用视觉效果设计器来增强图表的表头，页尾和数据序列元素的效果。

18.15.1 访问视觉效果设计器

可以通过使用 **C1Chart Tasks** 菜单(智能标记)，**C1Chart** 上下文菜单，或者属性表格编辑器来访问视觉效果设计器。

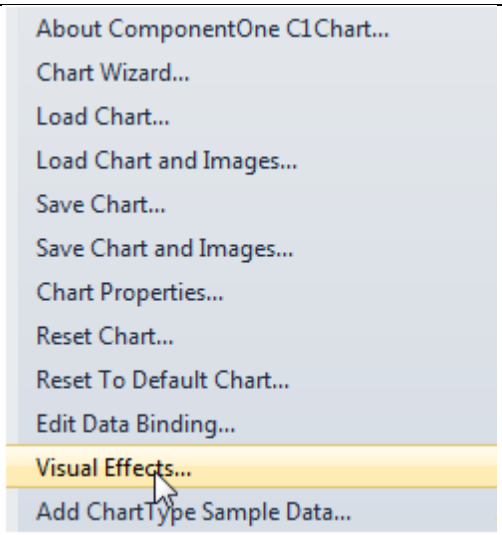
C1Chart Tasks 菜单

点击 C1Chart 的右下角的智能标记(▶)来打开 **C1Chart Tasks** 菜单，并且选择 Visual Effects。



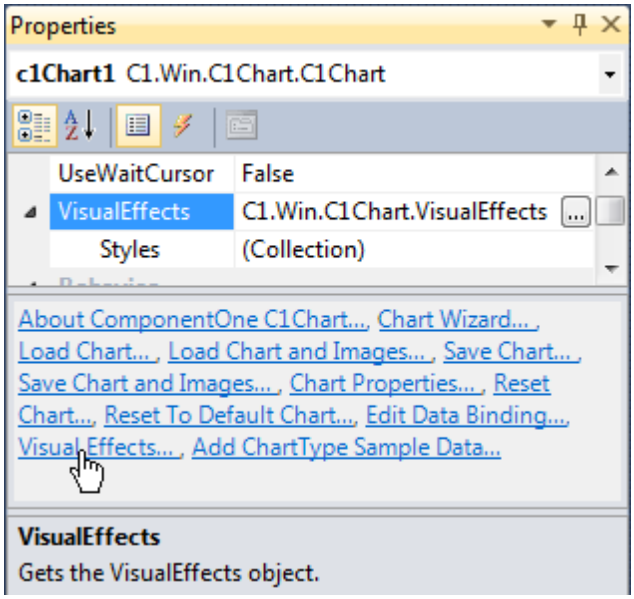
上下文菜单

在 **Chart2D** 控件上右键点击，然后在上下文菜单中选择 **Visual Effects**。



属性窗体

在 **Chart2D** 控件上右键并选择 **Properties**。点击 **ellipsis** 按钮来在属性窗体中找到 VisualEffects ,或者在 Visual Studio .NET 的属性窗体的底部的 Action 下拉列表区域点击 Visual Effects。下图显示了 C1Chart 属性窗体中的 VisualEffects 元素。



18.15.2 自定义表头和页尾

本章节说明了如何添加光模式，形状，阴影和预设样式到图表的表头和页尾中。除了光效果任务外 ,本章节还说明了如何通过改变光的色调或者增加减少亮度和饱和度来扩展图表表头既存的颜色。

18.15.2.1 给图表表头和页尾增加一个光模式

您可以通过设置 Scale 属性的值小于 1 来在一个图表元素中显示重复光模式。随着 Scale 属性值的减小，光模式重复的更多。Scale 属性的取值范围是从 0 到 1。

如果您需要给图表的表头和图表的页尾添加一个光模式，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开**视觉效果**设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 **Header** 和 **Footer** 的复选框。
4. 点击 **Parameters** 选项卡，设置光的 Shape 属性为 Rectangle。
5. 选择 **Scale** 属性并将 Scale 游标滑动到 0.20 或者直接在文本框中输入.20。

注意:Scale 属性仅对矩形光渐进有效。

矩形模式会在图表表头元素中重复。

矩形模式会在图表页尾元素中重复。

为了给图表的表头和页尾中的文字添加一个光模式。请完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
2. 在 **Available Elements** 下拉列表中，反选紧挨着 **Header** 和 **Footer** 的复选框。然后点击紧挨着 Header.Text 和 Footer.Text 的复选框。
3. 点击 **Parameters** 选项卡，设置光的 Shape 属性为 Rectangle。
4. 选择 **Scale** 属性并将 Scale 游标滑动到 0.20 或者直接在文本框中输入.20。

注意:Scale 属性仅对矩形光渐进有效。

矩形模式会在图表表头的文字元素中重复。

矩形模式会在图表页尾的文字元素中重复。

18.15.2.2 给图表表头和页尾增加一个光形状

如果您需要给图表的表头和图表的页尾添加一个光形状，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 **Header** 复选框，然后点击紧挨着 **Footer** 的复选框。

4. 点击 **Parameters** 选项卡，设置光的 Shape 属性为 Ellipse。预览面板会显示椭圆形状。

注意:两个新增的属性(Shift 和 Size)在椭圆形状中出现。

5. 为了减少椭圆形状的尺寸，选择 **Size** 属性，并向左滑动 **Size** 游标到 0.3。
6. 为了将椭圆形状的位置从中间移动到左边。选择 **Shift** 属性，并将 **Shift** 游标移动到左边。随着你向左移动游标，椭圆光形状会随之改变。

椭圆光形状会出现在图表表头元素的角落中。

椭圆光形状会出现在图表页尾元素的角落中。

如果您需要给图表的表头和图表的页尾的文字元素中添加一个光形状 ,那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见添**加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见添**加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中， 反选紧挨着 **Header** 和 **Footer** 的复选框。然后点击紧挨着 Header.Text 和 Footer.Text 的复选框
4. 点击 **Parameters** 选项卡，设置光的 Shape 属性为 Ellipse。

椭圆形状会出现在图表表头的文字中

椭圆形状会出现在图表页尾的文字中

18.15.2.3 给图表表头和页尾增加一个预设样式

如果您需要给图表的表头和图表的页尾添加一个预设样式，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 **Header** 复选框，然后点击紧挨着 Footer 的复选框。
4. 点击 **Presets** 选项卡 ,选择第一行的第三种样式。预览面板会显示选中的预设样式。

选中的预设样式会出现在图表表头中。

选中的预设样式会出现在图表页尾中。

如果您需要给图表的表头和图表的页尾的文字元素中添加一个预设样式，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，反选紧挨着 **Header** 和 **Footer** 的复选框。然后点击紧挨着 Header.Text 和 Footer.Text 的复选框。
4. 点击 **Presets** 选项卡，选择第一行的第三种样式。预览面板会显示选中的预设样式。

选中的预设样式会出现在图表表头的文字中。

选中的预设样式会出现在图表页尾的文字中。

18.15.2.4 给图表表头和页尾增加一个阴影

如果您需要给图表的表头和图表的页尾添加一个阴影，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 **Header** 复选框，然后点击紧挨着 Footer 的复选框。
4. 点击 **Parameters** 选项卡，并选择位于阴影组的 **Offset** 属性。
5. 向右滑动 Offset 游标到 2.5 或者在文本框中输入 **2.5**
   预览面板会在格子中显示阴影。
6. 为了让阴影看起来更加的不透明，选择 Transparency 并滑动游标到 160。
   预览面板会显示更暗的阴影。

一个阴影会出现在图表的表头中。

一个阴影会出现在图表的页尾中。

如果您需要给图表的表头和图表的页尾的文字元素中添加一个阴影，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中， 反选紧挨着 **Header** 和 **Footer** 的复选框。
   然后点击紧挨着 Header.Text 和 Footer.Text 的复选框。
4. 点击 **Parameters** 选项卡，并选择位于阴影组的 **Offset** 属性。
5. 向右滑动 **Offset** 游标到 2.5 或者在文本框中输入 2.5
   预览面板会在格子中显示阴影。
6. 为了让阴影看起来更加的不透明，选择 Transparency 并滑动游标到 160。
   预览面板会显示更暗的阴影。

一个阴影会出现在图表的表头的文字中。

一个阴影会出现在图表的页尾的文字中。

18.15.2.5 给图表表头和页尾调整光的焦点

如果您需要给图表的表头和图表的页尾调整光的焦点，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 **Header** 复选框，然后点击紧挨着 Footer 的复选框。
4. 点击 **Parameters** 选项卡，并选择位于光分组的 Gradient 属性为 Triangle。
   Focus 属性被添加到光分组中，其默认值是 0.1。
5. 选择 **Focus** 属性并滑动游标到.55，这样使得两面都会出现光。
   预览面板会在格子中显示光焦点的位置

三重光会聚焦在图表表头的两个角落上。

三重光会聚焦在图表页尾的两个角落上。

如果您需要给图表的表头和图表的页尾的文字元素调整光的焦点，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问视**觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，反选紧挨着 **Header** 和 **Footer** 的复选框。然后点击紧挨着 Header.Text 和 Footer.Text 的复选框。
4. 点击 **Parameters** 选项卡，并选择位于光分组的 Gradient 属性为 Triangle。
   Focus 属性被添加到光分组中，其默认值是 0.1。
5. 选择 **Focus** 属性并滑动游标到.55，这样使得两面都会出现光。
   预览面板会在格子中显示光焦点的位置

三重光会聚焦在图表表头的文字元素的开始和结束位置上。

三重光会聚焦在图表页尾的文字元素的开始和结束位置上。

18.15.2.6 使用颜色滑块来增强图表的表头和图表的页尾的既存颜色。

如果您需要增强图表的表头和图表的页尾的青色背景色，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中，点击紧挨着 Header 复选框，然后点击紧挨着 Footer 的复选框。在预览面板中青色的显示略有不同，因为默认的配色方案会应用到表头既存的颜色中。
4. 点击 **Color** 选项卡，然后滑动 HueShift 游标的来设置其值为 14，这样使得其有一个更蓝的色调。
5. 向右滑动 Saturation 游标并停留在 100 的位置。饱和度增加到 100%，这样使得绿色色调变得更加的生动和鲜艳。

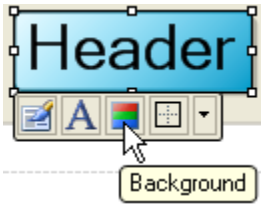
6. 向右滑动 Brightness 游标并停留在-22 的位置。这样略微地增加了色调的饱和度。

新颜色会出现在图表的表头元素中。

新颜色会出现在图表的页尾元素中

作为一个可选任务，如果您想进一步地定制颜色，请完成以下步骤。

1. 关闭视觉效果设计器，并在窗体上选择 Chart Header。表头工具栏就会出现。



2. 点击 Background 按钮，并从颜色下拉列表中选择 Web 选项卡。
3. 选择 Bule 来给既存颜色添加一项绿颜色。对图表的页尾也重复 1-3 步骤。

图表的表头的新的颜色看起来如下

图表的页尾的新的颜色看起来如下

如果您需要为图表的表头和图表的页尾创建自定义颜色的，那么需要完成以下步骤:

1. 添加一个图表的表头和图表的页尾。然后设置他们的 ForeColor 属性为 **Navy，BackColor** 属性为 **DeepSkyBlue**，并设置它们的文字为粗体。
   关于添加图表的表头的更多信息，请参见**添加一个图表表头**(330 页)。关于添加图表的页尾的更多信息，请参见**添加一个图表页尾**(329 页)。
2. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
3. 在 **Available Elements** 下拉列表中， 反选紧挨着 **Header** 和 **Footer** 的复选框。然后点击紧挨着 Header.Text 和 Footer.Text 的复选框。
4. 向右滑动 **Saturation** 游标并停留在 100 的位置。饱和度增加到 100%。
5. 5 向右滑动 Brightness 游标并停留在 24 的位置。这样会使得色调变得更加的明亮。
6. 然后滑动 HueShift 游标到最左边，这样使得其值为 0。

新颜色会出现在图表的表头元素的文字中。

新颜色会出现在图表的页尾元素的文字中

如果您想把颜色修改为粉色，仅需简单地滑动向右 HueShift 游标并停留在 66 的位置。图表的表头和图表的页尾的文字看起来如下:

18.15.3 自定义数据序列

本章节说明了如何添加光模式，形状，阴影和预设样式到图表的数据序列中。除了光效果任务外，本章节还说明了如何通过使用色调，明亮度，饱和度来自定义颜色。本示例使用饼状图表，并使用数据序列的默认颜色。

18.15.3.1 给图表数据序列增加一个光模式

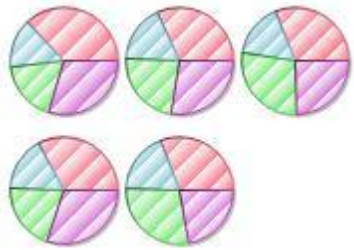
您可以通过设置 Scale 属性的值小于 1 来在一个图表元素中显示重复光模式。随着 Scale 属性值的减小，光模式重复的更多。Scale 属性的取值范围是从 0 到 1。

如果您需要给图表的数据序列添加一个光模式，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 点击 **Parameters** 选项卡，设置光的 **Shape** 属性为 **Rectangle**。
4. 选择 **Scale** 属性并将 Scale 游标滑动到 0.20 或者直接在文本框中输入.20。

注意:**Scale** 属性仅对矩形光渐进有效。

矩形模式会在饼状图表的数据序列中重复。



18.15.3.2 给图表数据序列增加一个光形状

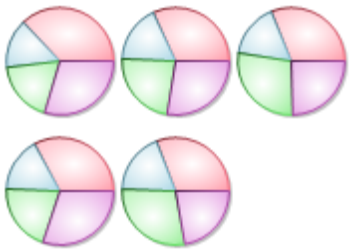
如果您需要给图表数据序列添加一个光形状，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见**访问视觉效果设计器**(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 点击 **Parameters** 选项卡，设置光的 **Shape** 属性为 **Ellipse**。预览面板会显示椭圆形状。

注意:两个新增的属性(Shift 和 Size)在椭圆形状中出现。

4. 为了增大椭圆形状的尺寸，选择 **Size** 属性，并向右滑动 **Size** 游标到 1。
5. 为了增加光的强度。选择 Intensity 属性。并向右滑动 Intensity 游标到 1，这样使得光的强度增加。

椭圆光形状会出现在数据序列的中间。

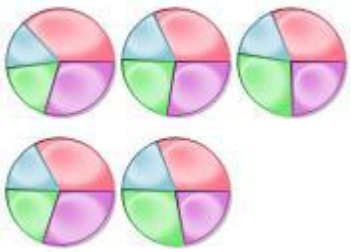


18.15.3.3 给图表数据序列添加一个预设样式

如果您需要给图表数据序列添加一个预设样式，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 点击 **Presets** 选项卡，然后选择第一行的第三个样式。

