WinForms版本ComponentOne菜单和工具栏概述

可以使用WinForms版本的ComponentOne菜单和工具栏在您的WindowsForms应用程序中创建多种多样的菜单以及停靠/浮动的工具栏该套件包括九种用户界面和导航工具。ComponentOne SmartDesigner®这种独特的可视化开发功能允许您仅在设计界面上工作，完全不需要编码。内置最新的Microsoft Office 2010视觉样式，您可以仅通过设置一个属性达到Office 2010的外观效果。WinForms菜单和工具栏提供丰富的设计时支持，使您能够在短时间内创建功能齐全的菜单和工具栏。

Windows窗体的菜单和工具栏包含各种示例工程，你可以通过点击下面的链接详细了解每一个工程：

- Menus and Toolbars 概述
- DockingTab 概述
- NavBar 概述
- OutBar 概述
- TopicBar 概述
- RadialMenu 概述

ComponentOne Studio WinForms帮助

入门指南

关于安装WinForms版本的ComponentOneStudio，许可授权，技术支持，命名空间以及创建使用控件的工程等更多信息，请访问Studio for WinForms入门。

新功能

关于添加到ComponentOne Studio WinForms的最新功能的列表，请访问Studio for WinForms新功能。

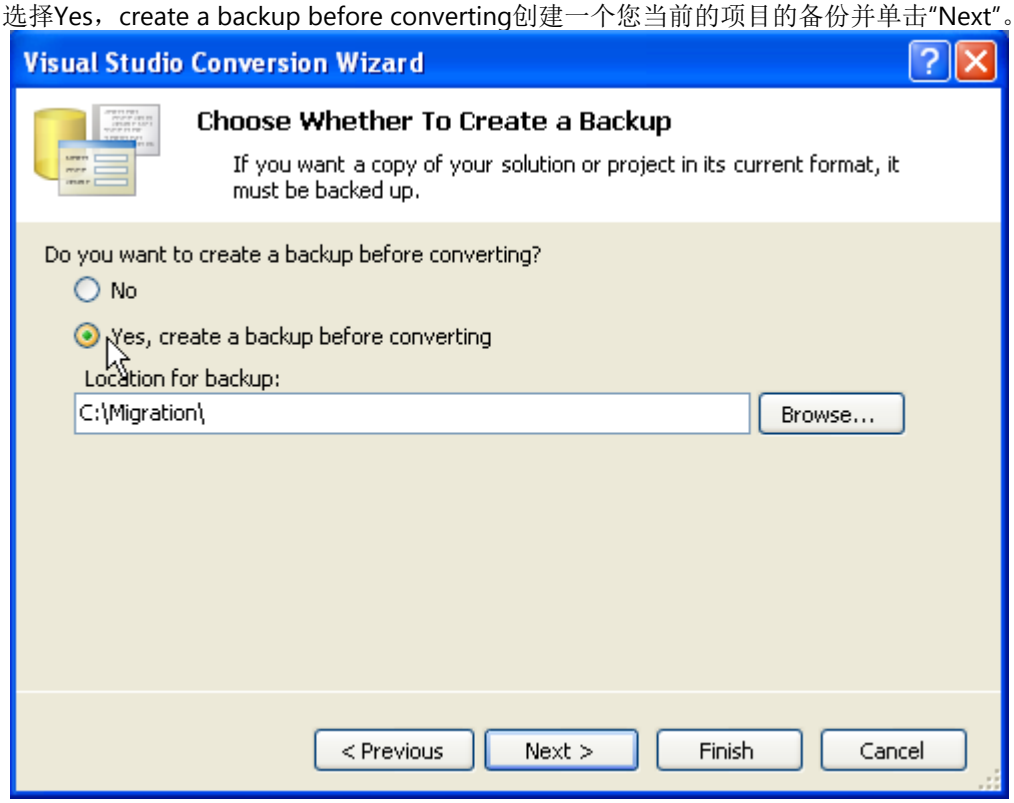
迁移C1Command工程到Visual Studio

迁移一个使用了ComponentOne的工程首先，你必须把你的工程转换为Visual Studio格式，包括删除对任何旧版本程序集的引用并添加对新版本程序集的引用。其次必须更新license文件（.licx文件），为了确保工程能够正确运行。

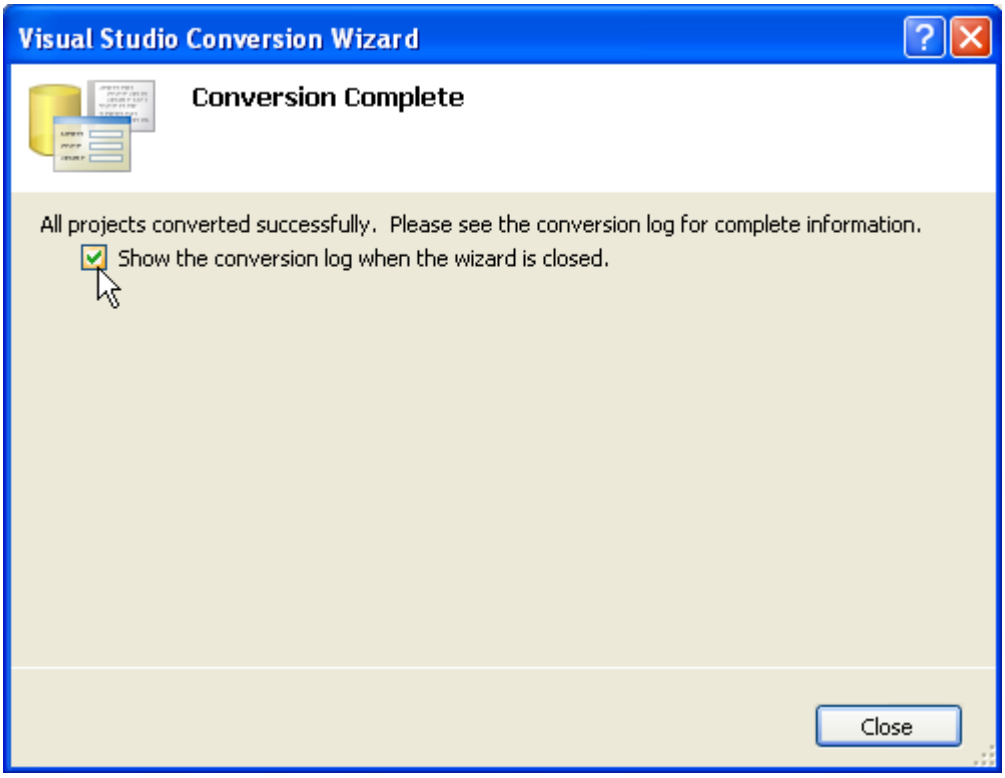
转换工程格式：

1. 打开Visual Studio并选择File，Open Project。
2. 找到您希望转换为Visual Studio格式的工程对应的.sln文件。选择它并单击Open。将出现Visual Studio的转换向导。

3. 单击“Next”。
4. 选择Yes，create a backup before converting创建一个您当前的项目的备份并单击“Next”。

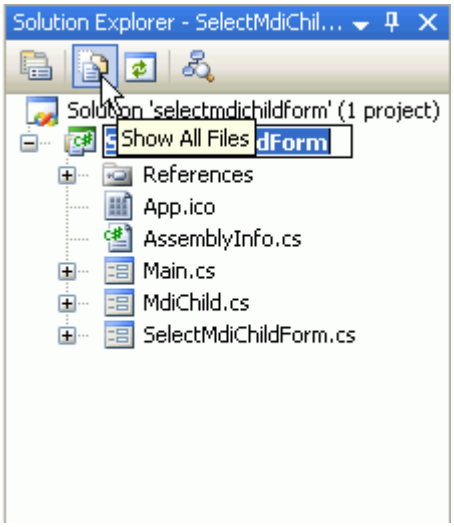


5. 单击“Finish”按钮，将您的项目转换为Visual Studio格式。转换完成窗口出现。
6. 如果您希望查看转换日志文件，请单击“Show the conversion log when the wizard is closed”。



7. 单击“Close”。该工程将打开。现在您必须删除对之前版本的ComponentOne .dll文件的引用并添加对新版本的引用。
8. 转到解决方案资源管理器（View，SolutionExplorer）并单击Show All Files按钮。

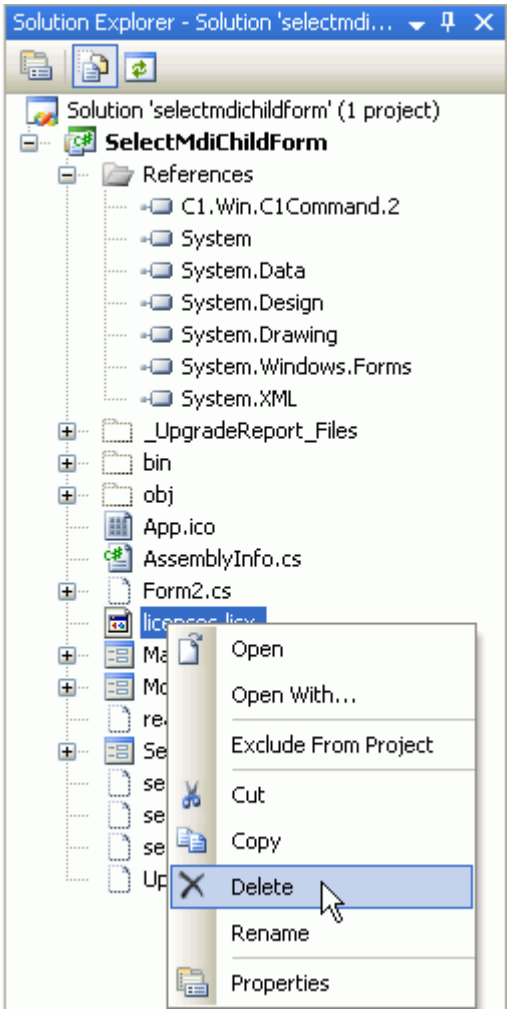
注意：如果解决方案的工程节点尚未被选中，则解决方案资源管理器的工具栏上不会出现“Show All Files”按钮。



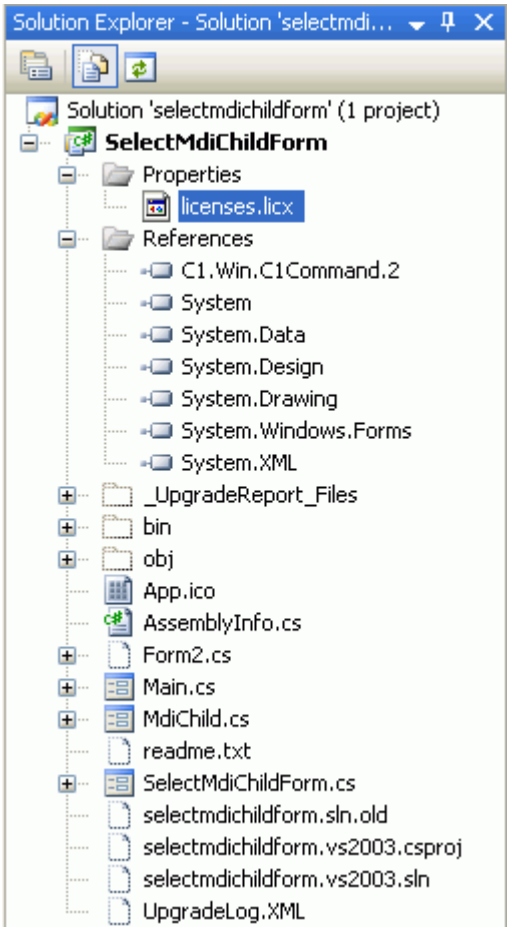
9. 展开“引用”节点，右键单击并选择删除C1.Common。同时按照相同的方式移除 C1.Win.C1Command。
10. 右击“引用”节点并选择Add Reference。
11. 找到并选择C1.Win.C1Command.4.dll。单击“确定”，将其添加到该工程。

更新.licx文件：

1. 在解决方案资源管理器中，右键单击licenses.licx文件并选择删除。



2. 单击"确定"永久删除licenses.licx工程必须重新创建一个新的.licx文件的更新版本。
3. 单击"调试"按钮开始编译并运行工程。新的.licx文件可能在解决方案资源管理器中看不到。
4. 选择File，Close关闭窗体，然后在解决方案资源管理器中双击Form.vb或Form.cs文件打开它。新的licenses.licx文件出现在文件列表。


Solution Explorer - Solution 'selectmdi...
Solution 'selectmdichildform' (1 project)
SelectMdiChildForm
Properties
licenses.licx
References
C1.Win.C1Command.2
System
System.Data
System.Design
System.Drawing
System.Windows.Forms
System.XML
_UpgradeReport_Files
bin
obj
App.ico
AssemblyInfo.cs
Form2.cs
Main.cs
MdiChild.cs
readme.txt
SelectMdiChildForm.cs
selectmdichildform.sln.old
selectmdichildform.vs2003.csproj
selectmdichildform.vs2003.sln
UpgradeLog.XML

迁移过程到此结束。

关键功能

WinForms版ComponentOne菜单和工具栏套件包含以下五个产品：Menus and Toolbars for WinForms，DockingTab for WinForms，NavBar for WinForms，OutBar for WinForms，以及TopicBar for WinForms.本段将列举每个产品的主要特点。

菜单和工具栏的主要特性

- **集成的C1Command框架**

  菜单和工具栏套件包中将菜单和工具栏集成到一个统一的系统，允许您在菜单项和工具栏项之间重用相同的对象和代码。相同的命令项（文本，图像，事件处理代码）可以同时被多个菜单和/或工具栏使用。

- **停靠/浮动工具栏**

  通过停靠/浮动行为，您可以创建一个最终用户易于定制的布局。为了创建一个最终用户可以在窗体周围移动停靠，或者浮动的工具栏，只需要简单地将C1ToolBar控件
  放置在C1CommandDock容器内。