预览面板会显示选中的预设样式。选中的预设样式会出现在数据序列上。

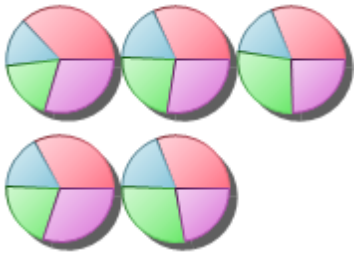


18.15.3.4 给数据序列添加阴影

如果您需要给图表数据序列添加阴影，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问**视觉效果设计器**(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 点击 **Parameters** 选项卡，并选择位于阴影组的 Offset 属性。
4. 向右滑动 **Offset** 游标到 3 或者在文本框中输入 3。
   预览面板会在格子中显示阴影。
5. 为了让阴影看起来更加的不透明，选择 Transparency 并滑动游标到 160。
   预览面板会显示更暗的阴影。

一个阴影会出现在数据序列中。

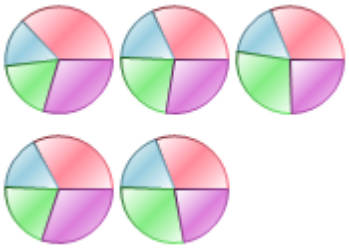


18.15.3.5 改变图表数据序列中光的焦点

如果您需要改变图表序列中光的焦点，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问视觉效果设计器(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 点击 **Parameters** 选项卡，并选择位于光分组的 **Gradient** 属性为 **Triangle**。
4. Focus 属性被添加到光分组中。
5. 选择 Focus 属性并滑动游标到.5，这样使得光的焦点出现在数据序列的中央
   选中 Intensity 属性并输入 1.0，来增加光的强度。

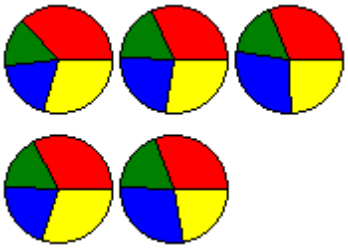
预览面板会在格子中显示光焦点的位置,三重光会聚焦在数据序列元素的中央。



18.15.3.6 使用颜色滑块来增强图表数据序列的既存颜色

如果您需要为图表的数据序列自定义颜色，那么需要完成以下步骤:

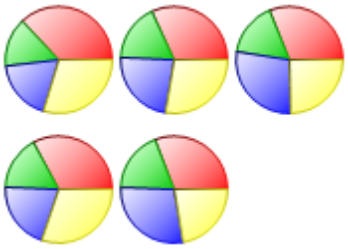
1. 打开**图表属性**设计器。
   关于如何访问图表属性设计器的更多细节，请参见访问图表属性设计器(139 页)。
2. 设置序列 0 为红色，序列 1 为蓝色，序列 2 为绿色。序列 3 为黄色。关于使用**图表属性**编辑器来给数据序列应用颜色的更多信息，请参见改变数据序列的外观(344 页)。每一个序列的颜色看起来如下:



3. 打开**视觉效果**设计器。

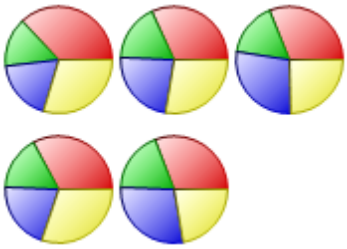
关于如何访问视觉效果设计器的更多细节，请参见访问视觉效果设计器(307 页)。

4. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。 这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。



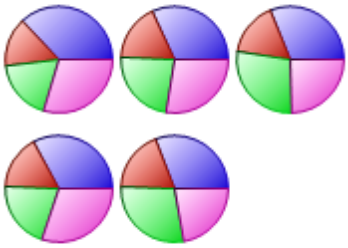
5. 点击 **Color** 选项卡，滑动 Brightness 游标并停留在-55 的位置。这样颜色看其来更暗。
6. 向右滑动 Saturation 游标并停留在 100 的位置。饱和度增加到 100%，这样使得绿色色调变得更加的生动和鲜艳。

数据序列的原有颜色得到了增强。



作为一个可选任务，为了改变数据序列的颜色，完成以下步骤。

- 点击 Color 选项卡，然后滑动 HueShift 游标的来设置其值为 247，注意到当你移动游标时颜色发生了变化。



18.15.3.7 增加数据序列中的符号的尺寸

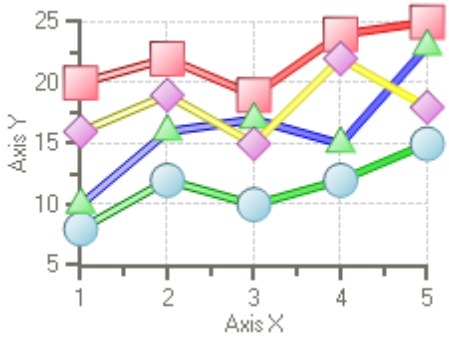
调整 **ScaleEffect** 属性来增加 XY 绘制中的符号的尺寸。

如果您需要增加数据序列中的符号的尺寸，那么需要完成以下步骤:

1. 打开视觉效果设计器。
   关于如何访问视觉效果设计器的更多细节，请参见访问视觉效果设计器(307 页)。
2. 在 **Available Elements** 下拉列表中，点击紧挨着 **Default** 的复选框。这样会为显示在 **Chart2D** 控件上的数据序列应用默认的视觉效果设置。
3. 选择 **Parameters** 选项卡，然后选择 ScaleEffect 属性，并把游标滑动到 0。6 或者

直接输入 0。6。

ScaleEffect 属性增加符号的尺寸



18.16 使用属性窗口创建和初始化图表元素

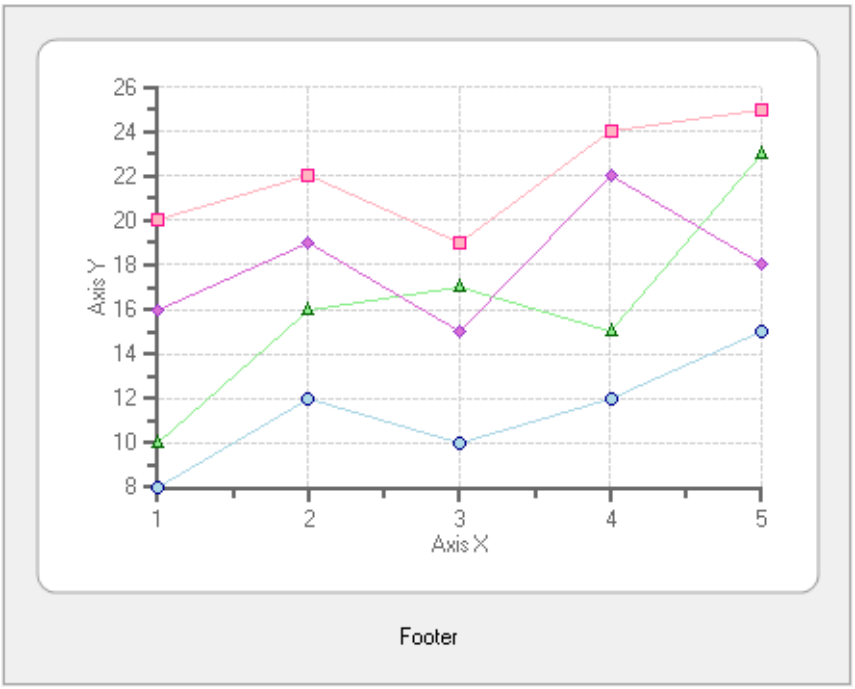
本节提供了若干使用属性窗口创建和初始化图表元素的任务。

18.16.1 通过属性窗口增加一个图表页脚

为了完成使用 **C1 图表**属性窗口创建一个图表页脚的目标，请按照以下步骤操作：

1. 鼠标右键单击 **C1 图表**控件，点击**右键菜单**的属性按钮，C1 图表的属性窗口将出现在画面右侧。
2. 展开**页脚节点**，并输入文字，例如，在文本属性后，输入“页脚"。
3. 将可见性属性设置为 **True。**

页脚元素在图表区域的下方显示，这是图表页脚元素的默认位置。

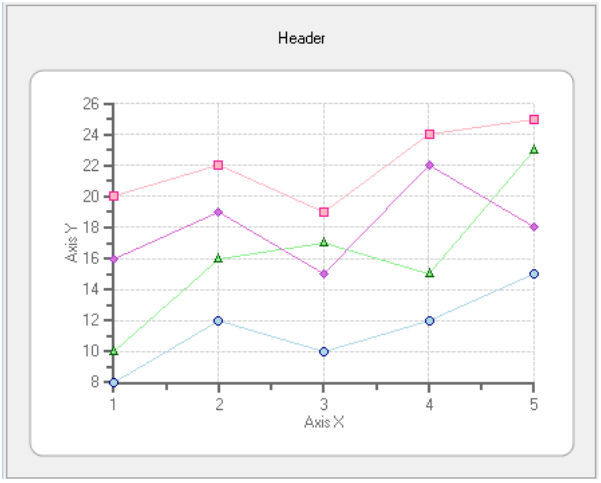


18.16.2 通过属性窗口增加一个图表标题

为了使用 **C1 图表**属性窗体增加一个图表标题的目标，请按照以下步骤操作：

1. 鼠标右键单击 **C1 图表**控件，点击**右键菜单**的属性按钮，C1 图表的属性窗口将出现在画面右侧。
2. 展开**标题节点**，并输入文字，例如，在文本属性后，输入"标题"。
3. 将可见性属性设置为 **True。**

标题元素在图表区域的上方显示，这是图表标题元素的默认位置。

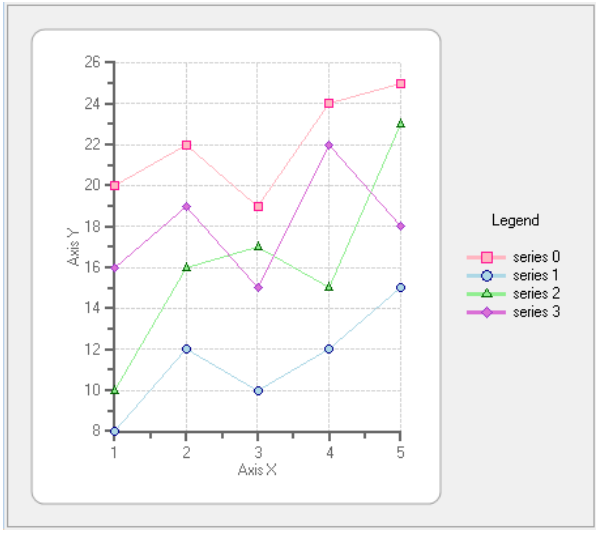


18.16.3 通过属性窗口增加图例说明

为了完成使用 **C1 图表**属性窗口增加图例说明的目标，请按照以下步骤操作：

1. 鼠标右键单击 **C1 图标控件**，点击**右键菜单**的属性按钮，C1 图表的属性窗口将出现在画面右侧。
2. 展开**图例说明**节点，并输入文字，例如，在文本属性后，输入 “图例说明"。
3. 将可见性属性设置为 **True。**

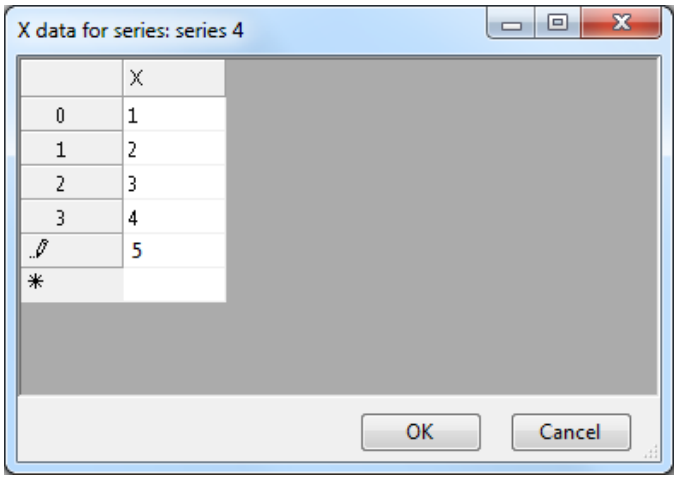
图例说明元素在图表区域的右侧或者东侧显示，这是图例说明的默认位置。



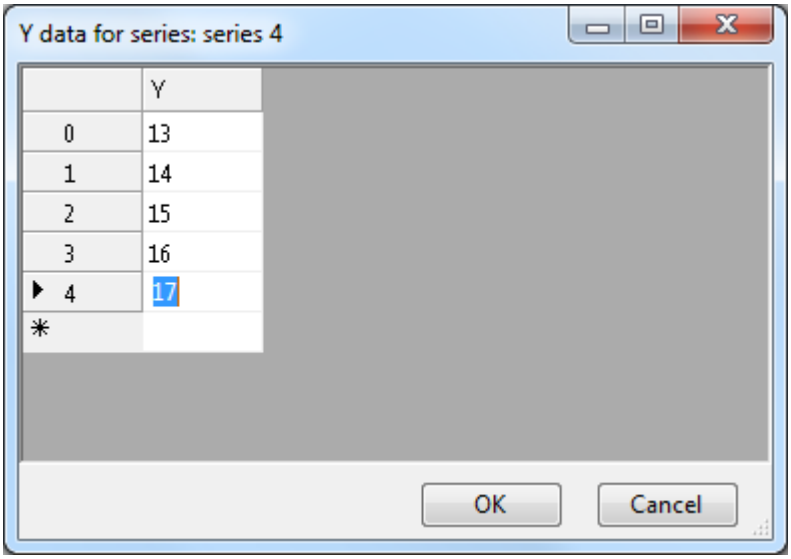
18.16.4 通过属性窗口将数据序列和数据增加到图表

为了完成使用 C1 图表属性窗口增加数据序列的目标，请按照以下步骤操作：

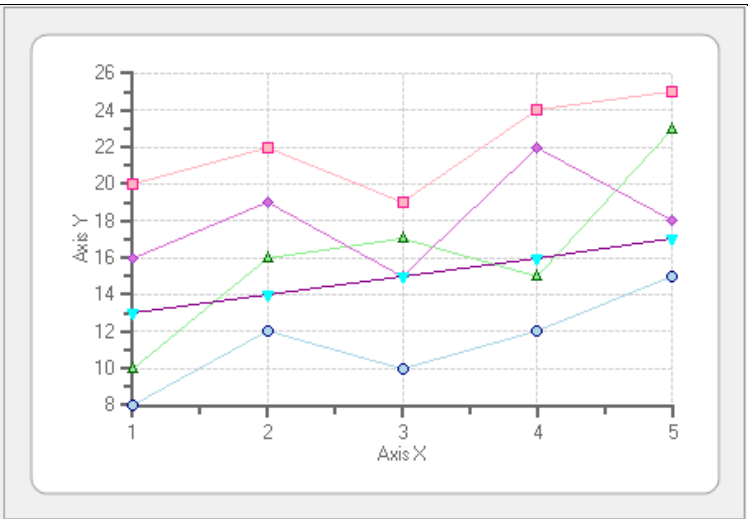
1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，**C1** 图表的属性窗口将出现在画面右侧。
2. 在 C1 图表属性窗口的杂项分组中，展开**图表分组**节点。
3. 展开 **Group0 -> ChartData**，然后点击**序列列表**属性旁边的**省略号**按钮。
4. 在图表序列集合编辑器上点击**增加按钮**，以便在 C1 图表控件上增加**一个新序列**。
5. 展开 **X 节点**，并且在对应的长度属性后输入 **5**。
6. 将数据类型设置为 **System.Single**。
7. 点击 X 属性后的**省略号**按钮，并输入 1、2、3、4、5，设置为 X 的值。



8. 展开 **Y 节点**，并且在对应的长度属性后输入 **5**。
9. 将其数据类型设置为 **System.Single**。
10. 点击 **Y 属性后**的省略号按钮，并输入 13、14、15、16、17，设置为 Y 的值，然后点击 **ok** 按钮。



新的 C1 图表序列出现在 C1 图表控件：

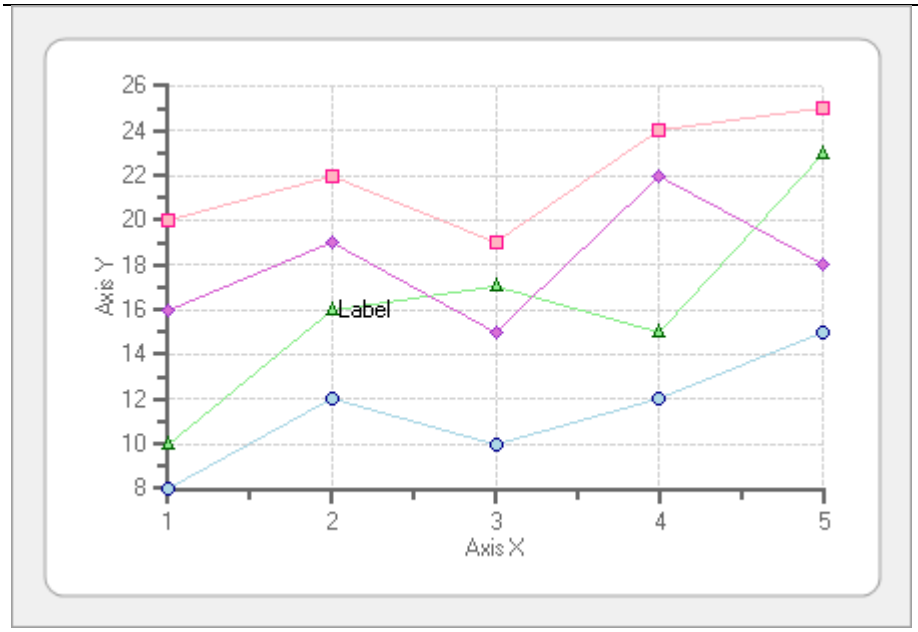


18.16.5 通过属性窗口为图表添加标签

为了完成使用 **C1Chart** 属性窗口为 C1 图表控件增加标签的目标，请完成以下步骤：

1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，C1 图表的属性窗口将出现在画面右侧。
2. 展开**图表标签**，并点击标签集合旁边的**省略号**按钮。
3. 在**标签集合编辑器**中，点击**增加按钮**来为 C1 图表控件增加一个新标签。
4. 在附加方法属性的下拉列表中选择**数据索引**。通过关联数据索引命令按钮得到附加方法枚举的**索引数据**，
5. 并在标签中设置**标签附加方法**属性的值，这将使标签在图表的图形区中和特殊的数据点关联起来。展开**附加方法数据**（AttachMethodData）节点. 注意点索引和**序列索引**属性都含有和它关联的值，这是因为数据索引附加方法通过数据点和标签进行关联。值为 **0** 的组索引代表图表分组 **0**，值为 **1** 的点索引代表序列中的**第二个数据**点，值为 **2** 的序列索引代表图表中的**第三个序列**。
6. 将**文本属性**设置为标签。
7. 点击 **ok** 按钮进行保存，并关闭**标签集合**编辑器。

标签出现在第三个数据序列的第二数据点上。



18.16.6 通过属性窗口为图表添加旋转标签

为了完成使用 **C1Chart** 属性窗口为 C1 图表控件添加标签的目标，请完成以下步骤：

1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，C1 图表的属性窗口将出现在画面右侧。
2. 展开**图表标签**，并点击标签集合旁边的**省略号**按钮。
3. 在标签集合编辑器中，点击**增加按钮**来为 C1 图表控件增加一个新标签。
4. 在附加方法属性的下拉列表中选择**数据索引**。通过关联数据索引命令按钮得到附加方法枚举的**索引数据**，
5. 并在标签中设置**标签附加方法**属性的值，这将使标签在图表的图形区中和特殊的数据点关联。展开**附加方法数据**节点。注意点**索引**和**序列索引**属性都含有和它相关数据。这是因为数据索引附加方法通过数据点和标签进行关联。值为 **0** 的组索引代表图表分组 **0**，值为 **1** 的点索引代表序列中的**第二个数据点**，值为 **2** 的序列索引代表图表中的**第三个序列**。
6. 将文本属性设置为**旋转标签**。
7. 将连接属性设置**为 True**。
8. 将偏移属性设置为 **10**。
9. 将旋转重写属性设置为 **60**。

当你通过代码来控制画面动画时，标签将会顺时针旋转 60 度角。

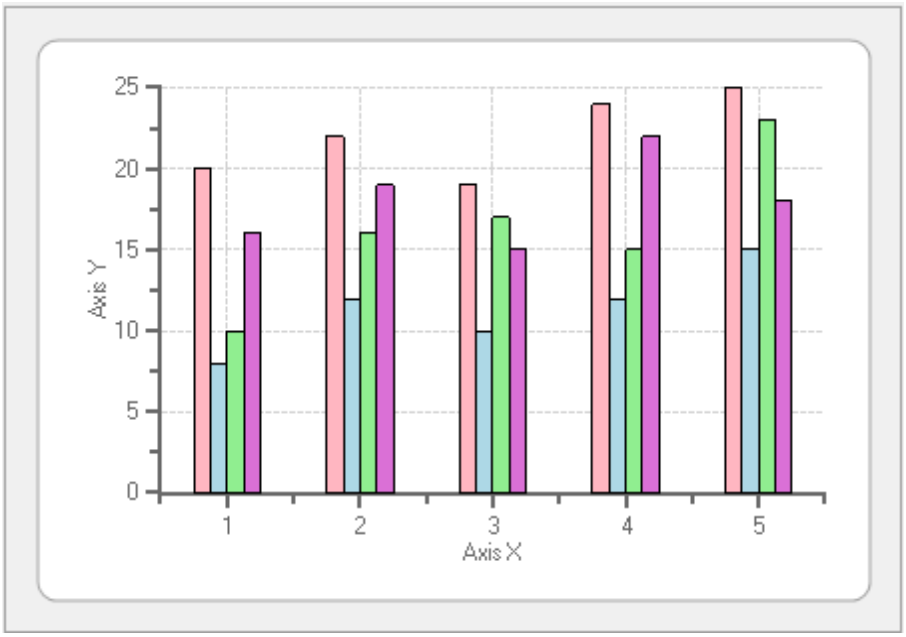
18.16.7 通过属性窗口选择图表类型

为了完成使用 **C1Chart** 属性窗口来选择图表类型的目标，请完成以下步骤：

1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，**C1** 图表的属性窗口将出现在画面右侧。

2. 展开**图表分组系列**节点，然后展开其中你想更改其图表类型的图表分组，比如组 1 或者组 **2**。
3. 点击**图表类型下拉箭头**，并从列表中选择类型为条的**图表类型**。

条图表类型显示在 C1 图表控件上。

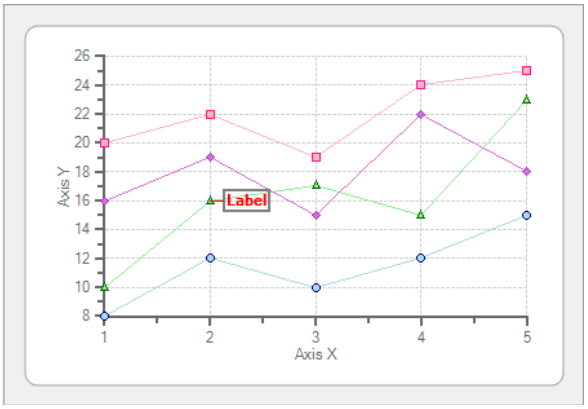


18.16.8 通过属性窗口修改图表标签的外观

为了完成通过 **C1Chart** 属性窗口编辑图表标签的目标，请完成以下步骤：

1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，**C1** 图表的属性窗口将出现在画面右侧。
2. 展开**图表标签**，并点击标签集合旁边的**省略号**按钮。
3. 选择**成员列表**中现存的标签，然后展开方式节点。
4. 展开边框节点，然后将边框样式设置为实线，颜色设置为**灰色**，将厚度设置为 **2**。
5. 将前景色设置为**红色**，并点击 **OK** 按钮。

标签的外观已经进行了修改：

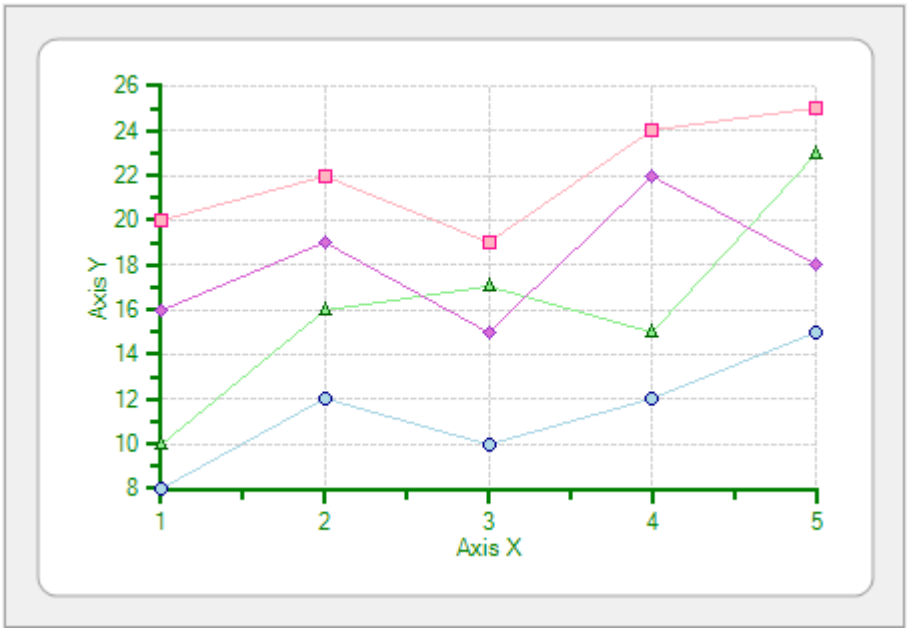


18.16.9 通过属性窗口修改 X 轴和 Y 轴的外观

为了完成通过 **C1Chart** 属性窗口修改 X 轴和 Y 轴外观的目标，请完成以下步骤：

1. 鼠标右键点击 **C1** 图标控件，点击**右键菜单**的属性按钮，**C1** 图表的属性窗口将出现在画面右侧。
2. 展开**图表区**节点，然后展开 **X 轴**节点。
3. 点击**前景色属性**旁的下拉箭头，并在自定义面板中选择绿色。
4. 开 **Y 轴**节点，并点击前景色属性旁的下拉箭头，并在**自定义面板**重选择**绿色**。

X 轴和 Y 轴的前景色已经被更改为绿色。



18.17 使用智能设计器来创建和格式化图表元素

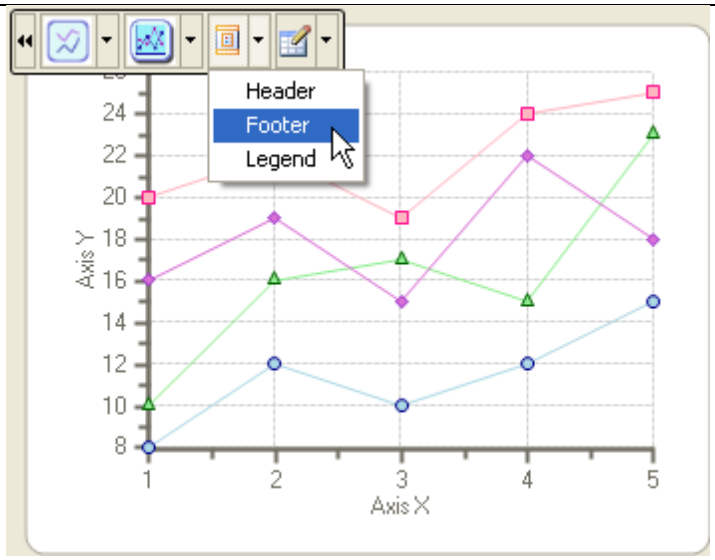
本章节提供了关于使用智能设计器来创建和格式化图表元素的任务.

本章节展示了如何使用智能设计器来创建和格式化图表元素,或者直接在窗体上修改既存的元素.例如,为了设置每一个图表元素的属性,您不用再在属性窗口中的图表对象上下拉动来进行设置,而可以简单地直接在窗体上修改图表元素.不写任何的代码就可以创建一个简单或者复杂类型的图表,这可以在设计时通过使用 C1Chart 编辑器来完成.

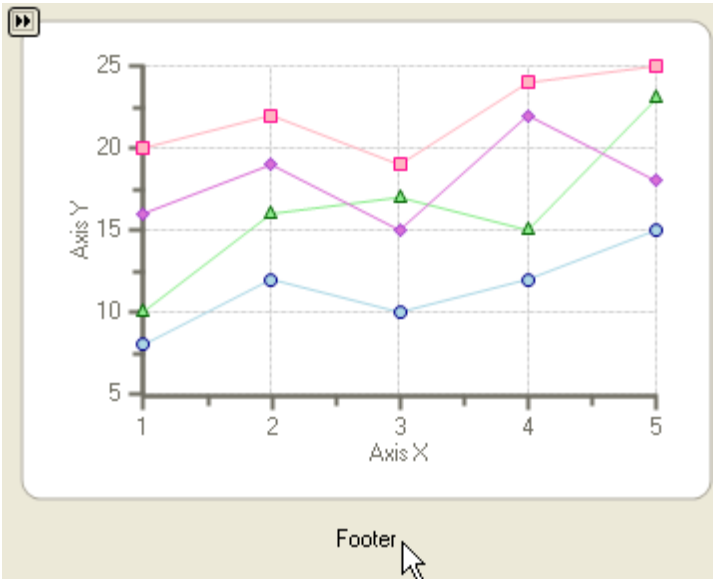
18.17.1 添加一个图表页尾

为了使用 C1Chart 浮动工具栏来添加一个图表页尾,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮  来打开 **C1Chart** 浮动工具栏,如果该工具栏还未打开的话.
2. 在 **C1Chart** 浮动工具栏上选择 **Layout** 下拉箭头,并且选中 **Footer** 条目.