- **免代码的设计体验**

  菜单和工具栏广泛地支持设计时功能，包括ComponentOne SmartDesigner®上下文敏感的浮动工具栏可以通过鼠标单击激活，在那里您做出的改变将立刻在设计器上生效。C1Command设计器甚至已经为您提供了超过五十个通用的命令，已经设置好这些命令的文本，图标以及快捷键。

- **内置的视觉风格**

  所有的菜单和工具栏控件支持视觉样式，模仿微软Office 2010包括蓝色，银色和黑色的风格。也可以从Office 2003，2007和Windows XP风格中间进行选择。

- **全局快捷键**

  您可以使用键盘通过快捷键和助记键访问菜单项，而不使用鼠标。快捷键也可应用于不出现在任何菜单或者工具栏上的命令。您也可以对快捷键文本进行本地化。

- **MDI应用程序的支持**

  对MDI（多文档界面）的应用程序有特别支持，包括内置的MDI子窗体列表以及分级的快捷键处理。在MDI应用程序中，您可以限制菜单项的列表显示的数量，在菜单列表中显示隐藏的MDI窗口，以及合并菜单项。

- **合并菜单**

  通过菜单和工具栏，您可以很容易地启用MDI子窗口和MDI父菜单和工具栏的合并功能。您也可以指定合并行为的类型，并选择是否添加，替换，删除或者合并菜单项，同时可以指定菜单项或工具栏按钮合并的顺序。

- **多级菜单**

  通过在主菜单添加子菜单创建一个分级的命令或选项列表。

- **内嵌控件**

  您可以很容易地在菜单和工具栏上嵌入任意控件，比如说文本框，单选按钮以及复选框等。

- **功能丰富的上下文菜单**

  通过使用C1ContextMenu控件，您可以容易地添加上下文菜单到任意控件。

DockingTab主要特性

- **停靠和浮动标签**
  DockingTab for WinForms提供停靠以及浮动行为，整个控件或者单独的页面（标签页）可以从窗体脱离并自动停靠到窗体的另一侧、其他的C1DockingTab 控件或者做为一个独立浮动的工具窗口。

- **仿VisualStudio停靠**
  C1DockingTab支持Visual Studio风格流行的停靠界面。只需要设置C1CommandDock控件的DockingStyle属性就可以实现这一行为，并对整个窗体的停靠布局提供支持。

- **自动隐藏**
  当放置在C1CommandDock控件内部时，C1DockingTab支持自动隐藏模式。这意味着标签页可以最小化到容器的任意边缘，当用户点击它们时，可以滑动打开/关闭。实现这一行为，仅需要设置AutoHiding属性的值为True。也可以通过设置CanAutoHide属性允许最终用户自行设置这一行为。

- **快速样式的设计时编辑器**
  通过这些易于上手的编辑器，您可以快速地编辑标签页的外观，比如说样式，尺寸以及布局。

- **标签行为**
  您可以通过行为相关的属性控制标签页的停靠，浮动，关闭以及重排。通过 CanCloseTabs 以及CanMoveTabs属性，给最终用户关闭以及移动标签页的能力。

- **标签列表**
  在下拉列表显示全部可用的标签，以便用户可以快速导航。仅需要设置ShowTabList属性的值为True。

- **更多的对齐选项，请选择不同的标签对齐选项：**
  - 沿着控件的顶部，底部，左侧或者右侧对齐标签
  - 相对于对齐方式，按照Near，Far，或者Center的设置放置标签
  - 可以拉伸或者压缩标签页以适应可用的空间，或者打开多行允许滚动以处理更多的标签。

- **隐藏标签**
  可以简单地通过隐藏C1DockingTab的标签以创建多页面窗体，比如说向导。使用C1DockingTab的好处是您可以对每一个标签页获取完全的设计时拖放支持，因此您可以容易地使得多个窗体包含在同一个窗体中。

- **更多的标签样式**
  向C1DockingTab的标签界面添加更多的样式。从全部的十二种视觉样式中进行选择， 包括全部的Office 2010，2007以及2003的设计。您还可以设置TabStyle属性获取倾斜或圆角标签。

NavBar主要特性

- **微软Outlook风格的用户界面**

分组的内容和导航菜单为不同的类别，就像使用微软Outlook导航系统。该模型可以帮助你组织内容并允许用户快速浏览。

- **内置的视觉样式**
  从8个内置的视觉样式中选择，包括Office 2007，2010，和2003的风格。这些都是通过一个属性即可设置完成。

- **可折叠**
  可以使得C1NavBar 折叠至其容器的任意一边，包括左侧，右侧，顶部以及底部。

- **内嵌控件**
  在每一个Nav内可以放置任意控件这可以在设计时通过简单的拖动控件至面板实现。

- **运行时定制**
  C1NavBar的按钮可以在运行时通过单击底部的下拉箭头按钮进行定制。用户可以重新整理按钮顺序并选择显示或隐藏某些按钮。

- **可堆叠的按钮**
  使用拆分条可以在导航栏底部的工具条上堆叠按钮。当按钮挤满时，它们将出现在选择下拉菜单。

- **关闭按钮**
  通过单个的属性可以允许用户关闭单个面板或者整个导航栏。

- **大量的设计时支持**
  ComponentOneNavBar具有广泛的设计时支持，包括ComponentOne SmartDesigner上下文敏感的浮动工具栏可以通过单击鼠标激活，在那里您所作出的改变将立刻反应在设计器中。这种免代码设计体验提供了对C1NavBar控件的轻松定制。

OutBar主要特性

- **手风琴式动画**

  展开和折叠的每一页都像一个传统的accordion 控件。只有一个页面可以在同一时间展开，从而节省屏幕空间。

- **内嵌任意控件**
  任意控制如文本框和图表可以嵌入在任何页面。

- **内置的视觉样式**
  十二种内置的视觉样式可供选择，包括所有的微软Office配色方案。

- **隐藏页面**

通过PageVisible属性，可以轻松地将任何页面变为不可见。

- **智能滚动**
  将C1OutBar配合C1ToolBar使用，提供对过长的工具栏命令项列表进行滚动。C1OutBar 可以在选中页面的上方或下方显示智能滚动按钮。

TopicBar主要特性

- **创建可折叠/可扩展的列表**
  通过C1TopicBar控件创建一个具有可折叠/收起的高度导航性窗体。这个强大的控件同时向您提供了使用动画的能力，比方说在折叠/展开页面时进行平滑过渡。

- **内置的视觉样式**

C1TopicBar 提供十二种内置的视觉样式，如Office 2010的配色方案以及Windows XP的外观。

- **免代码设计体验**
  TopicBar具有广泛的设计时支持，包括ComponentOneSmartDesigner®上下文敏感的浮动工具栏可以通过单击鼠标激活，在那里您所作出的改变将立刻反应在设计器中。

- **工具提示**
  主题栏有ToolTipText例如，您可以向每一个主题添加工具提示，向用户提供更多关于该分组的信息。

- **RTL支持**
  C1TopicBar为阿拉伯和希伯来语文化支持RTL（从右向左）的布局方式。仅需要设置RightToLeft属性为Yes。

RadialMenu主要特性

RadialMenu for WinForms 亮点包括:

- **强大的Command Framework:** ComponentOne RadialMenu WinForms借助C1Command的影响力和多功能提供最好的性能和灵活性。
- **完整的自定义功能**：利用Themes for WinForms改变RadialMenu的外观，也可以探究和自定义文本，添加图片等等。
- **良好的触摸**：RadialMenu for WinForms提供了可定制的接口能够轻松访问触屏或非触屏设备。

综述

MenusandToolbarsforWinForms组件套包允许您向应用程序添加外观漂亮的菜单以及可停靠/浮动的工具栏。该套件包具有诸多有用的功能，远超过表面看起来的那样。一些亮点包括：

- 微软的Visual Studio .NET/Office X以及Office 2007 / 2010的外观。
- 菜单和工具栏紧密融合；相同的命令项（文本，图像，事件处理代码）可以同时在多个菜单或工具栏使用。
- 集成在Visual Studio 窗体设计器的全面设计时支持。
- 停靠/浮动行为，布局最终用户可定制。
- 全局快捷键，对于没有出现在任何菜单或工具栏的命令同样有效。
- 特别支持MDI应用，包括内置的MDI子窗口的列表和分层的快捷键处理。
- 菜单/工具栏项的状态空闲时间自动更新。
- 上下文菜单，可以附加到窗体上的任意控件。
- 以及更多。

术语

在本文中，下列词语是用来指定特定的类或由.net 平台下ComponentOne菜单和工具栏产品提供的组件类分组：

- C1Command(两种含义)：作为一个简短的名字指定整个ComponentOne菜单和工具栏的.net 平台产品，以及用于指定在该产品的关键类。
- Command：用于指定C1Command类型的以及派生类型的对象，代表实际执行的命令，由菜单和工具栏按钮调用。
- Commandlink：用于指定C1CommandLink.类型的对象。
- Commandholder：用于指定C1CommandHolder.类型的对象。

类层次结构

本节总结了包括在C1Command套件包中比较重要的组件之间的类关系。

术语

C1Command包含以下用于创建菜单和工具栏的命令类型：

- C1Command:System.ComponentModel.Component
- 基本的可执行的命令。对于所有其他的命令类的基类。
- C1CommandMenu:C1Command
- 可以包含子菜单的C1Command命令类型。
- C1ContextMenu:C1CommandMenu
- 具有子菜单的命令，可以做为一个上下文菜单关联到另一个控件。
- C1CommandMdiList:C1CommandMenu
- 运行时扩展现有的MDI子窗体列表的命令。
- C1CommandControC1CommandMenu
- 可以关联到任意控件的命令。该控件显示在该命令链接之内（最多允许关联一个命令链接）。

其他组件和控件

- C1CommandHolder :System.ComponentModel.Component
  窗体上的所有命令的容器。每一个窗体仅允许添加一个对象，并且总是在向窗体添加一个菜单或工具栏是自动

创建。

- C1MainMenu:System.Windows.Forms.Control表示窗体
  主菜单的控件。
- C1ToolBar:System.Windows.Forms.Control
  表示工具栏的控件。必须放置在C1CommandDock之内才能够具有停靠以及浮动行为。
- C1CommandDock:System.Windows.Forms.Panel
  一个为工具栏提供停靠和浮动行为的容器。
- C1DockingTab:System.Windows.Forms.Control
  管理一组相关页面的Tab Control。
- C1OutBar:System.Windows.Forms.Contro
  一个outbar它提供了一个Outlook风格的标签的容器。
- C1NavBar:System.Windows.Form.Control
  用作将分组信息划分为不同的分类，帮助组织信息并进行快速信息导航。它由一些通过预设的按钮表示类别。
- C1TopicBar: System.Windows.Form.Control
  C1TopicBar表示一个topic bar。C1OutBar 控件提供了一组单个页面的集合，组织成一个分组，而C1TopicBar 包含一组划分为不同分组的页面。

设计时支持

设计时支持

C1Command提供可视化编辑，使得其可以容易地创建具有Microsoft Visual Studio .NET外观的菜单、工具栏、outbar、以及可停靠标签。

您可以用以下一个或多个可视化编辑控件对C1Command做出修改：

就地编辑

您可以通过就地编辑功能快速编辑菜单和工具栏项中的文本。关于使用就地编辑功能的更多信息，请参见就地文本编辑。

调用智能标签

你可以通过使用其智能标签很容易地设置C1Command的常见属性。关于C1Command智能标签的更多信息，请参见C1Command智能标签。

调用上下文菜单

通过使用关联的上下文菜单，您可以容易的在设计时定制任何C1Command组件。关于C1Command上下文菜单的更多信息，请参见C1Command 上下文菜单。

调用C1Command 编辑器

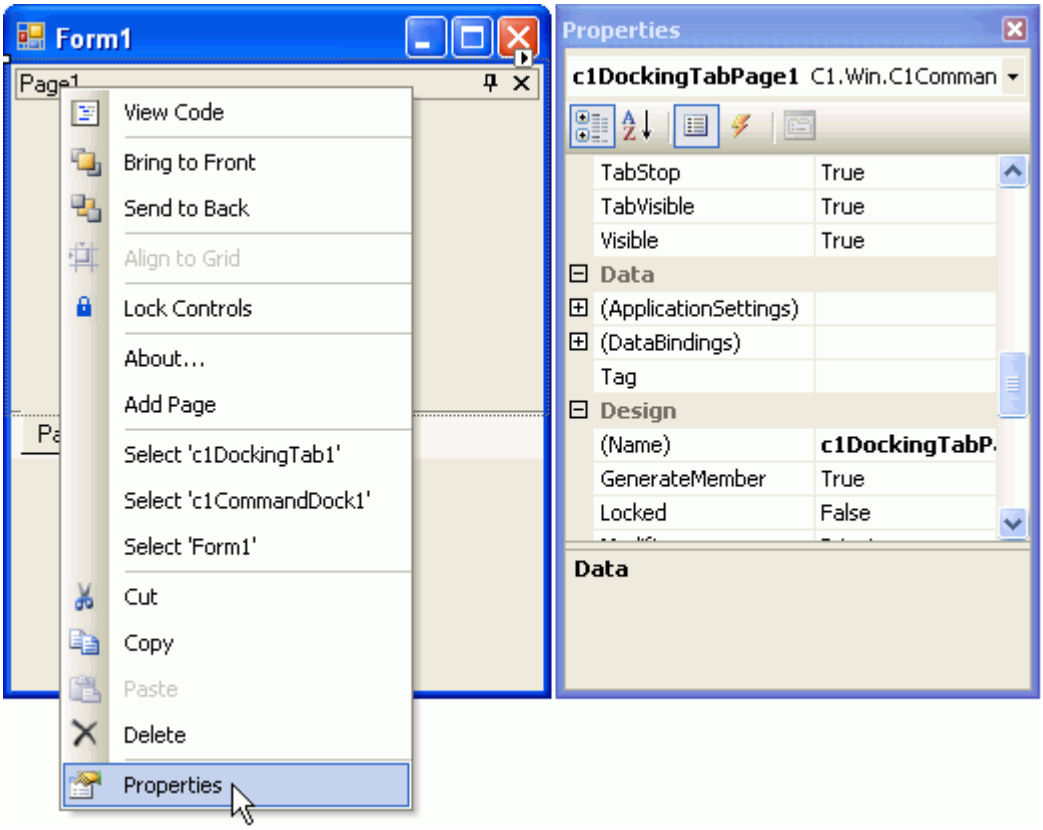
通过其编辑设计器，您可以容易地在设计时编辑C1CommandLinks。

设计时编辑器

C1Command提供七种集合编辑器：C1Command集合编辑器，C1CommandLink集合编辑器，C1DockingTabPage集合编辑器，C1NavBarButton集合编辑器，C1OutPage集合编辑器，C1TopicPage集合编辑器，和C1TopicLink集合编辑器。每一个编辑器的应用程序的主体部分是一个Windows 窗体，方便地允许用户编辑theC1MainMenu，C1ToolBar，C1DockingTab，C1NavBar，C1OutBar，或者C1TopicBar控件。

显示C1Command控件的属性

您可以简单地通过鼠标右键单击控件选择属性访问任何C1Commands组件的属性或者选中控件在属性窗体的下拉框选择类的属性。

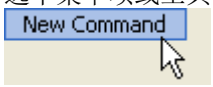


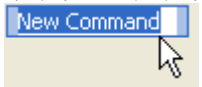
以下章节将详细介绍C1Command可用的每一种支持的类型。

就地文本编辑

相比跑到菜单或者工具栏的编辑器或者属性窗体去编辑文本属性，您可以就地直接编辑这些控件的文本属性。

要使用此功能，请按照以下步骤：

1. 选中菜单项或工具栏的项目，使其突出显示。
   

2. 按下ENTER键。
   菜单项或工具栏项的文本可以开始编辑。
   

3. 再次按Enter键接受变更或按ESC取消的变化。

C1Command智能标签

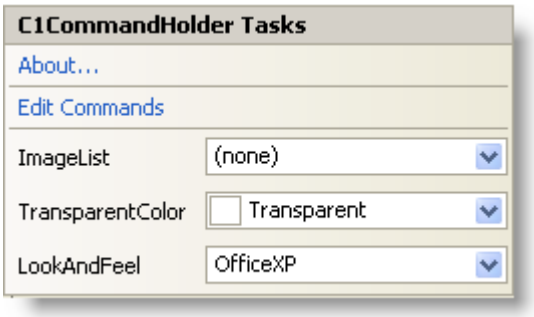
在Visual Studio中，C1Command的每一个组件/命令包括一个智能标记。智能标签代表了一个提供了每一个组件/命令最经常使用属性的一个快捷方式任务的菜单。

下面的部分介绍了每一个C1Command组件/命令的智能标签。

C1CommandHolder智能标签

通过智能标签，C1CommandHolder组件提供了对常用属性快速便捷的访问。

通过单击C1CommandHolder右上角的智能标签（）访问C1CommandHolderTasks菜单。这将打开C1CommandHolder的任务菜单。



C1CommandHolderTasks菜单可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置命令的外观样式。请注意，C1CommandHolder.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

ImageList

如果存在一个ImageList组件，可以选择ImageList的下拉按钮打开一个列表框，包含所有可用的图像列表项目。

TransparentColor

选择打开TransparentColor下拉框，打开具有Custom，选择transparentcolor下拉框打开自定义，网页列表框，和Custom系统的颜色选择。

Edit Commands

单击Edit Commands打开C1Command集合编辑器。关于如何使用C1Command集合编辑器的更多信息，请参见C1Command 集合编辑器。

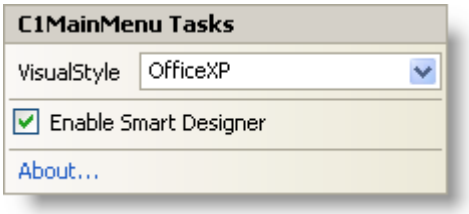
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1MainMenu 智能标签

通过智能标签，C1MainMenu控件提供了对常用属性快速便捷的访问。

通过单击C1MainMenu右上角的智能标签（）访问C1MainMenuTasks菜单。这将打开C1MainMenuTasks菜单。



C1MainMenuTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1MainMenu控件的外观样式。请注意，C1MainMenu.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

单击Smart Designer复选框启用C1MainMenu控件的智能设计器。

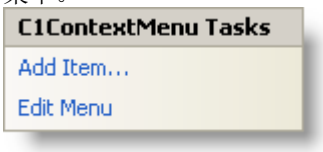
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1ContextMenu 智能标签

通过其LinktoCommand设计器，C1ContextMenu组件提供了方便快捷的添加并编辑菜单项的功能。这可以通过其智能标签完成。

通过单击C1ContextMenu组件右上角的智能标签（）访问C1ContextMenuTasks菜单。这将打开C1ContextMenuTasks菜单。



C1ContextMenuTasks可以执行的操作如下：

Add Item

单击Add Item将在当前的命令之后添加一个新的命令，并打开Link to Command编辑器。

关于如何使用Link to Command设计器的更多信息，请参见Link to Command设计器。

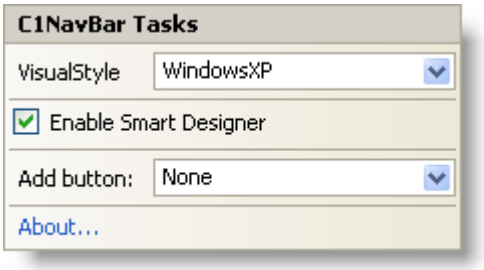
Edit Menu

单击Edit Menu项目将显示包含在关联到特定控件，比如说C1MainMenu或C1ToolBar，中的上下文菜单中的命令项目。

C1NavBar智能标签

通过智能标签，C1NavBar控件提供了对常用属性快速便捷的访问。

通过单击C1NavBar控件右上角的智能标签（）访问C1NavBarTasks菜单。



C1NavBarTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1NavBar控件的外观样式。请注意，C1NavBar.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

单击启用智能设计器复选框启用C1NavBar控件的智能设计器。

Add 按钮

单击Add按钮，下拉框将打开一个包含许多预先定义的按钮的列表框，允许您在以下选项中进行选择，Custom，Mail，Calendar，Contacts，Tasks，Notes，Folder，Shortcut，或者Journal。

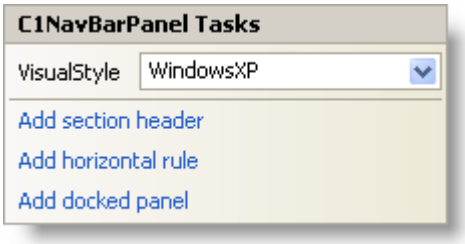
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1NavBar Panel智能标签

C1NavBarPanelTasks菜单使得修改视觉样式以及添加区域标题，水平标尺以及停靠面板更加容易。

通过单击C1NavBarPanel控件右上角的智能标签（）访问C1NavBarPanelTasks菜单。这将打开C1NavBarPanelTasks菜单。



C1NavBarPanelTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置章节标题、水平标尺、以及停靠面板的外观样式。请注意，C1NavBar.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

添加Section Header

单击Add Section Header项目向选中按钮的面板区域添加一个区域标题。

添加 Horizontal Rule

单击Add Horizontal Rule项目向选中按钮的面板区域添加一条横跨的水平线。

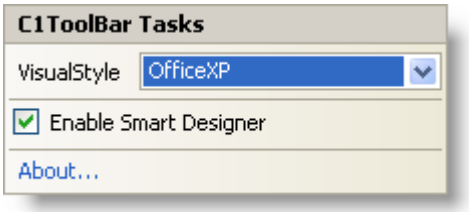
添加 Docked Panel

单击Add Docked Panel项目启用面板区域的可停靠功能。

C1ToolBar智能标签

C1ToolBar提供了对其常用属性的快速便捷地访问能力。

通过单击C1ToolBar控件右上角的智能标签（）访问C1ToolBarTasks菜单。这将打开C1ToolBarTasks菜单。



C1ToolBarTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1ToolBar.控件的外观样式。请注意，C1ToolBar.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

选中启用智能设计器复选框启用C1ToolBar控件的智能设计器。

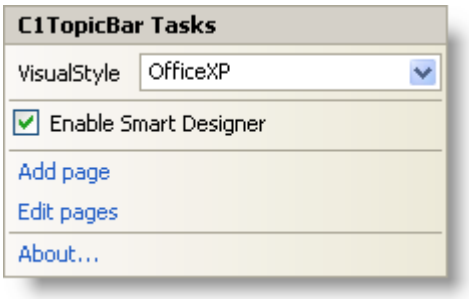
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1TopicBar智能标签

通过其智能标签，C1TopicBar组件提供对其C1TopicPageCollectionEditor以及其他许多常见的编辑行为的快速便捷地访问，比如说添加主题页。

通过单击C1TopicBar控件右上角的智能标签（）访问C1TopicBarTasks菜单。这将打开C1TopicBarTasks菜单。



C1TopicBarTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1TopicBar.控件的外观样式。请注意，C1TopicBar.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

单击启用智能设计器复选框启用C1TopicBar控件的智能设计器。

添加页面

单击AddPage项目，在C1TopicBar.上当前页面下方添加一个新的页面。

编辑页面

单击EditPage项目，将打开C1Topic Page Collection Editor。关于C1Topic Page Collection Editor的更多信息，请参见C1TopicPageCollectionEditor。

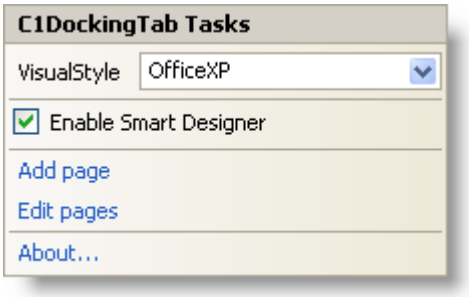
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1DockingTab智能标签

通过其智能标签，C1DockingTab 组件提供对其C1Docking Tab PagesCollectionEditor以及其他许多常见的编辑行为的快速便捷地访问，比如说添加新页。

通过单击C1DockingTab 控件右上角的智能标签（）访问C1DockingTabTasks菜单。这将打开C1TopicBarTasks菜单。



C1DockingTabTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置标签的外观样式。请注意，C1DockingTab.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

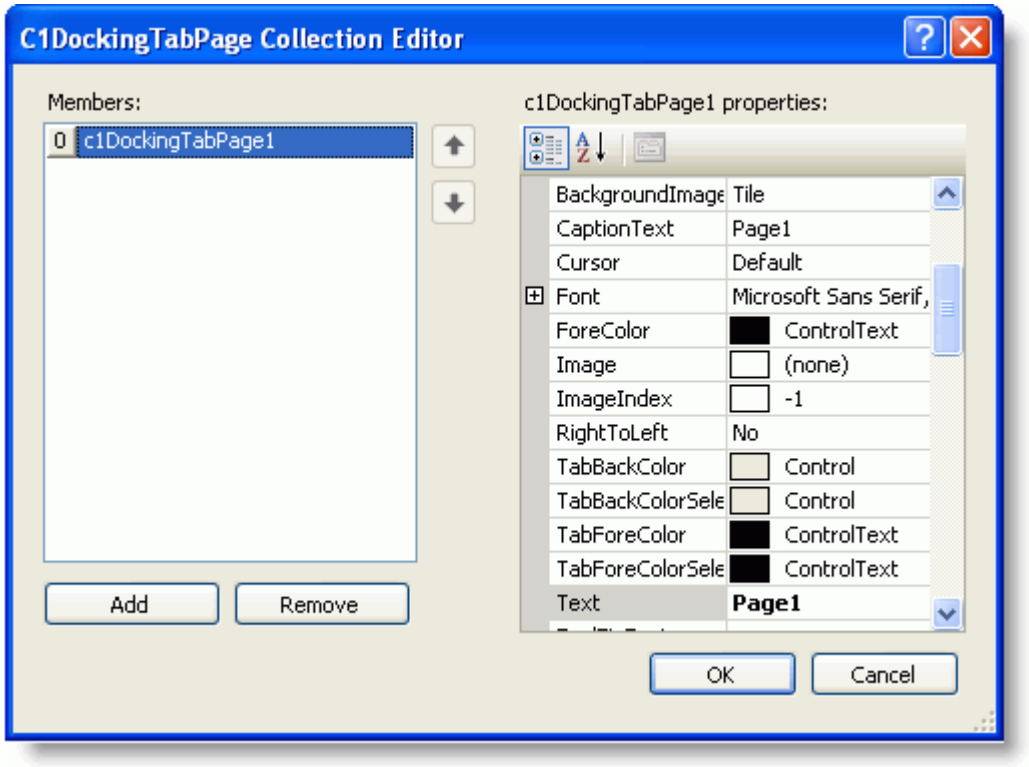
选择启用智能设计器复选框启用C1DockingTab控件的智能设计器。默认值为true（选中）。关于智能设计器元素的更多信息，请参见C1Command Smart Designer。

添加页面

单击Addpage添加一个新的C1Docking Tab Page。C1Docking Tab Page也具有智能标签。关于C1Docking Tab PageTasks菜单的更多信息，请参见C1Docking Tab Page智能标签。