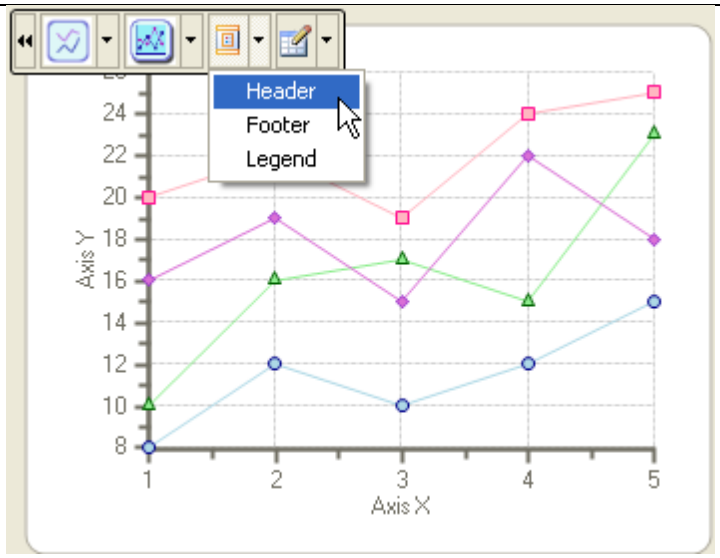
Footer 元素会出现在图表区域的下方,这里展示了图表的 Footer 元素的默认位置.



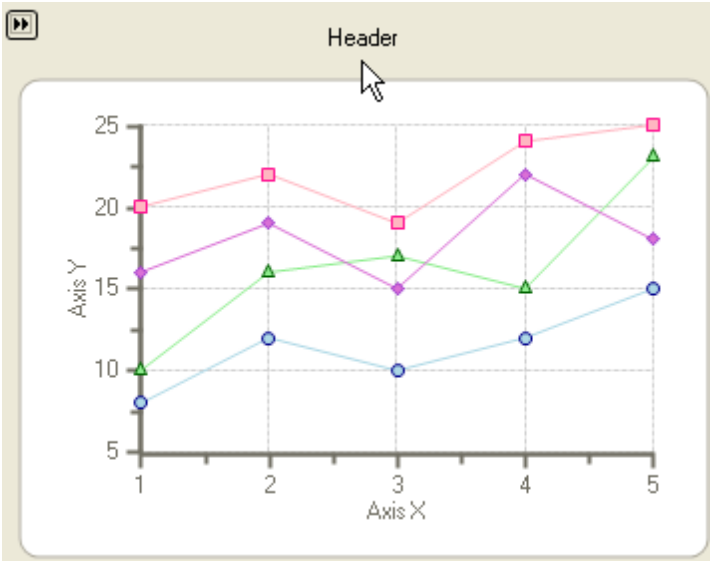
18.17.2 添加一个图表表头

为了使用 C1Chart 浮动工具栏来添加一个图表表头,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮 来打开 **C1Chart** 浮动工具栏,如果该工具栏还未打开的话.
2. 在 **C1Chart** 浮动工具栏上选择 Layout 下拉箭头,并且选中 Header 条目.



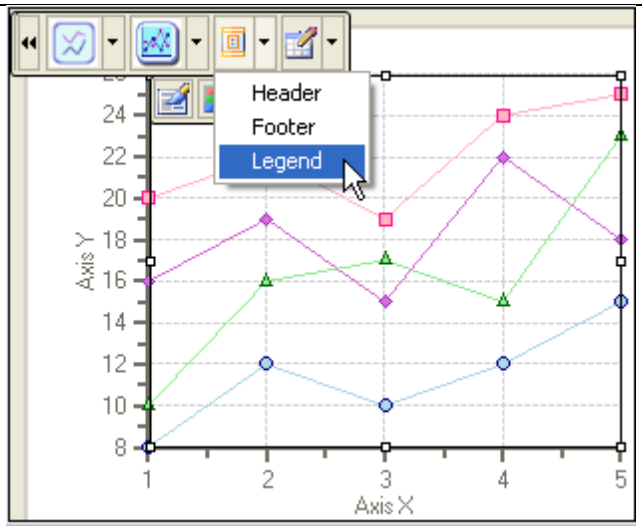
Header 元素会出现在图表区域的上方,这里展示了图表的 **Header** 元素的默认位置.



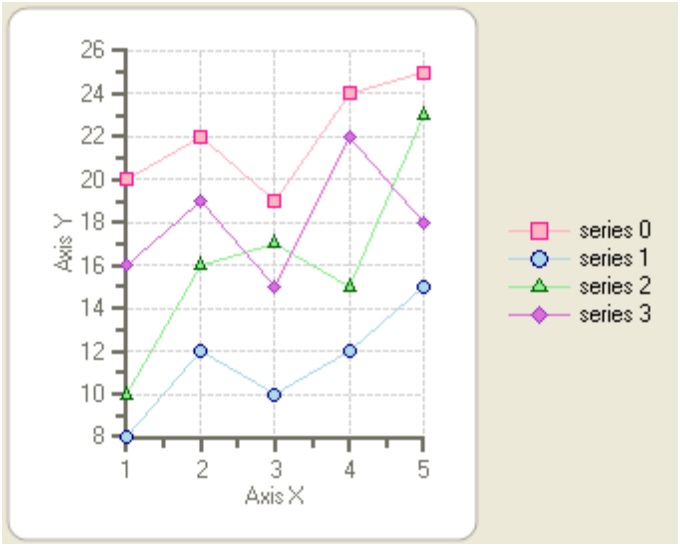
18.17.3 添加一个图表图例

为了使用 C1Chart 浮动工具栏来添加一个图表图例,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮 来打开 **C1Chart** 浮动工具栏,如果该工具栏还未打开的话.
2. 在 **C1Chart** 浮动工具栏上选择 Layout 下拉箭头,并且选中 Legend 条目.



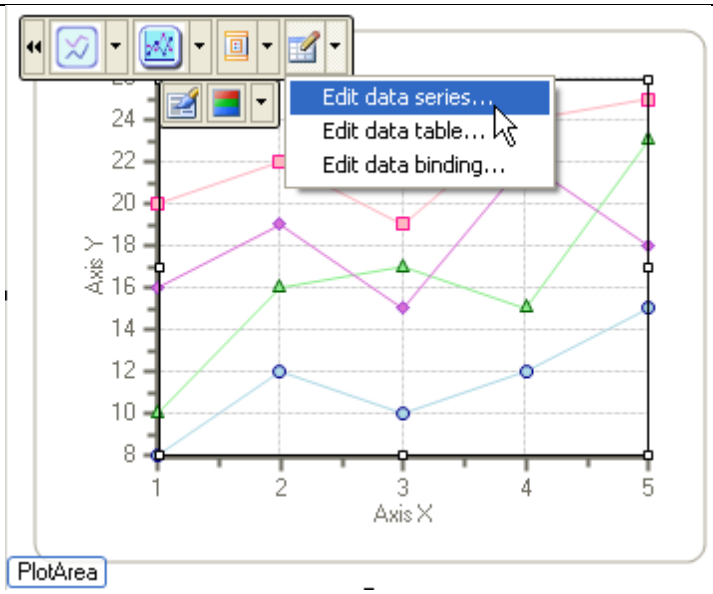
Legend 元素会出现在图表区域的右方或者东边,这里展示了图表的 **Legend** 元素的默认位置.



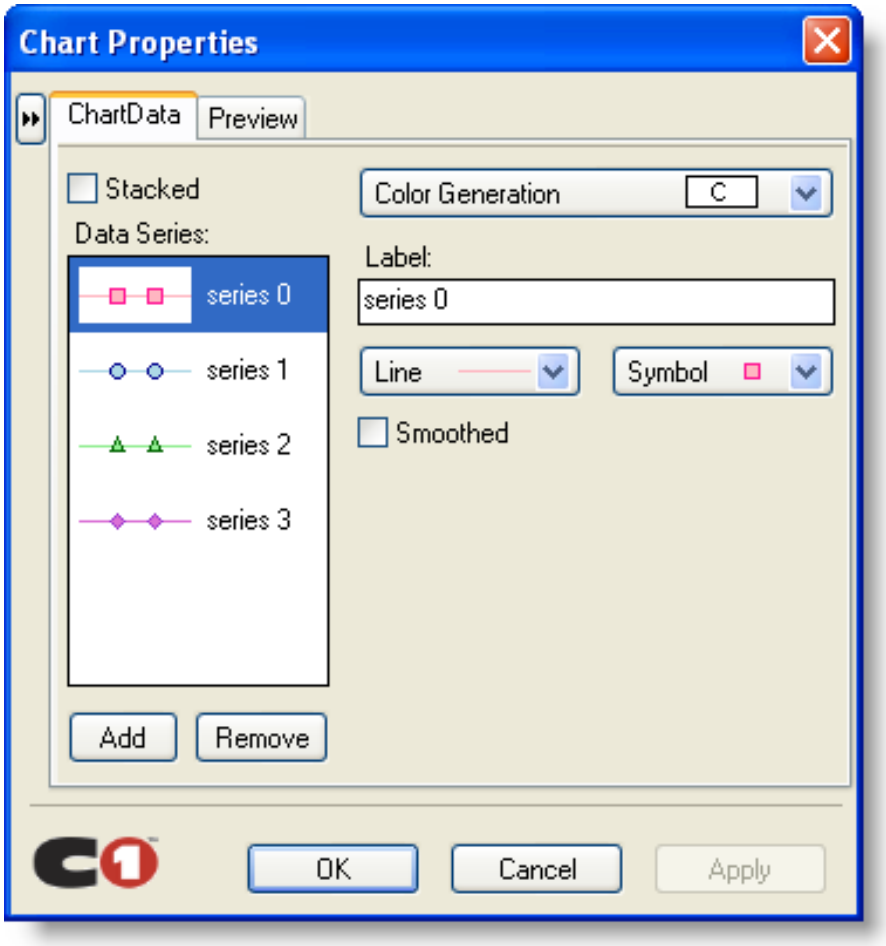
18.17.4 添加一个图表数据序列

为了使用 C1Chart 浮动工具栏来添加一个图表数据序列,请完成以下的步骤:

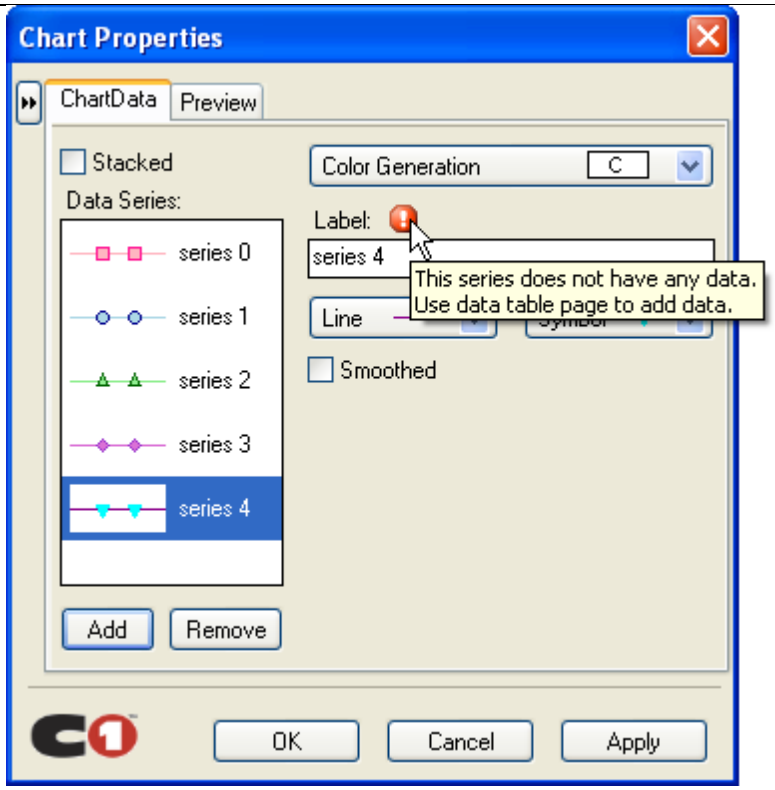
1. 选择 **C1Chart** 控件,并点击 **Open** 按钮  来打开 **C1Chart** 浮动工具栏,如果该工具栏还未打开的话.
2. 在 **C1Chart** 浮动工具栏上选择 Data 下拉箭头,并且选中 Edit data series 条目.



3. 图表属性设计器将会出现,来操作图表数据序列.点击 **Add** 按钮来给图表添加一个新的序列.



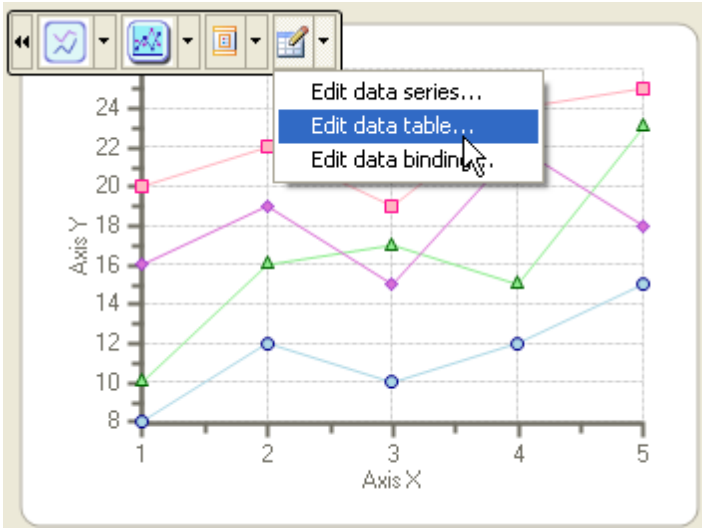
一个闪烁的感叹号将会出现,来提醒您数据需要被添加到新的序列中.



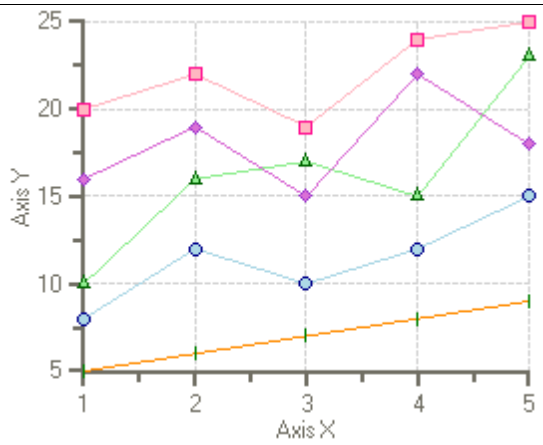
18.17.5 给数据序列中添加数据

为了使用 C1Chart 浮动工具栏来给数据序列中添加数据,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮 来打开 **C1Chart** 浮动工具栏,如果该工具栏还未打开的话.
2. 在 **C1Chart** 浮动工具栏上选择 Data 下拉箭头,并且选中 Edit data table 条目.



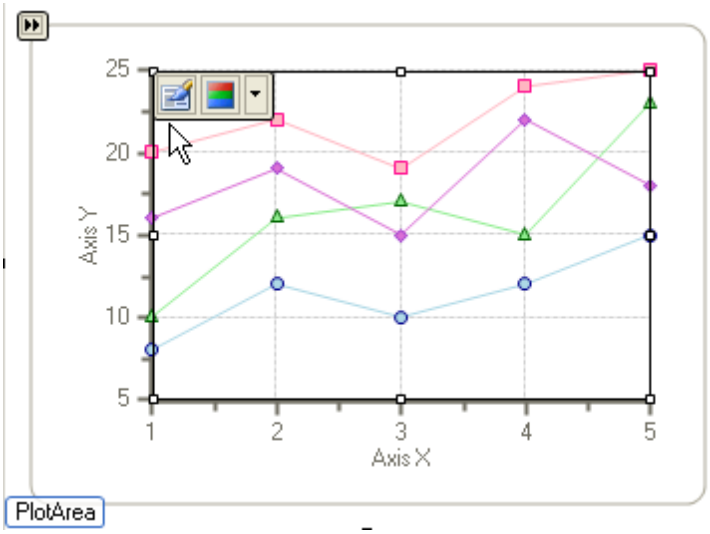
3. **Chart Properties Data Table** 编辑器将会出现,在表中定位到新序列上,并且为该序列任意输入一些 X 和 Y 值.
4. 当您完成新序列的数据的 X 和 Y 值的输入后,点击 **OK** 按钮,新图表序列将会出现在图表上.



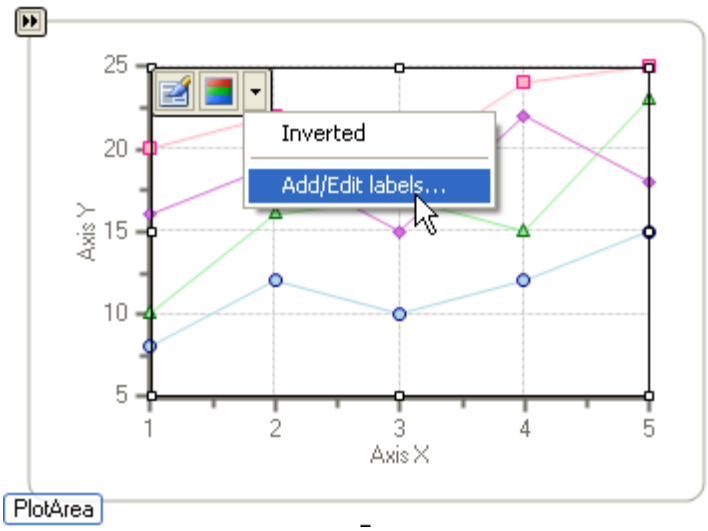
18.17.6 给图表添加标签

为了使用 Edit labels 编辑器来给数据序列添加图表标签,请完成以下步骤:

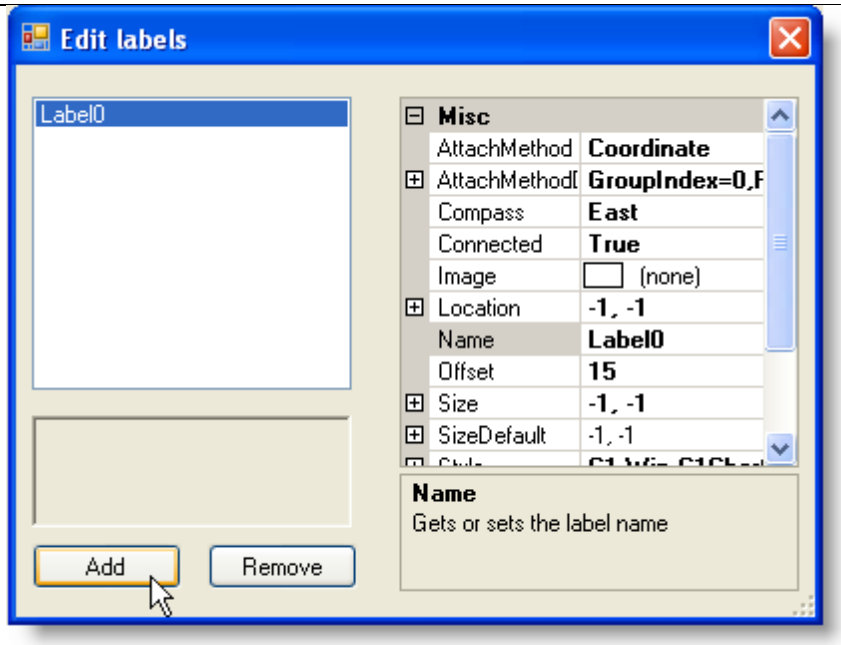
1. 滑动您的光标到绘制区域内,来展开 **PlotArea** 工具栏.



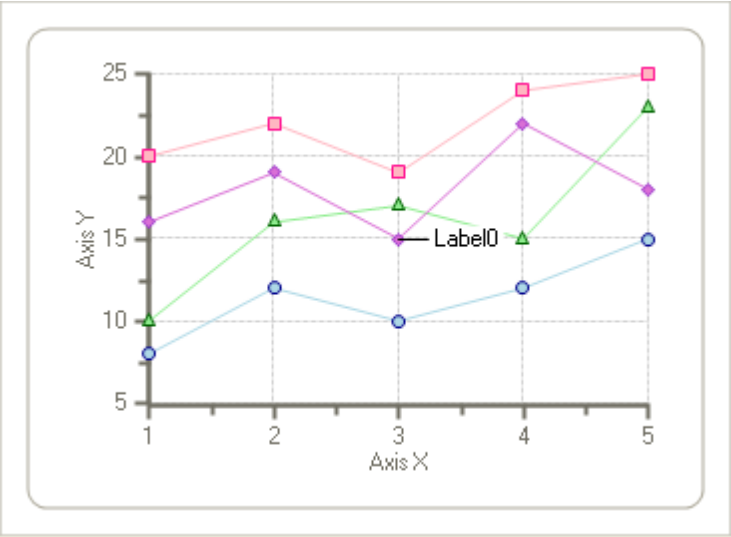
2. 选择 **Background** 下拉按钮并选中 **Add/Edit labels.**



3. 在 **Edit labels** 编辑器中选择 **Add** 按钮来添加一个新的标签.重复该步骤来添加更多的标签.如果您想删除一个标签,选择您希望删除的标签,然后点击 **Remove** 按钮.



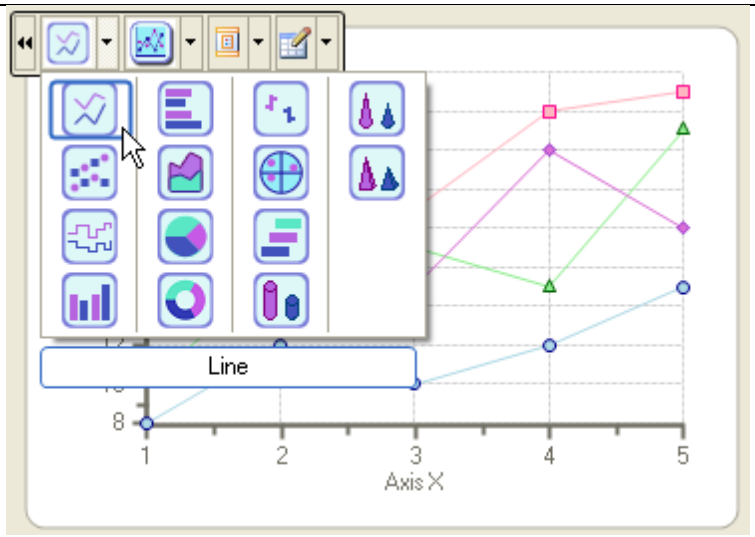
4. 点击 **Close** 按钮来关闭 Edit labels 编辑器.Label0 会出现在坐标系中.



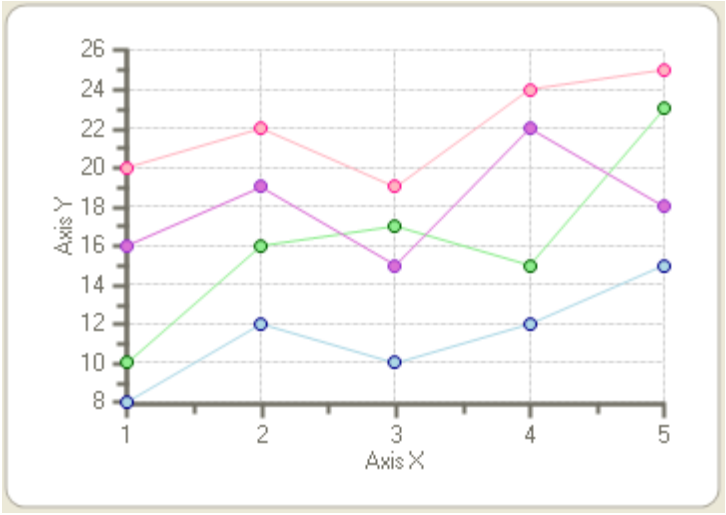
18.17.7 选择一个图表类型

为了使用 C1Chart 浮动工具栏来选择一个图表类型,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮 来打开 C1Chart 浮动工具栏,如果该工具栏还未打开的话.
2. 在 C1Chart 浮动工具栏上选择 **Chart Type** 下拉箭头,并且选择 Line 图表类型.



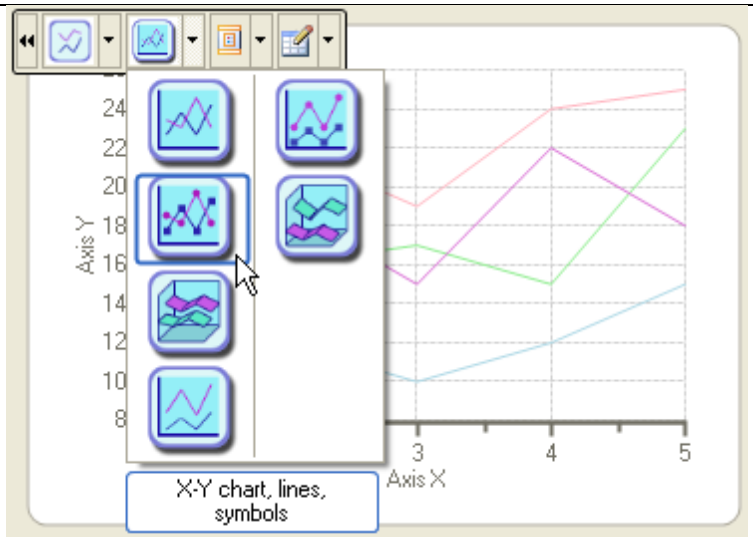
Line 图表类型会出现在 C1Chart 控件中.



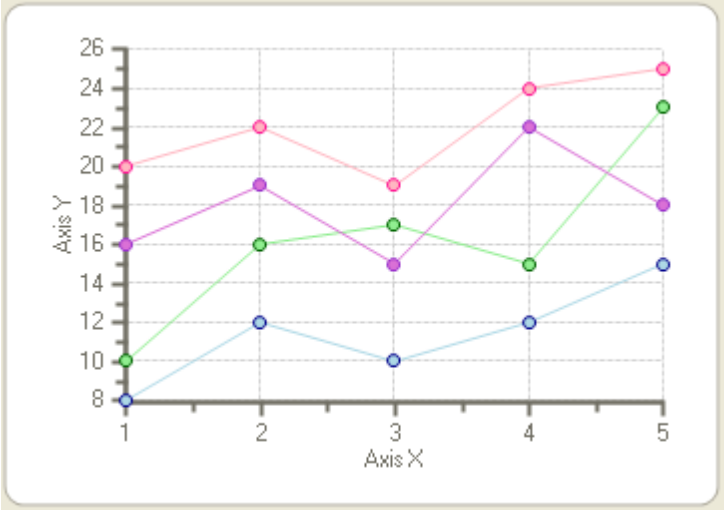
18.17.8 选择一个图表子类型

为了使用 C1Chart 浮动工具栏来选择一个图表子类型,请完成以下的步骤:

1. 选择 **C1Chart** 控件,并点击 **Open** 按钮 来打开 C1Chart 浮动工具栏,如果该工具栏还未打开的话.
2. 在C1Chart浮动工具栏上选择**Chart sub-type**下拉箭头,并且选择**X-Y chart, lines, symbols** 图表子类型.



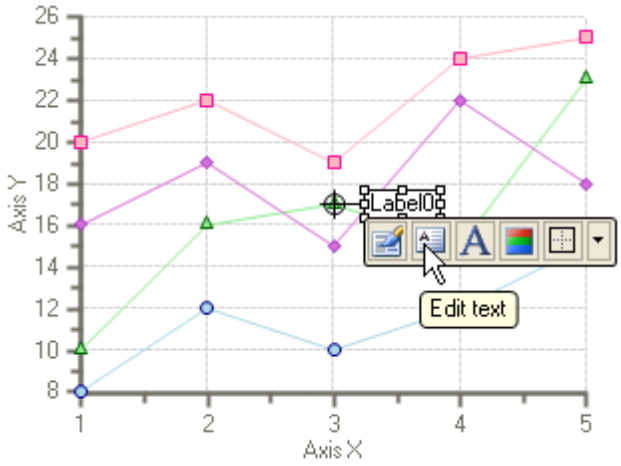
X-Y chart, lines, symbols 图表子类型将会出现在图表控件中.



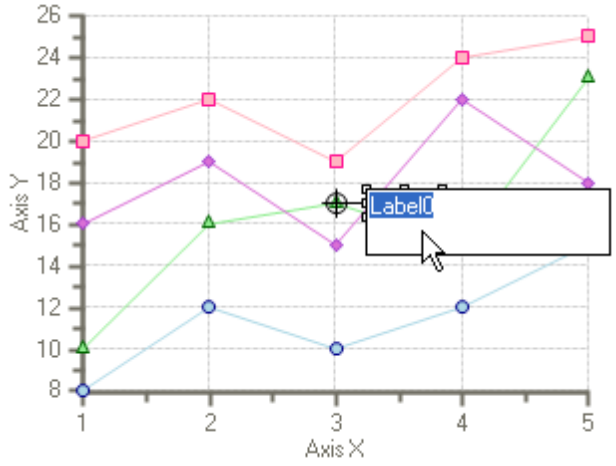
18.17.9 编辑图表标签

为了使用 Label 浮动工具栏来编辑图表标签,请完成以下的步骤:

1. 在图表上选择既存的标签,并且在 **Label** 浮动工具栏中选择 **Edit text** 按钮.