编辑页面

选择Editpages可以打开C1Docking Tab Page集合编辑器。



关于如何使用C1Docking Tab Page集合编辑器的更多信息，请参见C1Docking Tab Page 集合编辑器。

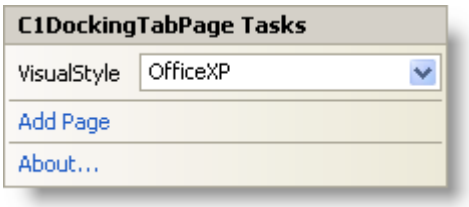
关于

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1DockingTabPage 智能标签

通过其智能标签，C1DockingTabPage提供了方便快捷的方式以改变其VisualStyle以及添加新的标签页。

通过单击C1DockingTabPage控件右上角的智能标签（）访问C1DockingTabPageTasks菜单。这将打开C1DockingTabPageTasks菜单。



C1DockingTabPageTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置标签的外观样式。请注意，

C1DockingTab.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

添加页面

单击AddPage添加一个新的C1DockingTabPage。

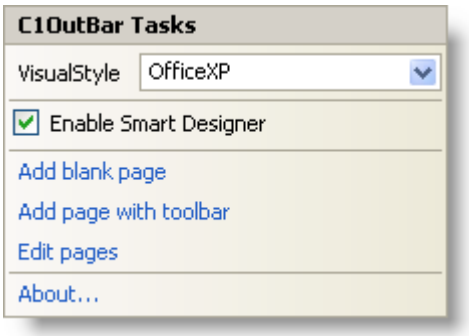
关于

单击About项目将显示AboutComponentOneCommand对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1OutBar智能标签

通过其智能标签，C1OutBar组件提供对其C1OutPageCollectionEditor以及其他许多常见的编辑行为的快速便捷地访问，比如说添加空白页或者添加具有工具栏的页面。

通过单击C1OutBar控件右上角的智能标签（）访问C1OutBarTasks菜单。这将打开C1OutBarTasks菜单。



C1OutBarTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1OutBar控件的外观样式。请注意，VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

启用智能设计器

选择Enable Smart Designer复选框可以启用C1OutBar控件的智能设计器。默认值为True（选中）。关于智能设计器元素的更多信息，请参见C1CommandSmartDesigner。

添加空白页

单击Add blank page将在当前的C1OutPage之后添加一个新的C1OutPage。单击新建的C1OutPage的内部将会显示一个可以弹出C1OutPageTasks菜单的智能标签。

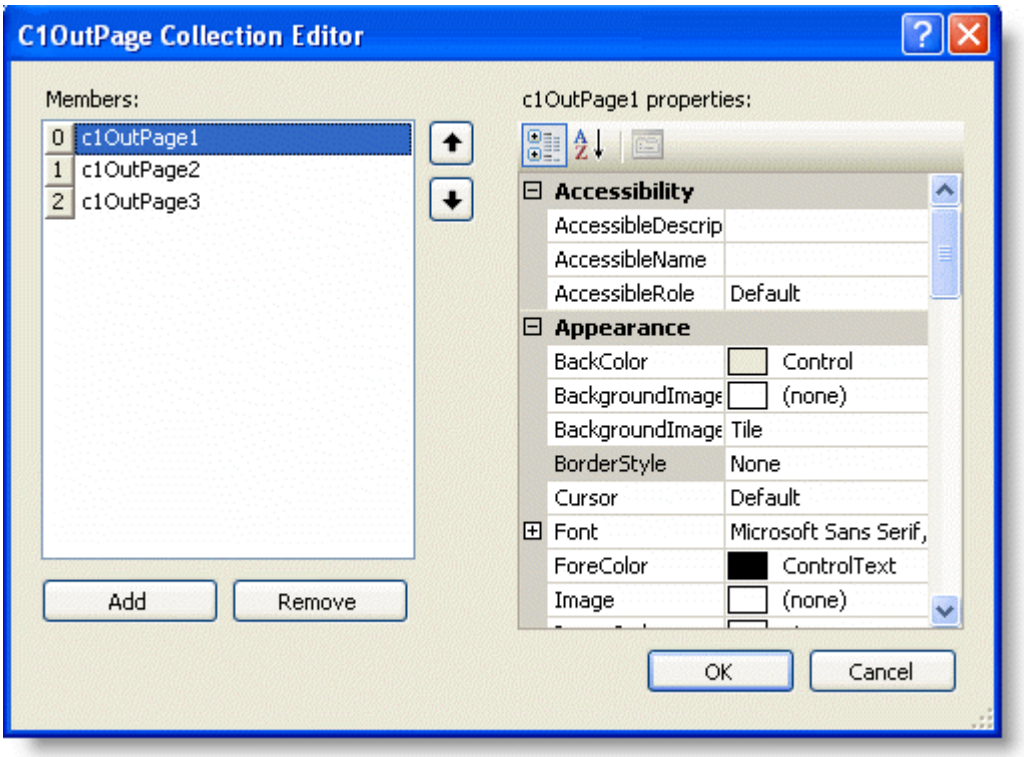
添加工具栏页面

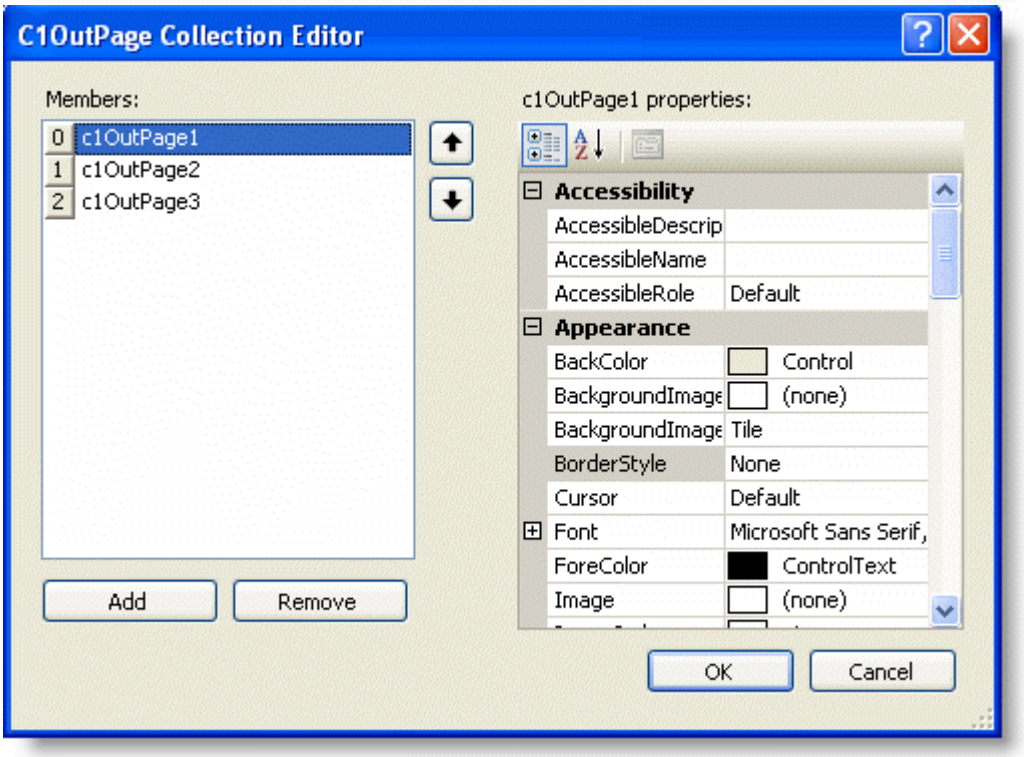
单击Add page with toolbar可以添加一个带有一个C1ToolBar的C1OutPage。新建的C1OutPage会出现一个C1OutPageTasks菜单。单击新建页面内部可以弹出C1ToolBarTasks菜单的智能标签。

关于如何使用C1ToolBarTasks菜单的更多信息，请参见C1ToolBar智能标签。

编辑页面

选择Edit pages将弹出C1OutPage集合编辑器。





关于如何使用C1OutPage集合编辑器的更多信息，请参见C1OutPage 集合编辑器。

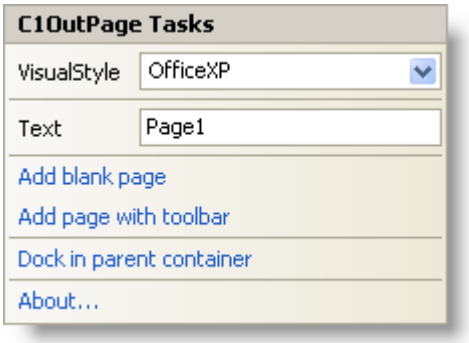
About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

C1OutPage 智能标签

通过其智能标签，C1OutPage控件提供了访问其C1OutPage Collection Editor以及其他许多常见的编辑行为，比如说添加一个空白页或者一个具有工具栏的页面，的方便快捷的方式。

通过单击C1OutPage控件右上角的智能标签（）访问C1OutPageTasks菜单。这将打开C1OutPageTasks菜单。



C1OutPageTasks可以执行的操作如下：

VisualStyle

选择VisualStyle的下拉框打开一个列表框，该列表框有几个项目可以进行选择（Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP），以设置C1OutPage控件的外观样式。请注意，C1OutBar.VisualStyle以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们已经被废弃。

文本

选中Text框，并输入C1OutPage的名称

添加空白页

单击Add blank page添加一个新建空白的C1OutPage（不包含C1ToolBar），该新建页面位于当前的C1OutPage之后。

添加工具栏页面

单击Add page with toolbar向当前C1OutPage位置之后添加一个新的C1OutPage。同时向其添加一个新建的C1ToolBar。

父容器停靠

单击Dock in parent container 将C1OutPage停靠到其父容器中。

About

单击About项目将显示About ComponentOne Command对话框，它可以帮助您查找C1Command版本号以及一些在线资源。

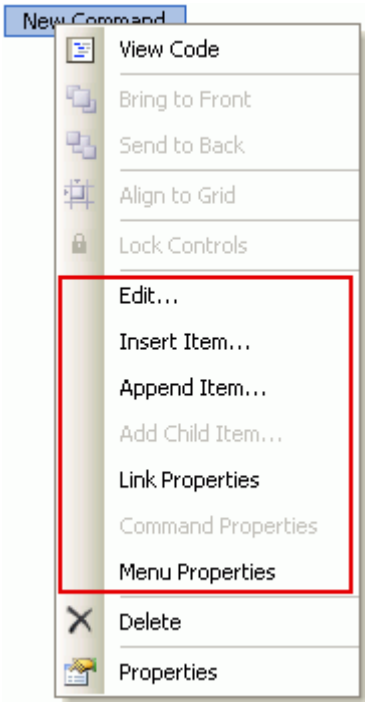
C1Command上下文菜单

每一个C1Command组件同时提供了一个上下文菜单，提供了设计时使用的额外功能。

Command上下文菜单

右键单击一个菜单项或者工具栏按钮的command link项将打开以下上下文菜单。

> **注意：**菜单上的CommandLinks和位于工具栏上的CommandLinks唯一的区别在于Menu



下表提供了一个C1MainMenu以及C1ToolBar控件在其设计时添加项目的简短描述：

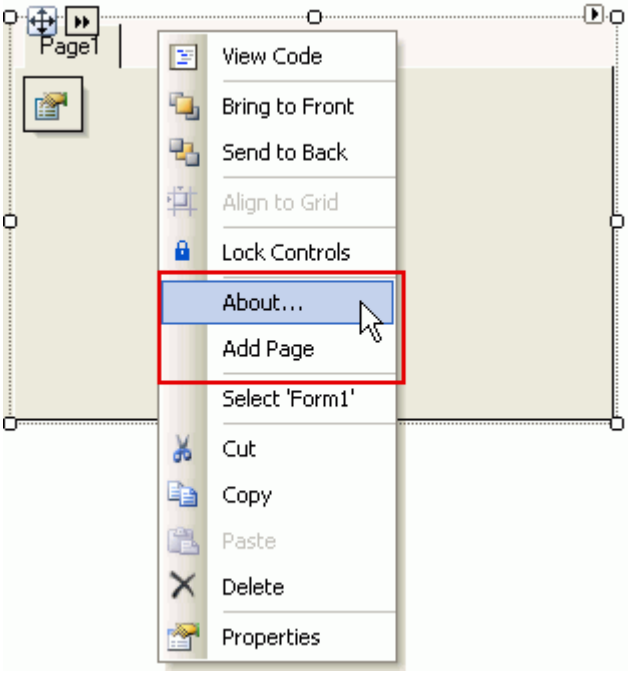
自定义项目	描述
Edit	为当前选中的commandlink调出Link to Command设计器。您可以编辑当前链接的命令，或链接到另一个（新的或现有的）命令。如果没有commandlink被选中，则该项目被禁用。
Insert Item	在当前commandlink之前插入一个新的，并打开Link to Command设计器。如果没有commandlink被选中，则该项目被禁用。
Append Item	在当前菜单或者工具栏最有一个项目的末尾，添加一个新的commandlink，并为这个新的commandlink调出命令编辑设计器。

Add Child Item	添加一个新的command link至菜单（C1CommandMenuor或者C1ContextMenu），关联到当先选中的commandlink，之后将打开Link to Command设计器。该项目为禁用状态，除非当前的command link链接到一个C1CommandMenuor或者C1ContextMenu类型的命令。
Link Properties	选择命令链接并在属性窗体显示其属性。
Command Properties	选中命令并在属性窗体显示其属性。注意：对于非空链接，点击一个链接将在显示链接自己的属性以及链接目标命令的属性的状态之间切换。
Menu Properties	选择菜单并在属性窗体显示其属性。

注意：所有的菜单编辑操作通过VisualStudio本身的Undo功能，可以完整的支持撤销。

C1DockingTab上下文菜单

右键单击C1DockingTab控件可以打开其上下文菜单。

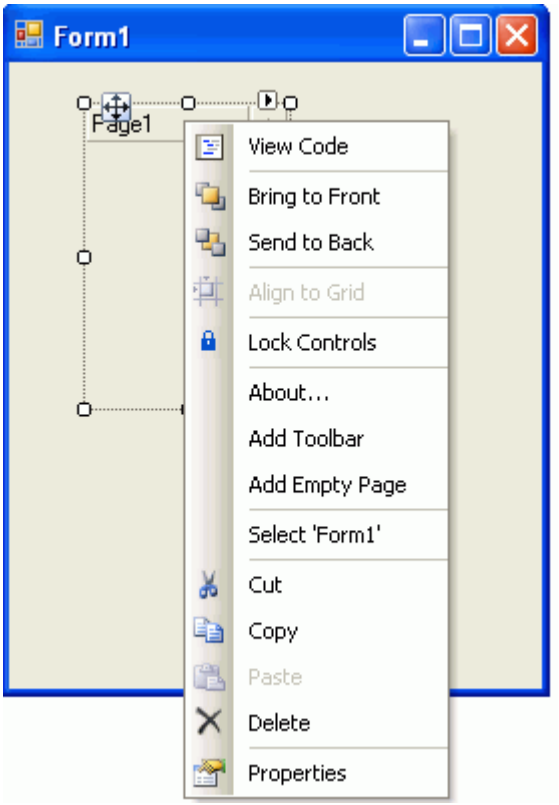


下表提供了一个关于C1DockingTab控件添加到其上下文菜单的自定义项目的简短的描述：

自定义项目	描述
About	显示AboutComponentOneCommand对话框，它可以帮助您查找C1Command版本号以及一些在线资源。
Add Page	添加一个新页面到C1DockingTab.

C1OutBar 上下文菜单

右键单击C1OutBar控件可以打开其上下文菜单。

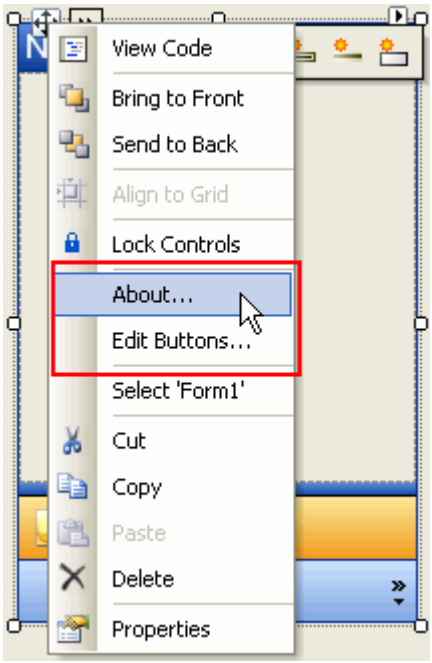


下表提供了一个关于C1OutBar控件添加到其上下文菜单的自定义项目的简短的描述：

自定义项目	描述
About	打开AboutComponentOneCommand对话框，它可以帮助您查找C1Command版本号以及一些在线资源。
Add Toolbar	添加一个带有C1ToolBar的新C1OutPage页面。
Add Empty Page	添加一个不带C1ToolBar的C1OutPage页面。

C1NavBar上下文菜单

右键单击C1NavBar控件可以打开其上下文菜单。

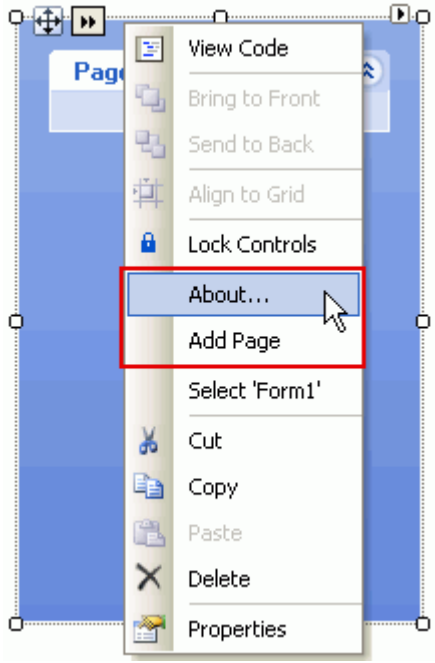


下表提供了一个关于C1NavBar控件添加到其上下文菜单的自定义项目的简短的描述：

自定义项目	描述
About	打开AboutComponentOneCommand对话框，它可以帮助您查找C1Command版本号以及一些在线资源。
Edit Buttons	打开C1NavBarButton集合编辑器。

C1TopicBar上下文菜单

右键单击C1TopicBar控件可以打开其上下文菜单。



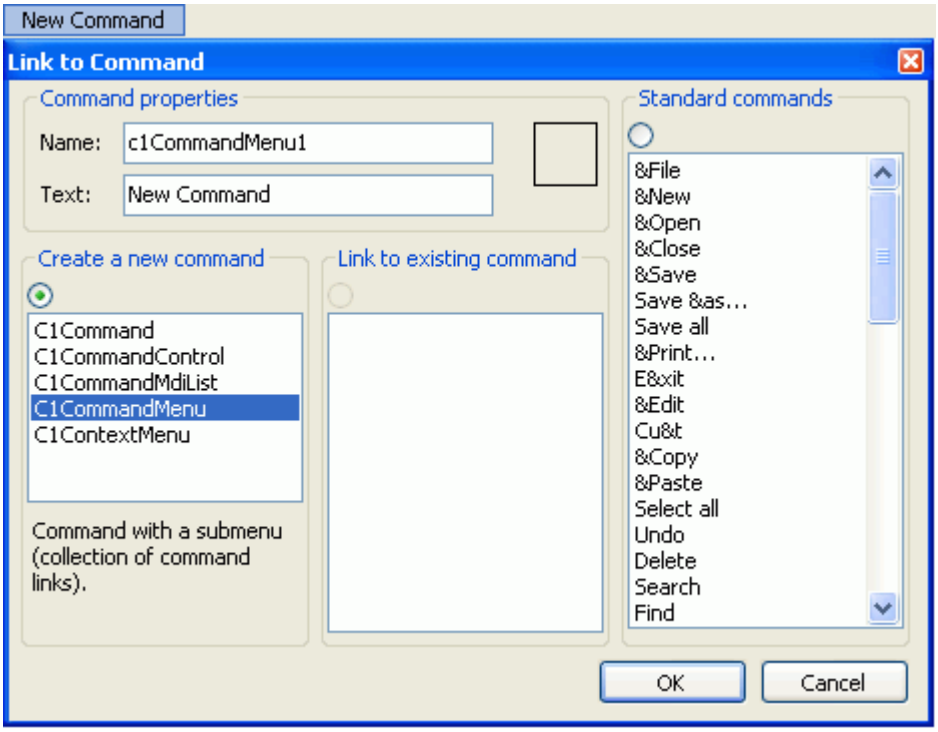
下表提供了一个关于C1TopicBar控件添加到其上下文菜单的自定义项目的简短的描述：

自定义项目	描述
About	打开AboutComponentOneCommand对话框，它可以帮助您查找C1Command版本号以及一些在线资源。
Add Page	向C1TopicBar控件添加一个新的C1TopicPage。

访问编辑设计器

访问编辑设计器:

双击菜单或者工具栏控件上任何命令项或者右键单击并从其上下文菜单选择"编辑"以打开Link to Command设计器。



此对话框用于创建新的命令，编辑现有的链接，或者链接命令链接至某个命令。同样的对话框被用作编辑下拉菜单以及工具栏上的command link。Link to Command设计器包含以下类别：命令属性，创建一个新的命令，链接到现有的命令，和标准的命令。

下表列出并描述了在Link to Command设计器中各个项目的功能：

分组框	描述
Command properties	Command属性分组框包含一个Name文本框以及一个Text文本框。Name文本框显示该命令的默认命令名称，或者您可以输入一个新的命令名称。Text文本框显示出现在命令项中的文本。
Create New Command	Create New Command分组框包含一个各种不同命令的列表框，可以从中选择一个特定的命令类型用作新建的命令。单选按钮允许您在创建一个新的命令和链接到一个现有的命令（位于现有命令列表框）之间进行切换。
Link to Existing Command	Link to existing command分组显示窗体已经创建的命令的列表框。当您在该列表中进行选择时，Create New Command单选框将自动清除选择状态。

Standard Commands	Standard command分组框提供了54种内置的命令以及40种标准命令的图片。当您从Standard commands列表选择了一个包含内置图片的标准命令时，将在Command属性分组框下方的方框中显示预览图片。

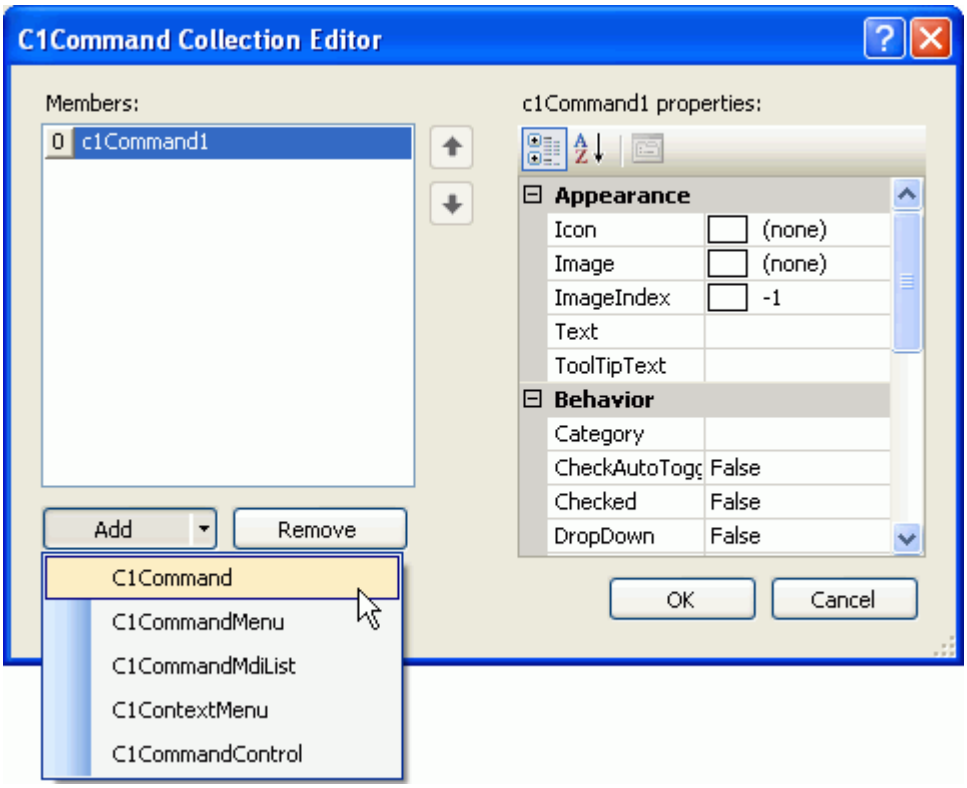
C1Command 集合编辑器

C1Command 提供以下七种类型的集合编辑器：C1Command 集合编辑器，C1CommandLink 集合编辑器，C1DockingTabPage 集合编辑器，C1NavBarButton 集合编辑器，C1OutPage 集合编辑器，C1TopicPage 集合编辑器，以及C1TopicLink 集合编辑器。每一个编辑器的应用程序的主体部分是一个Windows窗体，方便地允许用户编辑C1MainMenu, C1ToolBar, C1DockingTab, C1NavBar, C1OutBar, 或者C1TopicBar控件。

以下部分简要介绍了C1Command 集合编辑器并解释了如何访问并使用它们。

C1Command 集合编辑器

C1Command 集合编辑器允许用户编辑命令属性，并添加C1Command，C1CommandMenu，C1CommandMdiList，C1ContextMenu，以及层次结构的C1CommandControl类型。

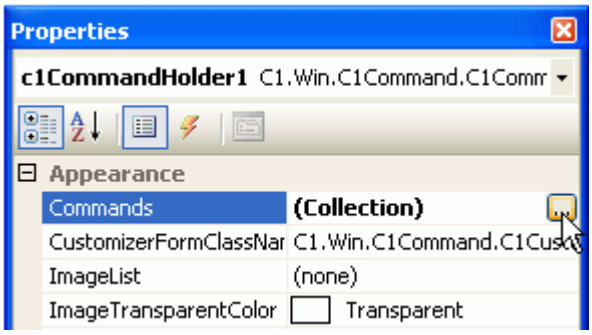


访问C1Command 集合编辑器：

1. 向窗体添加一个C1MainMenu或者C1ToolBar。

   这将自动地在窗体的组件区域创建一个C1CommandHolder组件。

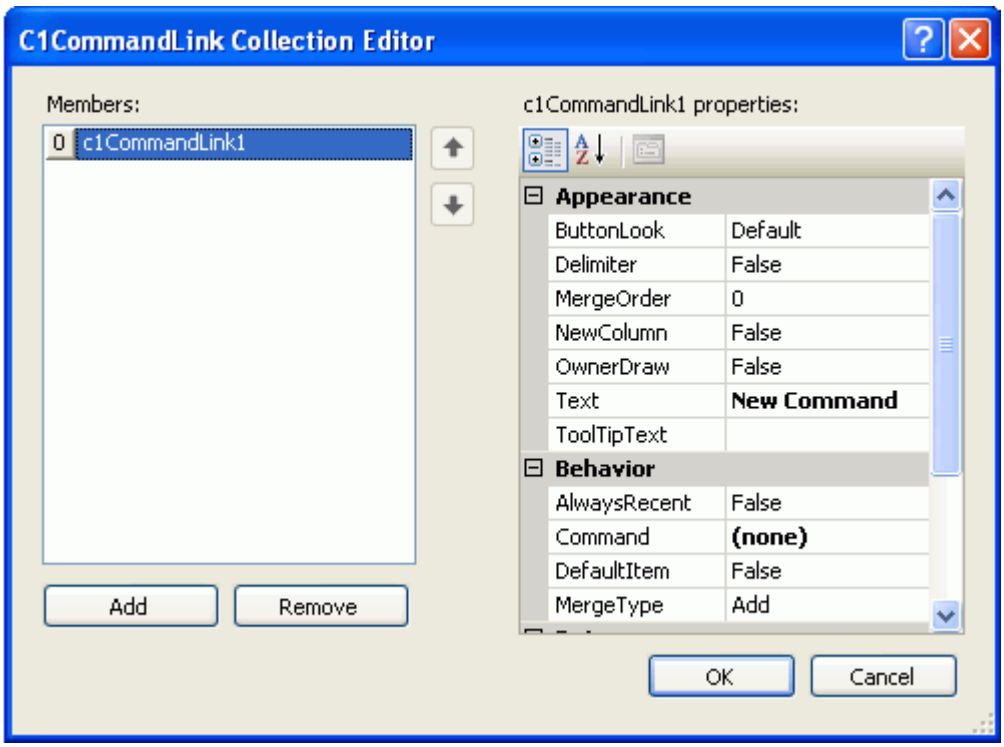
2. 在C1CommandHolders的属性窗体中，单击Commands属性旁边的省略号按钮。