2. 标签的文本输入框变得可以编辑

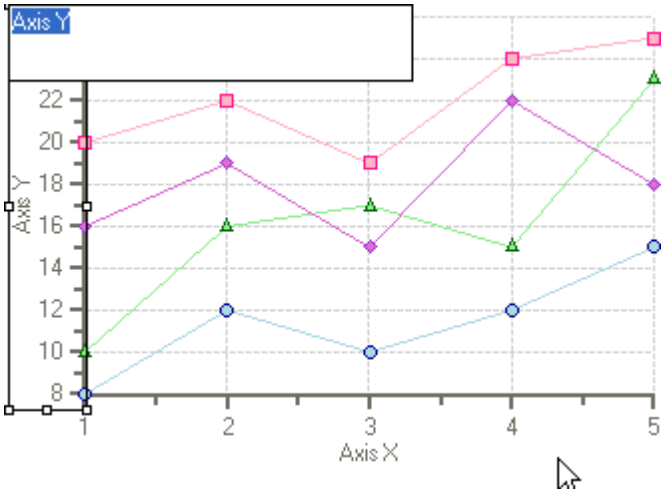


3. 将您的光标定位在 **lable0** 文字上,并且输入新的标签名称.

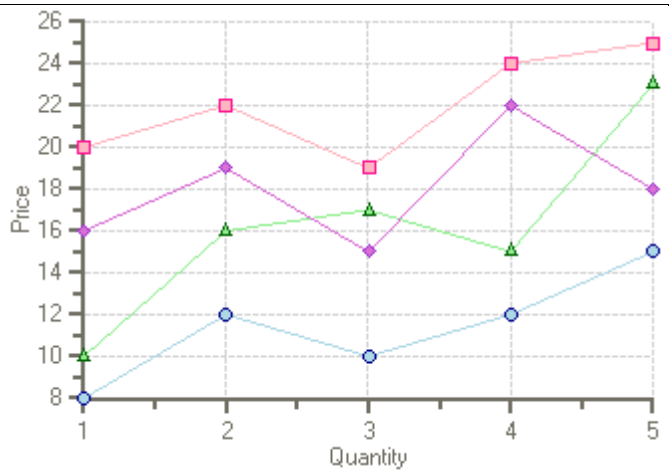
18.17.10 编辑 X 和 Y 轴线

为了在窗体上直接编辑 X 和 Y 轴线标签的名称,请完成以下的步骤:

1. 在 **C1Chart** 控件上选择 **Axis Y**.在 Axis Y 垂直框内点击,水平的 Axis Y 文本输入框将会出现.



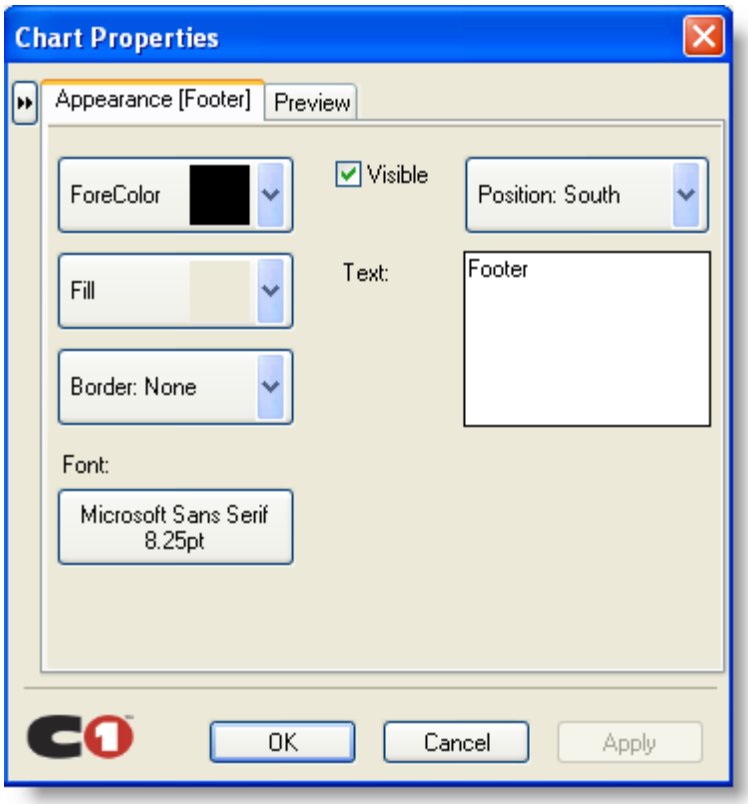
2. 在 **Axis Y** 文本输入框内输入 **Price** 并按下 Enter 键.
3. 在 C1Chart 控件上选择 **Axis X**,并且在水平框内点击,Axis X 的文本输入框将会出现.
4. 在 Axis X 文本输入框内输入 **Quantity** 并按下 Enter 键.Price 将会出现在 Y 轴线标签上,同时 Quantity 将会出现在 X 轴线标签上.



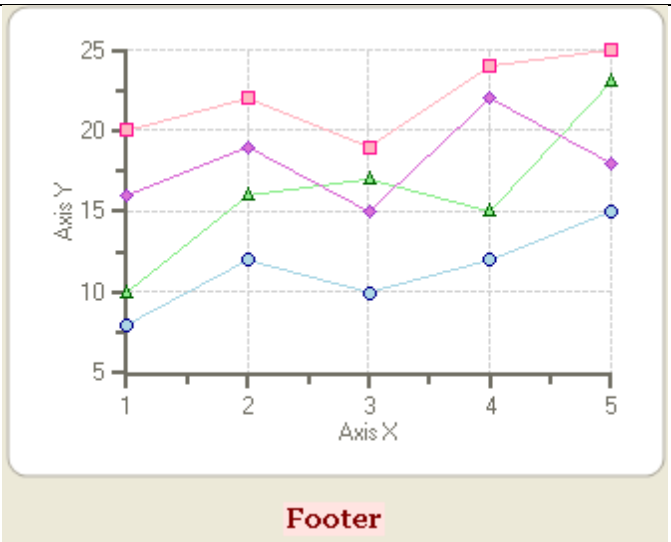
18.17.11 修改图表页尾的外观.

为了修改图表页尾的外观,请完成以下的步骤:

1. 在 Footer 浮动工具栏中选择 **Properties** 按钮,在图表属性编辑器中会出现 Appearance [Footer]选项卡.



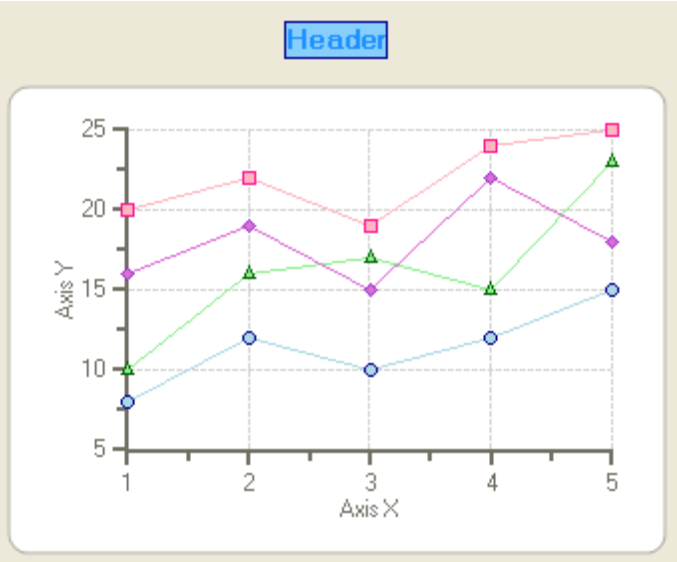
2. 按照以下的方式修改它的 **ForeColor, Fill**，和 **Border** 属性:
   - 设置它的 ForeColor 值为 **Maroon.**
   - 设置它的 Fill color's style 值为 Gradient. Color1 属性值为 **MistyRose ,Color2** 属性值为 **RosyBrown.**
   - 设置它的 Font type 属性值为 Georgia, Font style 属性值为 **Bold**，并且 Font size 属性值为 **10.**
3. 在对话框中选择 OK.在图表中会展示对 Footer 元素新的修改.



18.17.12 修改图表表头的外观.

为了使用 Header 浮点工具栏来修改图表表头的外观,请完成以下的步骤:

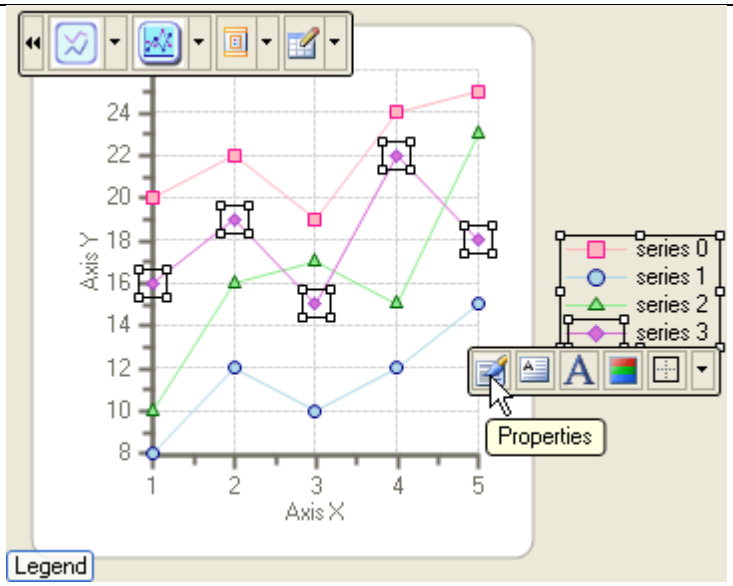
1. 在 **Header** 浮动工具栏中选择 **Chart Properties** 按钮,
2. 按照以下的方式修改它的 **ForeColor, Fill, 和 Border** 属性:
   - 设置它的 ForeColor 值为 **Dodger Blue**.
   - 设置它的 Fill color's style 值为 **LightSkyBlue.**
   - 设置它的 Border style 属性值为 **Solid,**并且 Border color 属性值为 **DarkBlue.**
   - 设置它的 Font style 属性值为 **Blod**,并且 Font Size 为 **10.**
3. 在对话框中选择 **OK.**在图表中会展示对 **Header** 元素新的修改.



18.17.13 修改图表图例的外观.

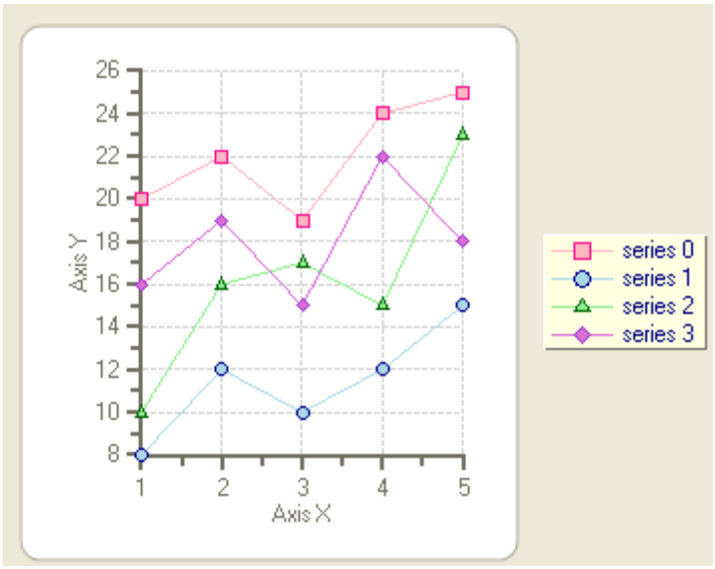
为了使用 **Legend** 浮点工具栏中的属性按钮来修改图表图例的外观,请完成以下的步骤:

1. 在 Legend 浮动工具栏中选择 Properties 按钮.



Appearance Legend properties 将会出现在 **Chart Properties** 编辑器上.

2. 按照以下的方式修改它的 **ForeColor, Fill,** 和 **Border** 属性:
   - 设置它的 **ForeColor** 值为 **Navy**.
   - 设置它的 **Fill color** 值为 **Info.**
   - 设置它的 **Border** 值为 **Raised**
3. 在对话框中选择 OK.在图表中会展示对 **Legend** 元素新的修改.



18.17.14 修改图表数据序列的外观

为了使用 C1Chart 浮动工具栏来修改图表数据序列的外观,请完成以下的步骤:

1. 选择 C1Chart 控件,并点击 **Open** 按钮 来打开 C1Chart 浮动工具栏,如果该工具栏还未打开的话.
2. 在 C1Chart 浮动工具栏上选择 **Data** 下拉箭头,并且选中 **Edit data series** 条目.

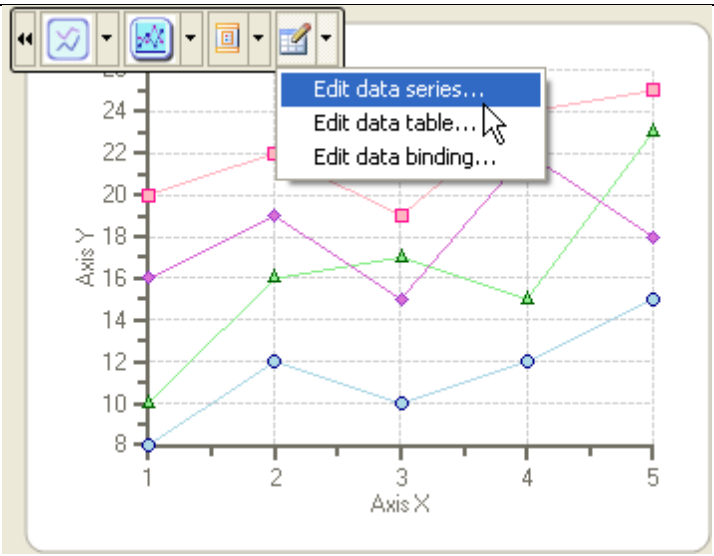
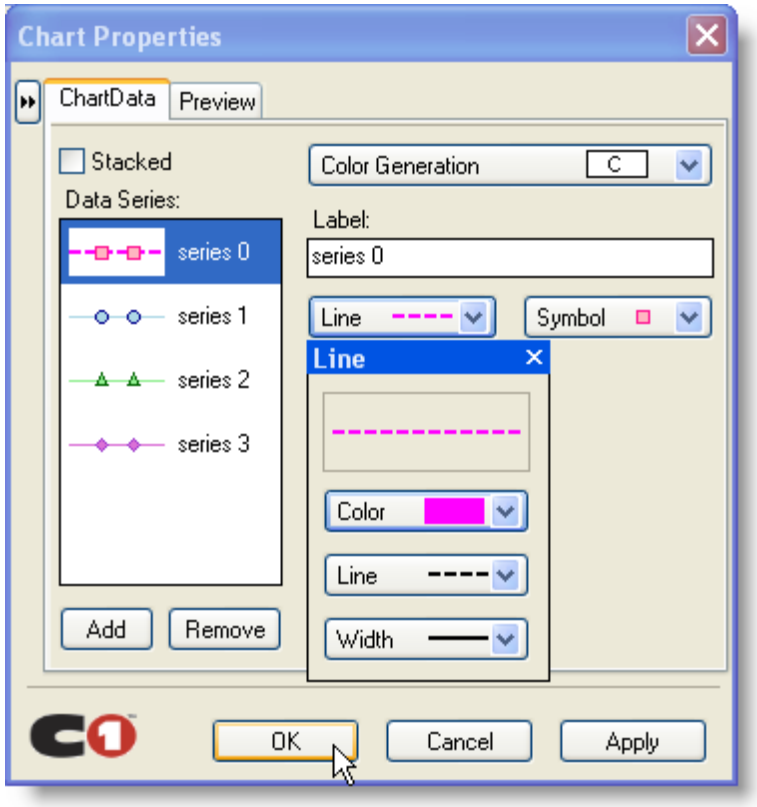
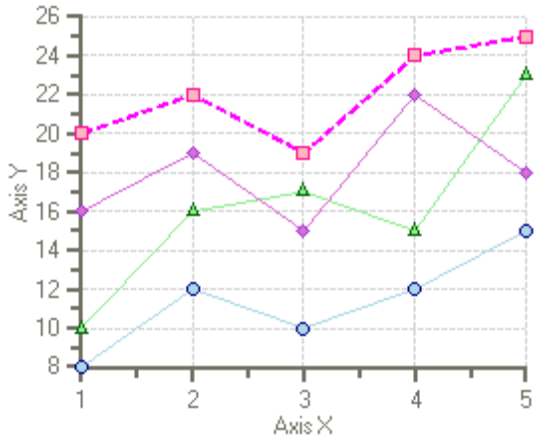


Chart Properties 编辑器将会出现.