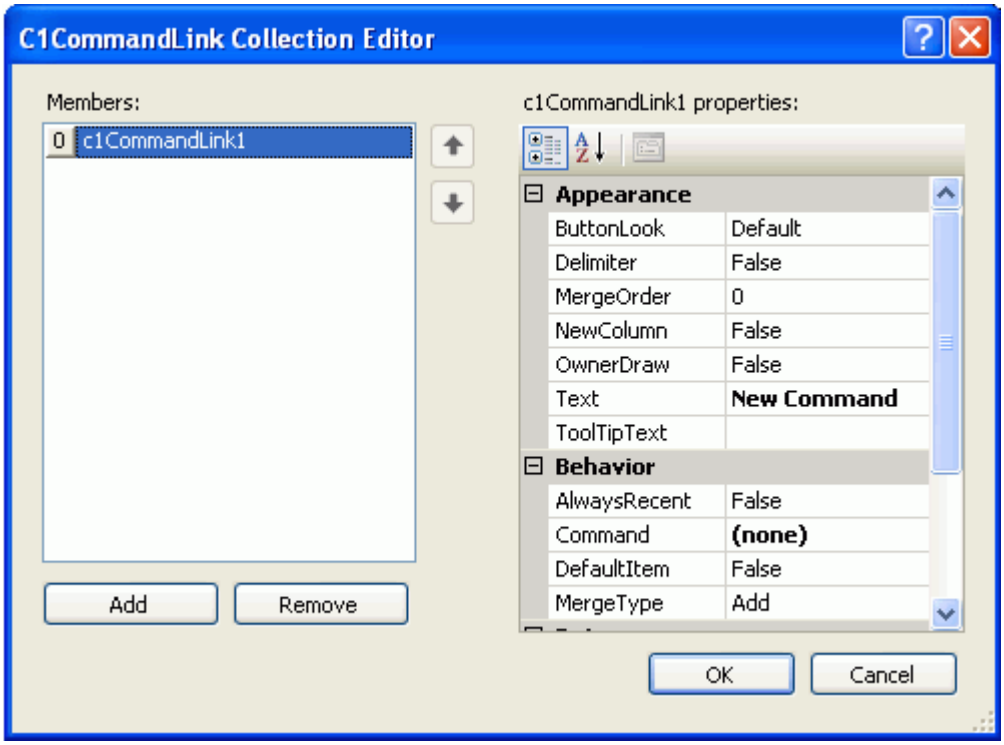


将出现C1Command集合编辑器。

C1CommandLink集合编辑器

C1CommandLink集合编辑器允许用户添加或删除命令链接或者链接已存在的命令。



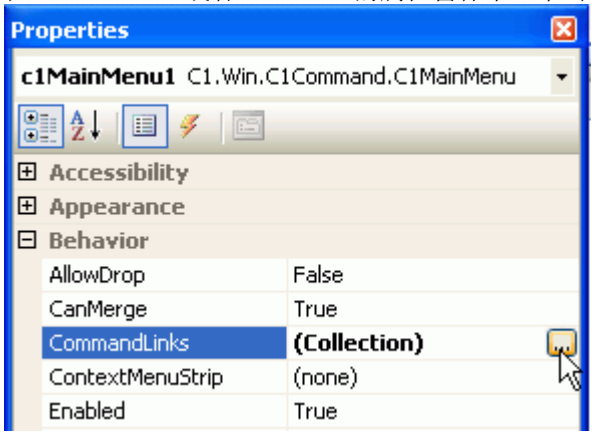


访问C1CommandLink集合编辑器:

1. 向窗体拖拽添加一个C1MainMenu或者一个C1ToolBar。

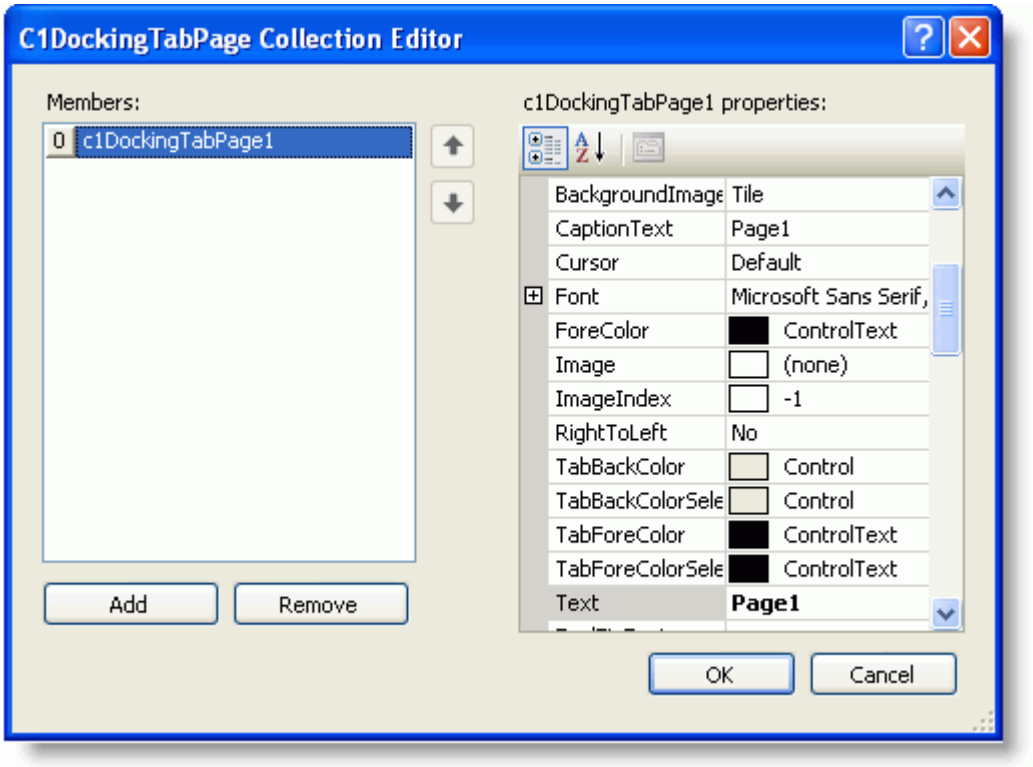
   这将自动地在窗体的组件区域创建一个C1CommandHolder组件。

2. 在C1MainMenu或者C1ToolBar的属性窗体中，单击CommandLinks属性旁边的省略号按钮。



C1DockingTabPage集合编辑器

C1DockingTabPage集合编辑器允许用户添加或者删除标签页，或者编辑标签页的属性。

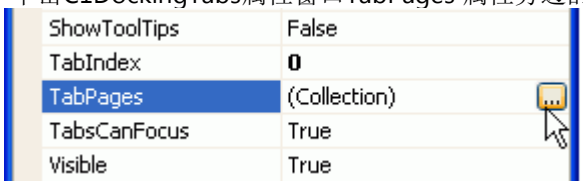


访问C1DockingTabPage集合编辑器:

有两种方法来访问C1DockingTabPage 集合编辑器，可以通过"属性"窗口中的TabPage属性，或编辑位于C1DockingTab任务菜单标签页菜单项。关于如何使用C1DockingTab 任务菜单的更多信息，请参见C1DockingTab智能标记。

选项1:

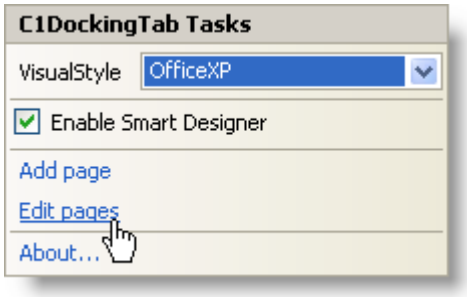
1. 向窗体拖拽添加一个C1DockingTab。
2. 单击C1DockingTabs属性窗口TabPages 属性旁边的省略号按钮。



将出现C1DockingTabPage集合编辑器。

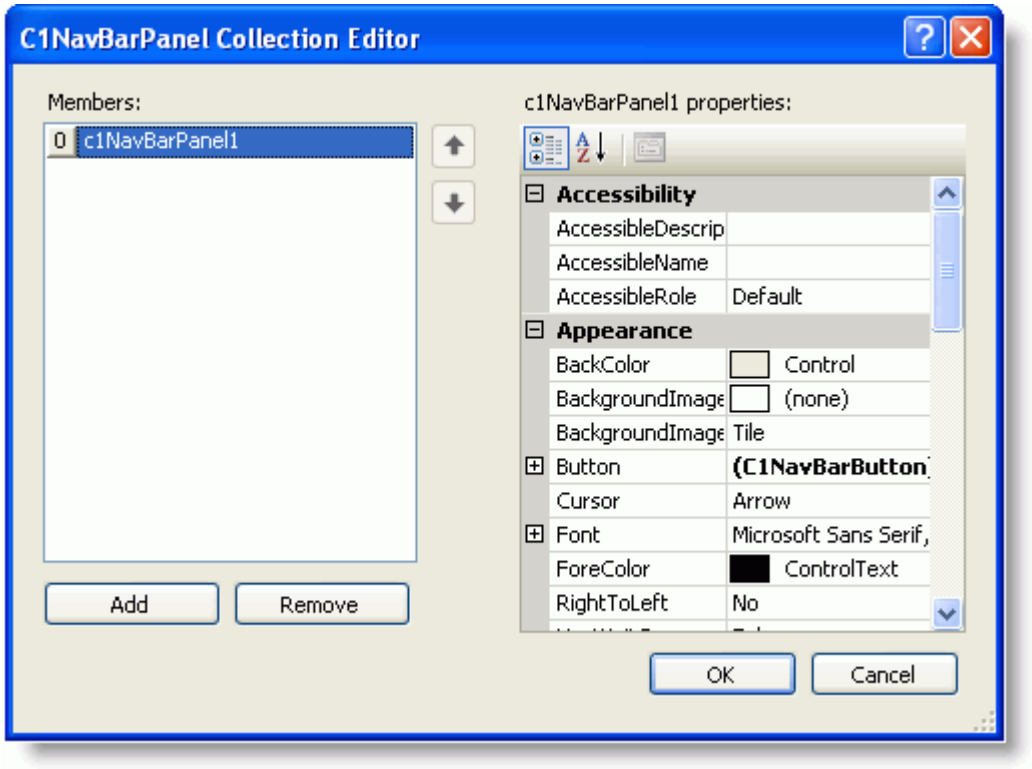
选项2

1. 向窗体拖拽添加一个C1DockingTab控件。
2. 单击C1DockingTab控件右上角的智能标签（▶），然后从C1DockingTab任务菜单单击Edit pages。



C1NavBarPanel集合编辑器

C1NavBar集合编辑器允许用户添加删除C1NavBar控件的面板，同时也可以编辑面板的属性。

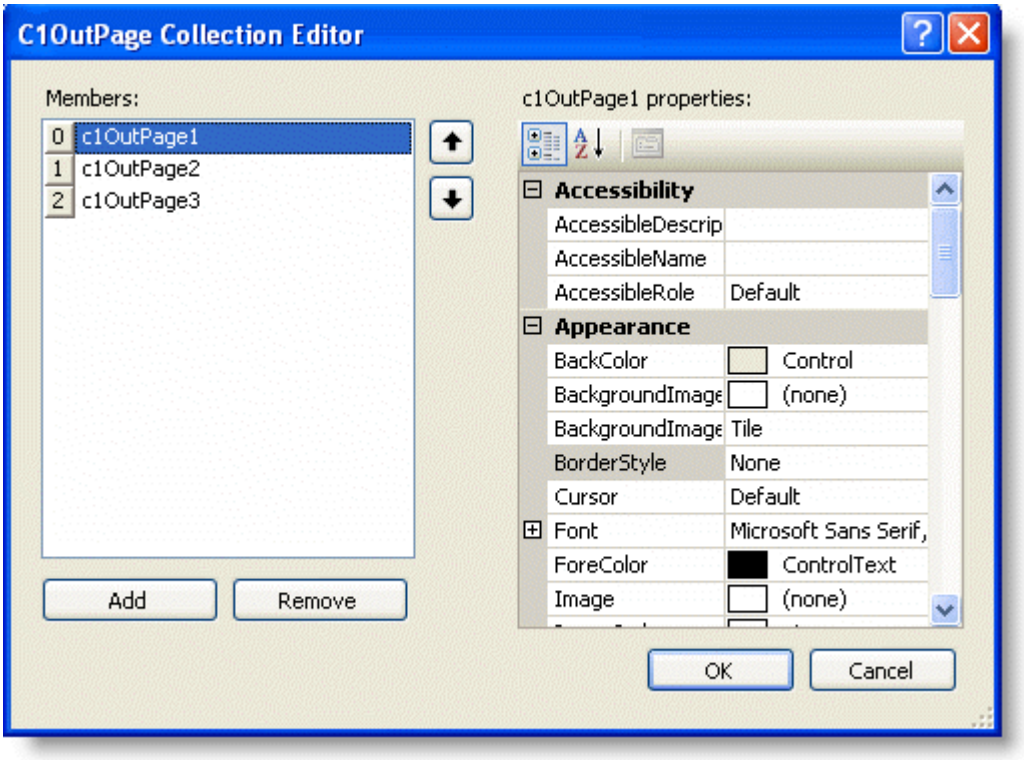


访问C1NavBar集合编辑器:

1. 向窗体拖拽添加一个C1NavBar。
2. 在C1NavBar的属性窗体单击Panels属性旁边的省略号按钮。
   将出现C1NavBar集合编辑器。

C1OutPage集合编辑器

C1OutPages集合编辑器允许用户添加或删除位于C1OutBar控件中的页，同时允许编辑每一个页面的属性。

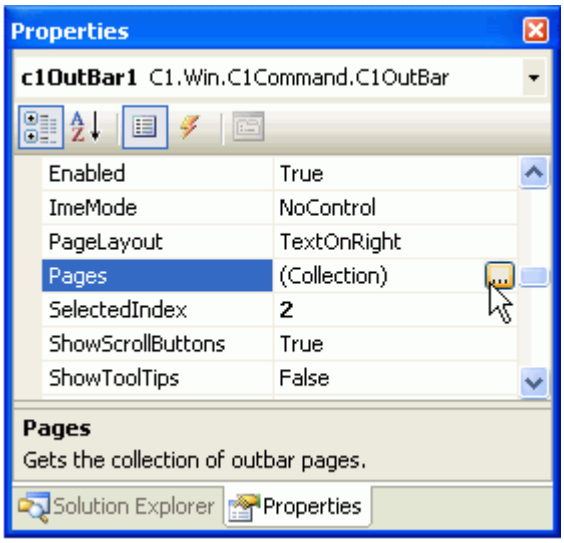


访问C1OutPage集合编辑器:

有两种方法来访问C1OutPage 集合编辑器，可以通过属性窗体中的Pages属性，或编辑页的C1OutPage 任务菜单。关于如何使用C1OutBar 任务菜单的更多信息，请参见C1OutBar Smart Tag。

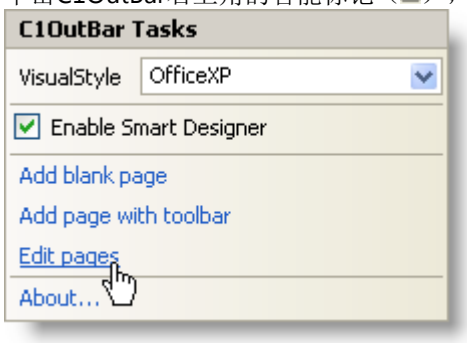
选项1

1. 向窗体拖拽添加一个C1OutBar控件。
2. 单击位于C1OutBar属性窗体Pages属性旁边的省略号按钮。
   将出现C1OutPage 集合编辑器。



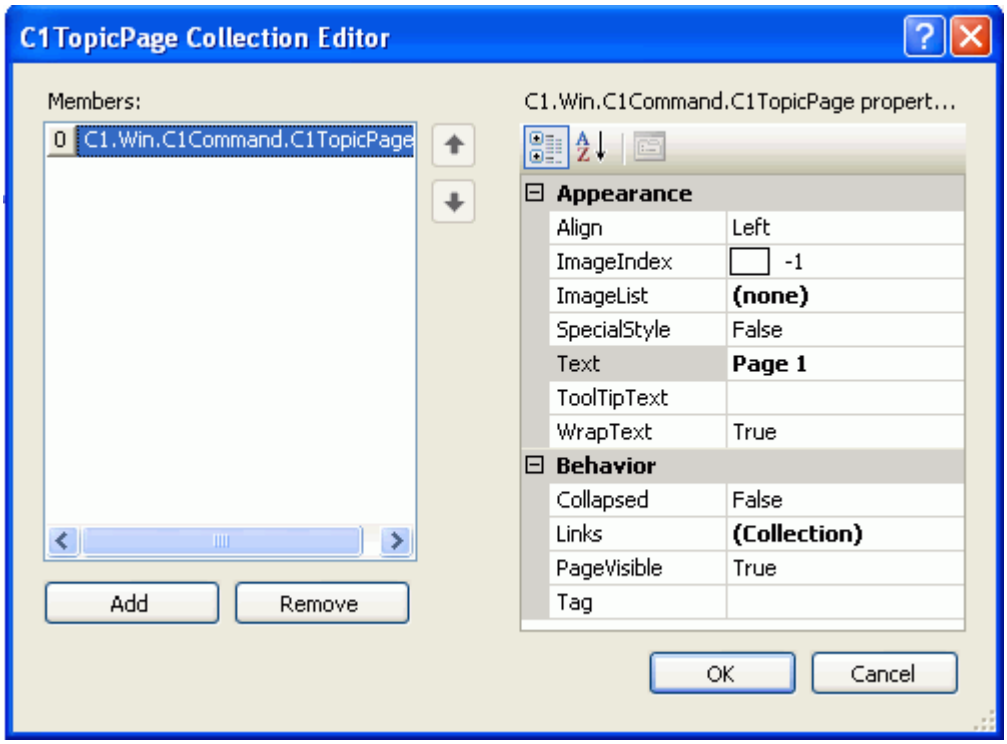
选项2

1. 向窗体拖拽添加一个C1OutBar控件。
2. 单击C1OutBar右上角的智能标记（），然后从C1OutBar的任务菜单单击Edit Pages。



C1TopicPage集合编辑器

C1TopicPage集合编辑器允许用户添加删除C1TopicBar控件中的页，同时允许编辑每一个页面的属性。



访问C1TopicPage集合编辑器:

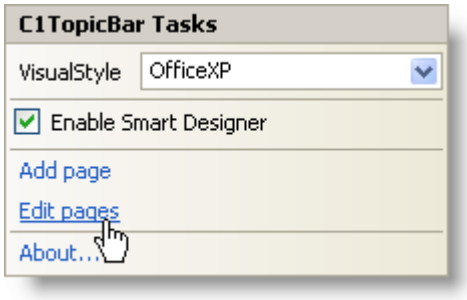
有两种方法来访问C1TopicPage 集合编辑器，可以通过属性窗体中的Pages属性，或编辑页的C1TopicPage任务菜单。关于如何使用C1TopicPage 任务菜单的更多信息，请参见C1TopicPage 智能标记。

选项1

1. 向窗体拖拽添加一个C1TopicBar。
2. 单击位于C1TopicBars属性窗体Pages属性旁边的省略号按钮。

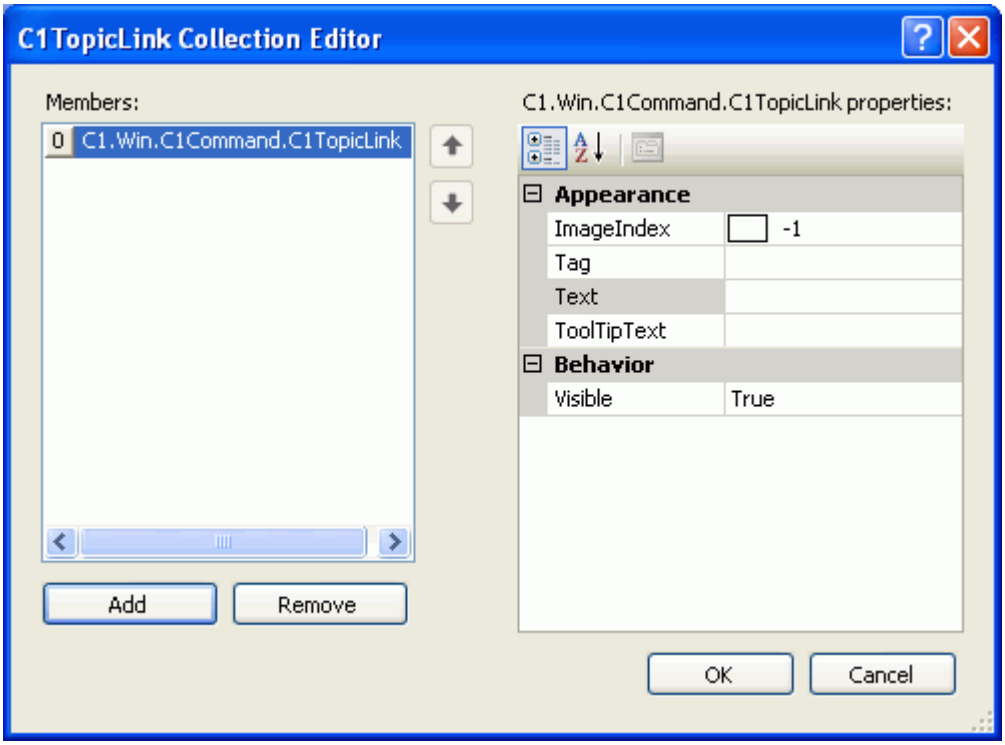
选项2

1. 向窗体拖拽添加一个C1TopicBar。
2. 单击C1TopicBar控件右上角的智能标记，然后从C1TopicBar 的任务菜单单击Edit Pages。



C1TopicLink集合编辑器

C1TopicLink集合编辑器允许用户添加位于C1TopicBar控件的链接，同时可以编辑每一个链接的属性。



访问C1TopicLink集合编辑器:

您可以通过位于C1TopicPages集合编辑器的Links属性访问C1TopicLink 集合编辑器。

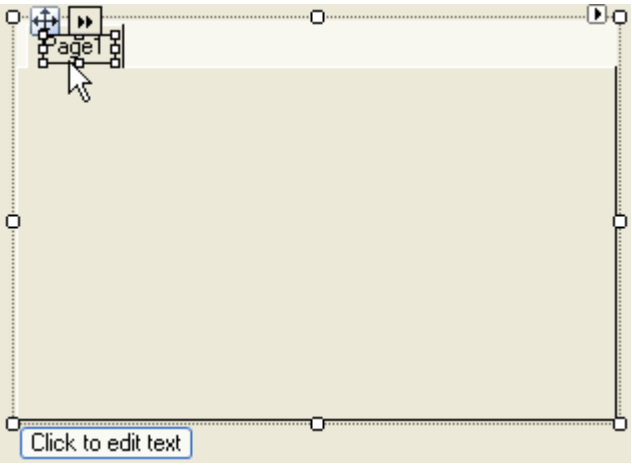
1. 向窗体拖拽添加一个C1TopicBar。
2. 单击C1TopicBar控件右上角的智能标记（），然后从C1TopicBar的任务菜单单击Edit Pages。单击Links属性旁边的省略号按钮。

C1Command智能设计器

C1Command提供了一个智能设计器，以提高设计时C1MainMenu, C1ToolBar, C1DockingTab, C1OutBar, C1NavBar, 以及 C1TopicBar控件的交互性。当您选择C1Command控件之一，并用鼠标悬停经过它们时，将出现一个Open按钮 。单击Open按钮将打开和窗体上选中控件相关联的浮动工具栏。

每一个工具栏为其每一个工具栏项目提供了工具条提示，以提高用户交互性。此外，每一个工具栏提供了命令按钮以及常用的属性对话框，以便在不离开设计器窗体的情况下快速配置C1Command控件。这解决了之前必须在属性窗体中层层深入设置C1Command控件属性的问题。

除了内置的工具栏，对话框以及命令按钮，C1DockingTab控件的智能设计器同时包含自定义标签的指示，让您的设计时用户体验更加直观。当您的鼠标悬停在C1DockingTab控件上方时，将出现一个带有简单命令语句的标签，指示您应当执行什么样的动作。

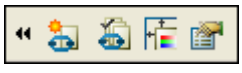


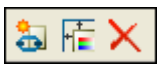
您可以通过智能设计器功能在设计时创建一个功能菜单，工具栏，停靠面板，outbar，导航栏或者主题栏。

该章节描述了关联到每一个C1Command控件的智能设计器的工具栏以及如何弹出每一个浮动工具栏。

智能设计器浮动工具栏

C1Command套件的智能设计器由以下浮动工具栏组成：

浮动工具栏	描述
	C1MainMenu工具栏：C1MainMenu 工具栏允许您添加新的命令链接，编辑命令链接，编辑主菜单外观以及各种主菜单属性。
	C1CommandLink/C1Command工具栏：C1CommandLink 工具栏允许您添加，移除，或者改变现有的命令。
	C1CommandMenu工具栏：C1CommandMenu工具栏允许您添加新的命令链接，编辑命令链接，编辑命令属性，编辑C1CommandMenu的外观，以及编辑副标题的属性。
	C1ToolBar工具栏：C1ToolBar工具栏允许您编辑或者向工具栏添加命令链接，设置工具栏的外观属性，启用或禁用合并，折行或者工具栏按钮的工具条提示，并设置工具栏按钮的颜色以及字体样式。
	C1DockingTab工具栏：C1DockingTab工具栏允许您添加标签页至C1DockingTab控件，启用或禁用C1DockingTab行为设置，以及设置C1DockingTab控件的颜色和字体样式。
	C1DockingTabPage工具栏：C1DockingTabPage工具栏允许您设置标签页的颜色和字体样式。

	C1NavBar工具栏：C1NavBar工具栏允许您向C1NavBar添加按钮，设置C1NavBar的外观属性以及设置其他各种属性。
	C1NavBarButton工具栏：C1NavBarButton工具栏允许您修改C1NavBar控件上选中按钮的属性。
	C1OutBar工具栏：C1OutBar工具栏允许您添加空白页或者带有工具栏的页面，设置C1OutBar的外观属性，以及设置C1OutBar控件的其它属性。
	C1OutPage工具栏：C1OutPage工具栏允许您应用属性设置至选中的C1OutPage。
	C1TopicBar工具栏：C1TopicBar工具栏允许您添加新的页面链接至C1TopicBar，编辑页面，编辑C1TopicBar 的外观和布局，以及编辑C1TopicBar的其他属性。
	C1TopicPage工具栏：C1TopicPage工具栏允许您添加一个主题链接，删除主题页面，或者编辑C1TopicPage的外观属性。
	C1TopicLink工具栏：C1TopicLink工具栏允许您修改选中主题链接的外观或者移除现有的主题链接。