3. 选中 S**eries0** 并且在 Line 属性的下拉箭头上点击,然后按照以下的方式来修改 **Line** 属性的 **Color,Line style** 和 **Width style** 子属性.
   - 从它的 **Color** 下拉列表框中选中 **Fuchsia**.
   - 从它的 **Line** 下拉列表框中选中 **dashed line.**
   - 从它的 **Width** 下拉列表框中选中 **level 2.**



4. 点击 **OK,**然后为 series0 做的修改将会被反映到图表上.



18.17.15 修改图表数据序列的颜色主题

为了使用 C1Chart 浮动工具栏来修改图表数据序列的颜色主题,请完成以下的步骤:

1. 选择 C1Chart 控件,并点击 **Open** 按钮  来打开 C1Chart 浮动工具栏,如果该工具栏还未打开的话.
2. 在 C1Chart 浮动工具栏上选择 **Data** 下拉箭头,并且选中 **Edit data series** 条目.

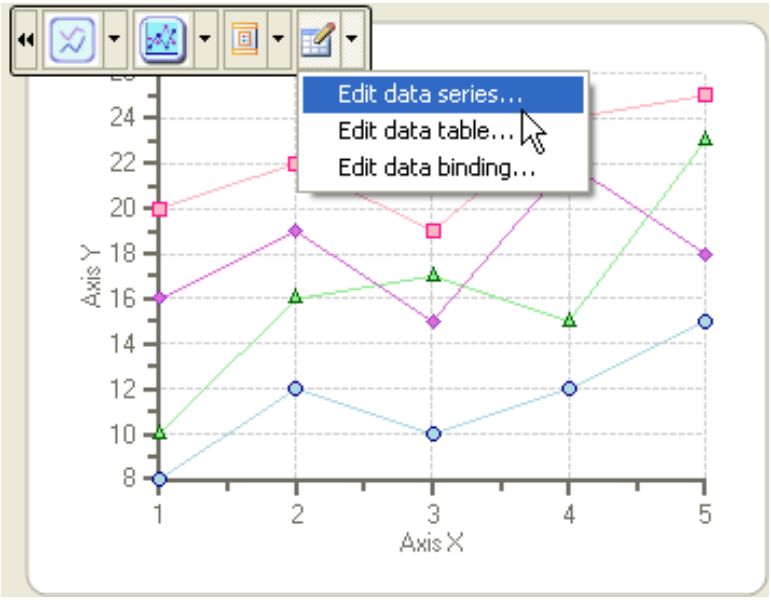
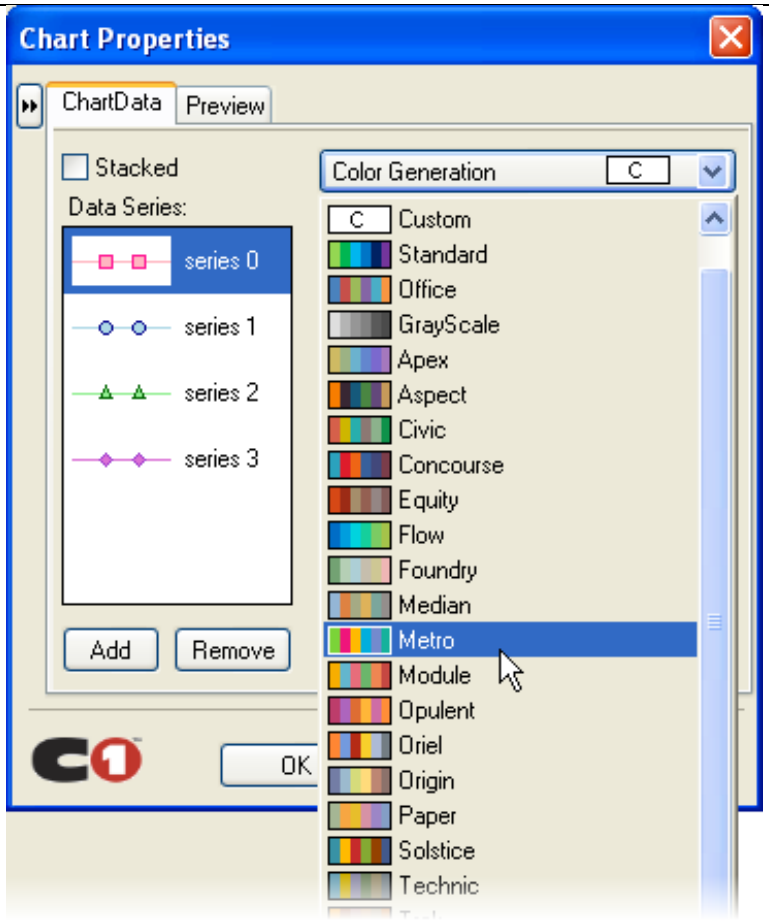


Chart Properties 编辑器将会出现.

3. 选中 **Series0**,并且点击 **Color Generation** 的下拉箭头,然后选中一个颜色主题,例如 Metro.



颜色主题会应用到所有的数据序列上.

4. 点击 **OK,**图表将会反映修改.

18.17.16 附加图表标签

Label 浮动工具栏包含三种附加图表标签的方法.以下的步骤展示了如何在设计时使用每一种方法来附加标签.如果您想知道如何使用代码的方式来进行标签的附加,请参加使用**像素坐标来添加图表标签**(200 页),使用**数据坐标来添加图表标签**(201 页),使用数**据点来添加图表标签**(201 页),或者使用数据点和 Y 值来添加图表标签(201 页).

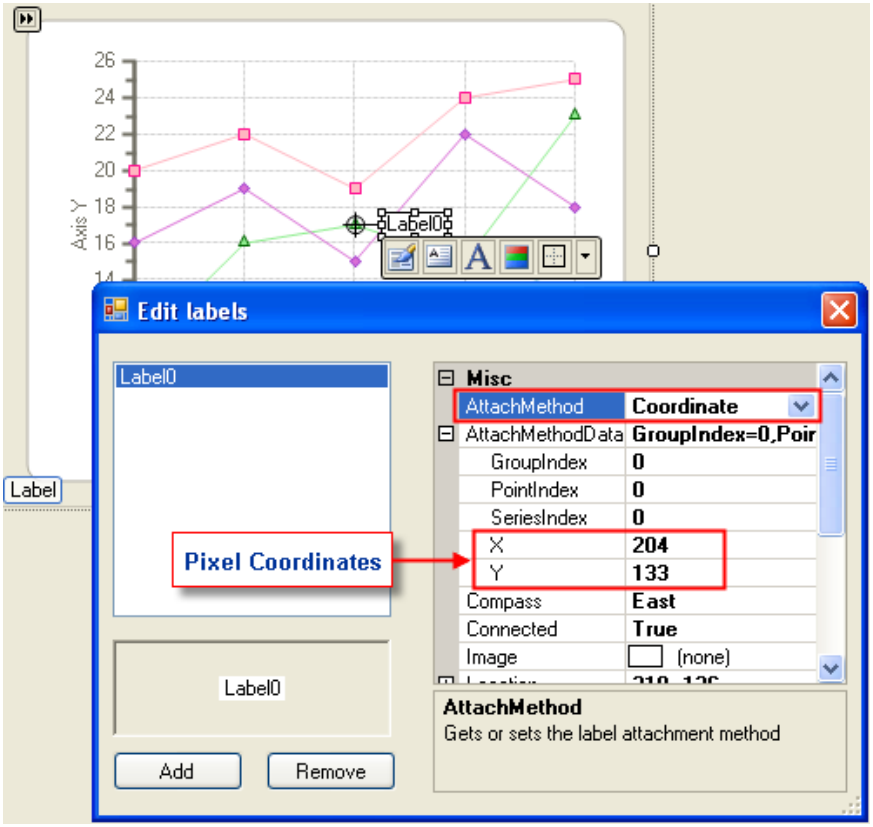
18.17.16.1 使用坐标附加

使用坐标来附加的方法通过获取 AttachMethodEnum 的 coordinate 成员,并且设置标签的 AttachMethod 属性值来完成.它可以在图表的任何地方附加标签.完成以下的步骤来添加一个标签:

1. 选中一个既存的图表标签,并且从它的下拉列表菜单中选中 Attached By Coordinate. 如果不存在一个标签那么就添加一个标签.关于在设计时添加一个图表标签的更多信息,请参见给图表添加一个标签(335 页).
2. 在 Label 浮动工具栏上点击 Properties 按钮,并且在它的 Edit lables 编辑器上观察以下的事实:
   - 因为 Attached by Coordinate item 被选中的原因, AttachMethod 属性现在被设置为 Coordinate.

- 定位X和Y坐标值,并且注意到他们现在被设置为像素值.这是因为坐标附加方法是用X和Y像素坐标值来附加标签.

下面的图展示了以上的两个观察内容.

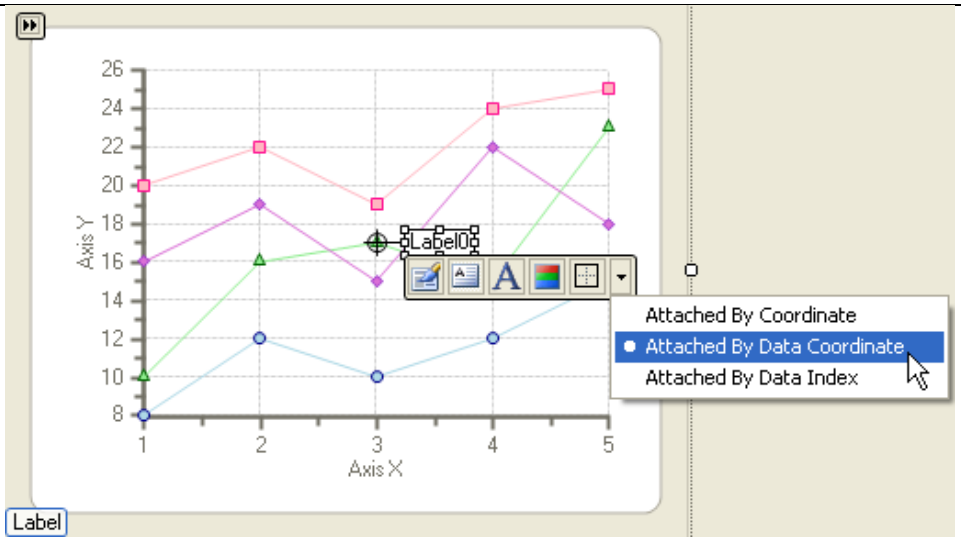


3. 如果您想为标签指定一个不同的位置,那么简单地改变X和Y的坐标像素值到您想要的图表上的位置即可.

18.17.16.2 使用数据坐标附加

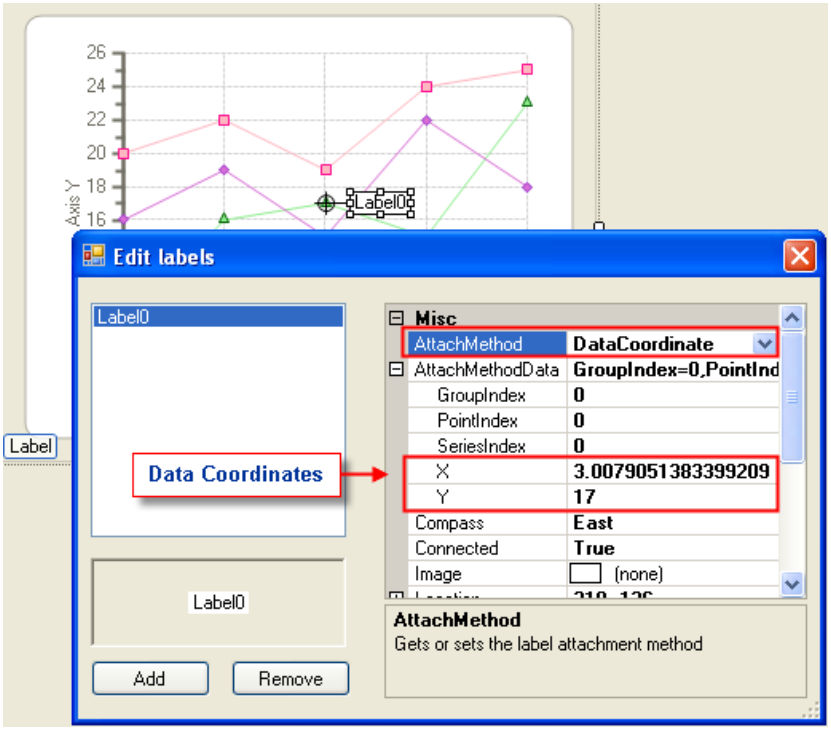
使用数据坐标来附加的方法通过获取 AttachMethodEnum 的 **DataCoordinate** 成员,并且设置标签的 AttachMethod 属性值来完成.它可以在图表绘制区域的任何地方或者在指定的坐标位置上附加标签.完成以下的步骤来添加一个标签:

1. 1,选中一个既存的图表标签,并且从它的下拉列表菜单中选中 **Attached By Data Coordinate.**

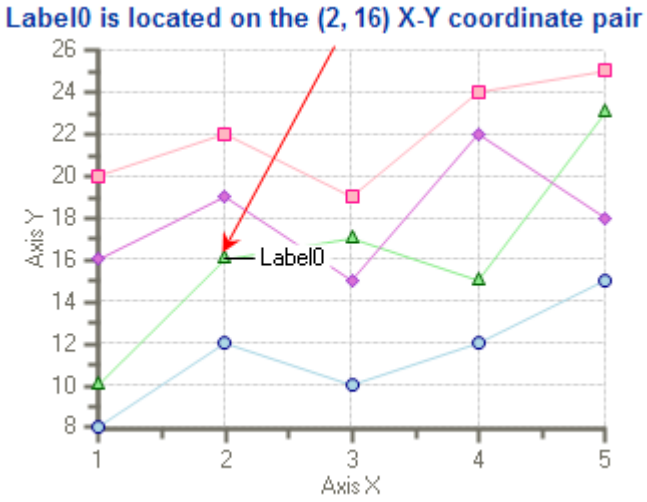


2. 在 Label 浮动工具栏上点击 Properties 按钮,并且在它的 Edit lables 编辑器上观察以下的事实:
   - 因为 Attached by data Coordinate item 被选中的原因, AttachMethod 属性现在被设置为 DataCoordinate .
   - 定位 X 和 Y 坐标值,并且注意到他们现在被设置为 X 和 Y 坐标值.这是因为数据坐标附加方法是用 X 和 Y 坐标值来附加标签.

下面的图展示了以上的两个观察内容.



3. 如果你想让标签代表一个不同的坐标对,那么简单地在 Edit labels 编辑器上改变 X 和 Y 的坐标值即可.例如,如果您想标签位于(2,16)坐标对上,那么在 Edit labels 编辑器上我们改变 X 值为 2,Y 值为 16.标签将会出现在(2,16)X-Y 坐标对上.下面的图展示了标签的新坐标.

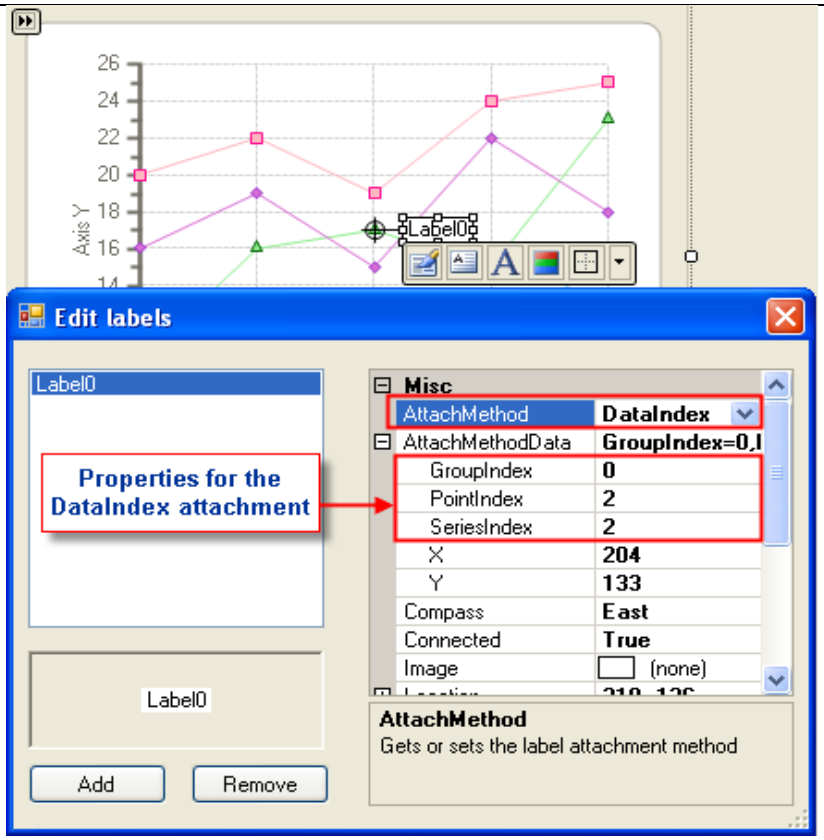


18.17.16.3 使用数据索引附加

使用数据索引来附加的方法通过获取 AttachMethodEnum 的 DataIndex 成员,并且设置标签的 AttachMethod 属性值来完成.它在图表的绘制区域上的一个特定数据点上附加标签.完成以下的步骤来添加一个标签:

1. 选中一个既存的图表标签,并且从它的下拉列表菜单中选中 **Attached By Data Index.**
2. 在 **Label** 浮动工具栏上点击 **Properties** 按钮,并且在它的 **Edit lables** 编辑器上观察以下的事实:
   - 因为 **Attached by Data Index** 被选中的原因, AttachMethod 属性现在被设置为 **DataIndex**.
   - 展开 AttachMethodData 对象.注意到 Point Index 和 Series Index 属性都含有一个值.这是因为数据索引附加方法通过数据点来附加标签. GroupIndex 值为 0 代表 ChartGroup0, PointIndex 值为 1 代表序列上的第二个数据点, SeriesIndex 值为 2 代表图表上的第三个序列.

下面的图展示了以上的两个观察内容.



3. 如果您想让标签代表一个不同的数据点,那么在 **Edit labels** 编辑器中改变 PointIndex 和 SeriesIndex 属性的值.例如,如果我们想在 series1,point3 上附加标签.那么在 **Edit labels** 编辑器上我们改变 SeriesIndex 的值为 1,并且 PointIndex 值为 3,标签会出现在 series1,point3 上.

18.18 烛台图任务

以下主题展示了在烛台图中的任务

18.18.1 增加烛体的宽度大小

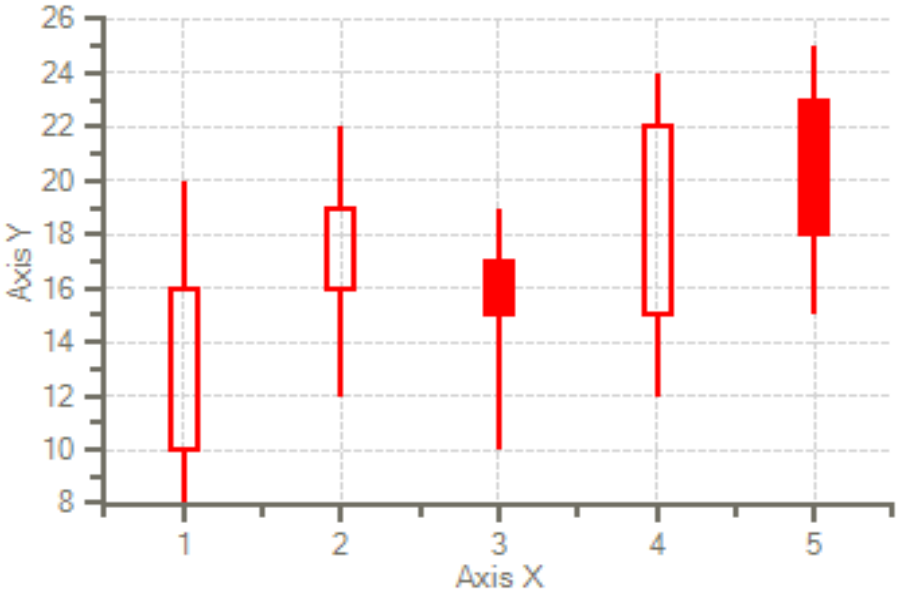
为了让图表烛体的宽度看起来更加的宽,为 SymbolStyle.Size 属性指定一个更大的值,像下述代码一样.注意,烛体的默认宽度大小为 5.

- Visual Basic

```
c1Chart1.ChartGroups(0).ChartData.SeriesList(0).SymbolStyle.Size = 10
```

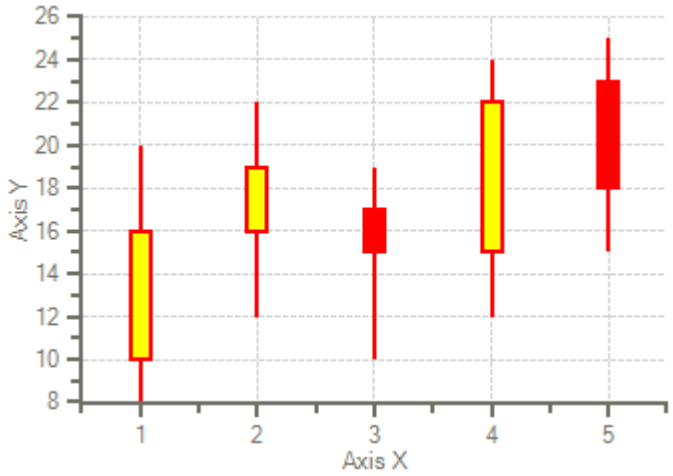
- C#

```
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].SymbolStyle.Size = 10;
```



18.18.2 为上升烛体创建一个填充色

为了给上升价格创建一个填充颜色,使用 FillTransparent 和 SymbolStyle 的颜色属性.



- Visual Basic

```
c1Chart1.ChartGroups(0).HiLoData.FillTransparent = False
c1Chart1.ChartGroups(0).ChartData.SeriesList[0].SymbolStyle.Color = Color.Yellow
```

- C#

```
c1Chart1.ChartGroups[0].HiLoData.FillTransparent = false;
c1Chart1.ChartGroups[0].ChartData.SeriesList[0].SymbolStyle.Color = Color.Yellow;
```

18.19 常见问题

本章节为常见的 2D 图表问题提供了解决方案.

18.19.1 如何改变图表类型?

为了改变图表类型,完成以下的步骤:

1. 在属性窗口中,展开 ChartGroups 节点.
2. 点击紧挨着 **ChartGroupsCollection** 的 **ellipsis** 按钮, **ChartGroups Collection Editor** 将会出现.
3. 从 ChartType 下列列表菜单中选择想要的图表类型.

注意:当使用 2D 图表时,以下的图表类型是可用的: XY-Plot, Pie, Area, Polar, Radar, Bubble, HiLo, HiLoOpenClose, 和 Candle.

关于图表类型的更多信息,**请参见特殊的 2D 图表**(80 页).

18.19.2 如何改变图表数据的绘制方式?

为了改变图表数据的绘制方式,完成以下的步骤:

1. 在属性窗口中,展开 **ChartArea** 节点.
2. 展开对应轴线的节点.
3. 在 **Grid Minor** 中,设置 **Max** 和 **Min**(展示在轴线上的最大和最小间隔),并设置 **UnitMajor**(轴线上的数值的间隔).

例如,如果 **Max** 被设置为 **20,Min** 被设置为 0,并且 **UnitMajor** 被设置为 5,在轴线上将会显示 5 的倍数的数字,0,5,10,15,20.

关于此的更多信息,请参见**轴线范围**(209 页).

18.19.3 如何改变展示在图表中的颜色?

改变序列颜色:

1. 在图表控件上右键点击.
2. 选择 **Chart Properties.**
3. 选择 **Data**.
4. 展开 **SymbolStyle** 节点.
5. 从 **Color** 下拉菜单中选择想要的颜色.

改变图表区域的颜色:

1. 在属性窗口中,展开 **ChartArea** 节点.
2. 展开 **Style** 节点.
3. 从 **BackColor** 下拉菜单中选择想要的颜色.

改变轴线的颜色:

1. 在属性窗口中,展开 **ChartArea** 节点.
2. 展开 **Style** 节点.
3. 展开 **Font** 节点.
4. 从 **ForeColor** 下拉菜单中选择想要的颜色.

关于此的更多信息,请参见**轴线外观**(205 页).

18.19.4 如何改变图例的定位?

为了改变图例的定位,完成以下的步骤:

1. 在属性窗口中,展开 **Legend** 节点.
2. 从 **Compass** 下拉菜单中选择 **East, West, North,** 或 **South.**

关于此的更多信息,请参见**图表图例**(239 页).

18.19.5 如何添加或者修改边框?

为了添加或者修改边框,完成以下的步骤:

1. 在属性窗口中,展开 **ChartArea** 节点.
2. 展开 **Style** 节点.
3. 展开 **Border** 节点.
4. 从 **BorderStyle** 下拉菜单中选择样式类型.
5. 从 **Color** 下拉菜单中选择想要的颜色.
6. 增大 **Thickness** 属性的值,来让边框看起来更加的突出,或者减小其值让边框看起来更加的不明显.

关于此的更多信息,请参见**图表边框**(242 页).

18.19.6 如何在多图表中同步 X 轴线?

在展示多图表的应用中,您可以同步每一个图表的 X 轴线.

使用 Margins 属性来对齐几个图表区域中的轴线位置.例如,下面的方法为两个图表布置了 X 轴线的位置,请注意可以修改它来同步三个图表.

- Visual Basic

```
Private Sub AdjustAxisPositions()
C1Chart1.ChartArea.Margins.Left = 10
C1Chart1.ChartArea.Margins.Right = 10
C1Chart2.ChartArea.Margins.Left = 10
C1Chart2.ChartArea.Margins.Right = 10
Dim img As Image = C1Chart1.GetImage()
If img IsNot Nothing Then
img.Dispose()
End If
img = C1Chart2.GetImage()
If img IsNot Nothing Then
img.Dispose()
End If
'
Dim ch1_X As Rectangle = C1Chart1.ChartArea.AxisX.GetAxisRect()
Dim ch2_X As Rectangle = C1Chart2.ChartArea.AxisX.GetAxisRect()
Dim d As Integer = ch1_X.Left - ch2_X.Left
If d > 0 Then
C1Chart2.ChartArea.Margins.Left += d
ElseIf d < 0 Then
c1Chart1.ChartArea.Margins.Left -= d
End If
d = ch1_X.Right - ch2_X.Right
If d > 0 Then
C1Chart1.ChartArea.Margins.Right += d
ElseIf d < 0 Then
C1Chart2.ChartArea.Margins.Right -= d
End If
End Sub
```

- C#

```
void AdjustAxisPositions()
{
c1Chart1.ChartArea.Margins.Left = 10;
c1Chart1.ChartArea.Margins.Right = 10;
c1Chart2.ChartArea.Margins.Left = 10;
c1Chart2.ChartArea.Margins.Right = 10;
Image img = c1Chart1.GetImage();
if( img!=null)
img.Dispose();
img = c1Chart2.GetImage();
if( img!=null)
img.Dispose();
//
Rectangle ch1_X = c1Chart1.ChartArea.AxisX.GetAxisRect();
Rectangle ch2_X = c1Chart2.ChartArea.AxisX.GetAxisRect();
int d = ch1_X.Left - ch2_X.Left;
if( d>0)
c1Chart2.ChartArea.Margins.Left += d;
else if( d<0)
c1Chart1.ChartArea.Margins.Left -= d;
d = ch1_X.Right - ch2_X.Right;
if( d>0)
c1Chart1.ChartArea.Margins.Right += d;
else if( d<0)
c1Chart2.ChartArea.Margins.Right -= d;
}
```