智能设计器的工具栏被叫做浮动工具栏，是因为它们的行为和
典型的工具栏有一点点不同。智能设计器的工具栏只有在控件被激活时出现，同时它们不能停靠到其他控件上。

本章节描述了每一个浮动工具栏上按钮的功能。

C1MainMenu工具栏

C1MainMenu工具栏将出现在C1MainMenu控件上。为显示C1MainMenu工具栏，请选择C1MainMenu控件并移动鼠标至C1MainMenu上。此时将出现打开C1MainMenu工具栏的Open按钮 。

打开和关闭C1MainMenu工具栏

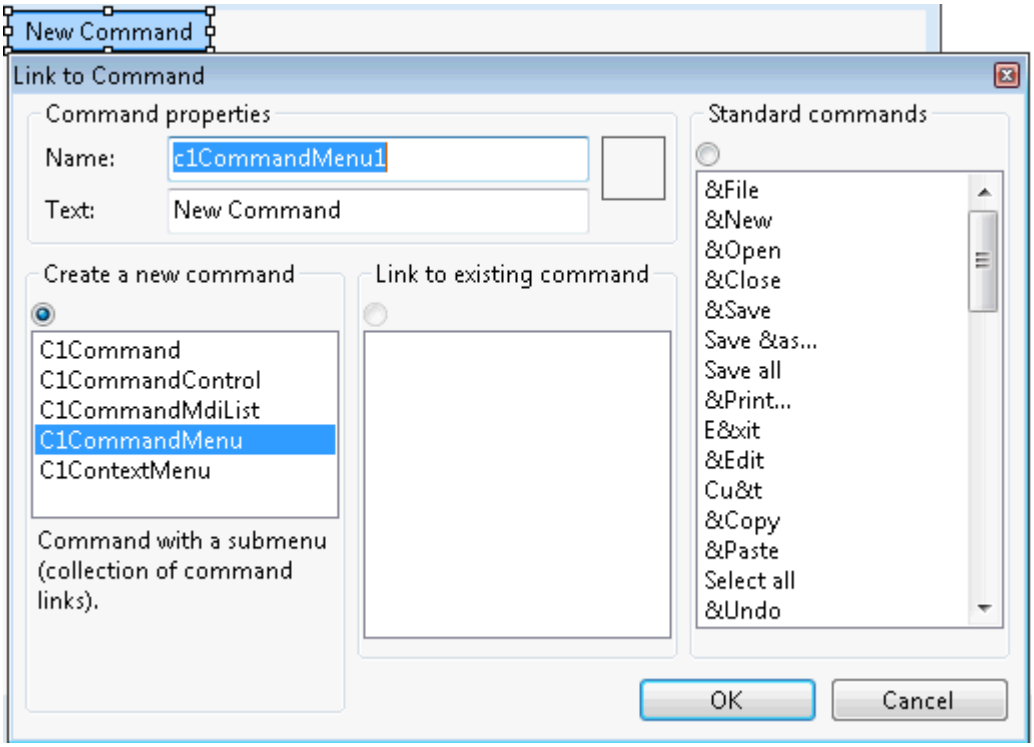
为打开C1MainMenu工具栏，点击  按钮。为关闭C1MainMenu工具栏，点击按钮。

C1MainMenu工具栏包含以下命令按钮：

工具栏按钮	描述
	添加新的命令链接/命令：向主菜单添加一个新的命令，位于当前命令位置之后。
	编辑命令链接：打开C1CommandLink 集合编辑器以编辑命令链接。
	编辑主菜单的外观：打开C1MainMenu外观对话框，您可以为C1MainMenu控件设置常规的外观属性。
	编辑杂项属性：打开C1MainMenu的杂项对话框，在这里您可以应用行为设置至C1MainMenu控件。

添加新的命令链接/命令

点击添加新的命令链接/命令按钮在当前命令之后添加一个新命令。在新建命令下方显示到Command设计器的链接，因此您可以很容易地编辑新建命令，而不需要离

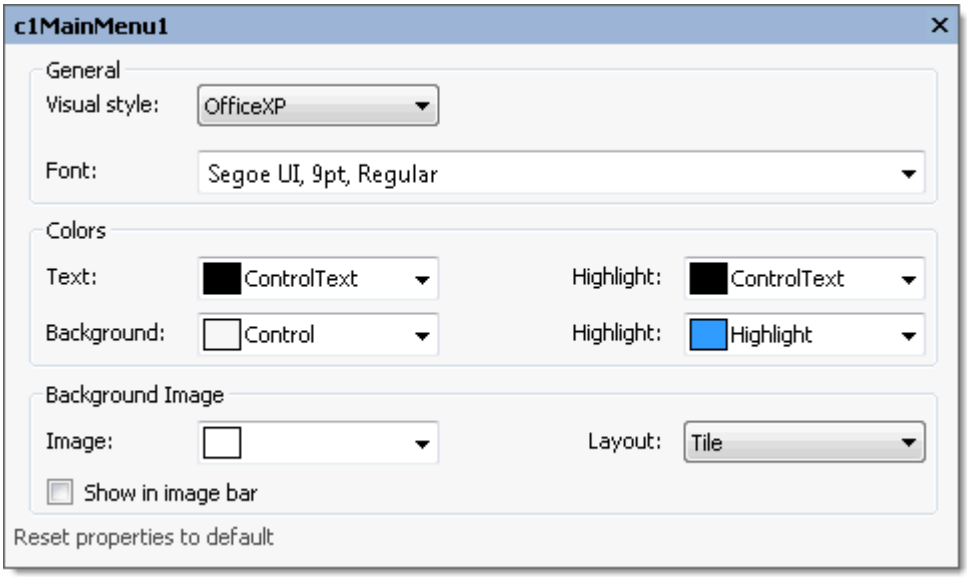


编辑命令链接

点击编辑命令链接按钮打开C1CommandLink 集合编辑器，在此您可以添加或删除命令链接并编辑commandlink的属性。

编辑主菜单的外观

点击编辑主菜单外观按钮可以打开C1MainMenu外观对话框，在此您可以修改菜单项的外观属性。

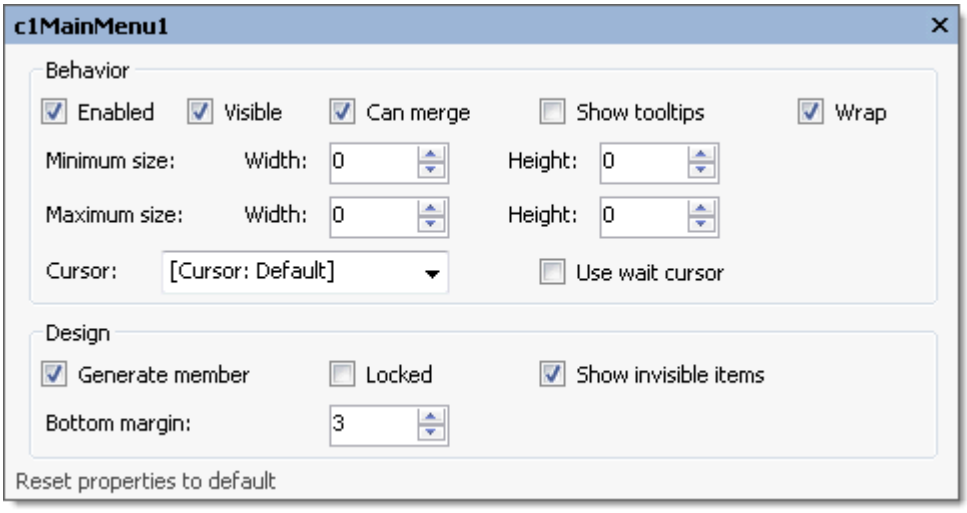


下表定义了包含在外观对话框中的项目：

项目	描述
通用	
Text	Text文本框显示与C1MainMenu控件关联的文本名称。为了重命名命令的文本名称，选中Text文本框中的文本，并输入期望的文本名称。
Font	字体下拉框将打开字体对话框，这里您可以修改C1MainMenu控件的字体样式属性。
颜色	
Foreground	Foreground下拉框包含Custom，System，以及Web分类的颜色，您可以从这里选择颜色并设置C1MainMenu控件的前景色。
Highlight (ForeHiColor)	Highlight下拉框包含下拉框包含Custom，System，以及Web分类的颜色，您可以从这里选择颜色并设置C1MainMenu控件的高亮项目的文本颜色。
Background	Background下拉框包含Custom，System，以及Web分类的颜色，您可以从这里选择颜色并设置C1MainMenu控件的背景色。
Highlight (BackHiColor)	Hightlight下拉框包含Custom，System，以及Web分类的颜色，您可以从这里选择颜色并设置C1MainMenu控件的高亮显示项目的背景色。
背景图像	
Image	Image下拉框打开Open对话框，您可以应用选中的背景图片至C1MainMenu控件。
Layout	布局下拉框打开布局项目列表（None，Tile，Center，Stretch，以及Zoom），您可以在此选择设置C1MainMenu的背景图片的布局方式。
Show in image bar	在图像栏显示复选框决定当工具栏的样式为Drop-downMenu时，是否在图像栏显示背景图。
Reset properties to default	选择重置属性为默认值选项将已经修改过的C1MainMenu的外观属性修改回其默认值。

编辑其他属性

单击编辑其他属性按钮打开C1MainMenuMiscellaneous
对话框，在此您可以编辑C1MainMenu控件的其他属性。



下表定义了包含在C1MainMenuMiscellaneous属性对话框中的项目：

项目	描述
行为	
Enabled	Enabled复选框指示是否选中的命令在运行时启用。默认情况下，C1Command.Enabled属性设置为True。
Visible	Visible复选框指示是否选定的命令将在运行时显示。默认情况下，C1Command.Visible属性设置为True。
Minimum Size	MinimumSize字段包含了一个Width以及一个Height的NumericUpDown控件，用来指定C1MainMenu控件最小宽度和高度。
Maximum Size	MaximumSize包含了一个Width以及一个Height的NumericUpDown控件，用来指定C1MainMenu控件最大宽度和高度。
Can Merge	CanMerge复选框指示是否将MDI子菜单与MDI父菜单合并。选择复选框启用此属性，取消选择该复选框禁用此属性。
Show tooltips	Showtooltips复选框指示是否当鼠标光标悬停在菜单项上方时显示工具提示文本。选择复选框启用此属性，取消选择该复选框禁用此属性。
Wrap	Wrap获取一个值（如果选中则为True；否则为False）表示当菜单项无法在单行显示下时，是否将菜单折行或者显示一个\"More...\"按钮。
Cursor	Cursor下拉框打开一个不同光标项目的列表（None，Tile，Center，Stretch，以及Zoom），您可以从中选择当光标悬停在C1MainMenu控件上方时显示的光标形状。
Use wait cursor	Usewaitcursor复选框指示是否使用表示等待状态的光标。
设计	
Generate member	Generatemember复选框指示是否为C1MainMenu控件的各成员生成代码。（如果选中则为True；否则为False）
Locked	Locked复选框指示是否C1MainMenu为锁定状态。（如果选中则为True；否则为False）
Show invisible items	Showinvisibleitems复选框指示是否显示在C1MainMenu控件中不可见的项目。

Bottom margin	Bottommargin获取底部的页边距。
Reset properties to default	选择重置属性为默认值项目将重置已经修改的C1MainMenu的属性回复为其默认值。

C1CommandLink工具栏

C1CommandLink或者C1Command工具栏显示在C1MainMenu以及 C1ToolBar中的C1CommandLinks/C1Command上。

显示C1CommandLink工具栏

选择C1ToolBar或者C1MainMenucontrol中的C1Command项目以显示C1CommandLink工具栏。C1CommandLink工具栏将出现在菜单项或者工具栏项的下方，如下图所示：

C1CommandLink 工具栏按钮

C1CommandLink工具栏包含以下命令按钮：

工具栏按钮	描述
	**插入命令链接：在选中的命令之前添加一个新的命令。**
	**删除命令链接：删除选定的命令。**
	**编辑命令链接属性：应用样式，布局，以及标签布局至C1DockingTab控件。**
	**编辑命令属性：设置文本和字体样式并应用图像至命令。**
	**改变链接的命令：打开LinktoCommand设计器，在此您可以在link to existing command列表框中改变到命令的链接。**

插入命令链接

单击Insert command link按钮在选中的命令之前插入一个新的命令。

删除命令链接

单击Deletecommandlink按钮删除选中的命令。

CommandLink 属性

单击CommandLink属性按钮打开CommandLink属性对话框，在此您可以应用设置到选中的命令。

下表定义了包含在C1CommandLinkC1CommandLink属性对话框中的项目：

项目	描述
命令	
Command	Command下拉列表框显示属于选中的C1CommandMenu的命令类型列表。
Text	Text文本框显示出现在选中命令上的文本名称。为重命名命令的文本名称，请选中文本框中的文本并键入期望的文本名称。
Tooltip	ToolTip文本框显示出现在选中命令的工具提示。如果没有提示的定义，则ToolTip文本框是空的。为修改或创建一个提示，请在ToolTip文本框中输入文本。
外观	
Button look	Button look下拉列表包含以下值供您选择：Default，Text，Image，以及TextAndImage以设置命令链接按钮的外观。
Owner draw	选择 Owner draw 复选框将启用C1CommandLink.OwnerDraw属性。
Default 项目	选择Default item复选框获取选中命令链接的默认外观。
行为	
Merge type	Mergetype下拉列表框包含以下合并类型的值供您选择：Add，Replace，MergeItems，以及Remove，以获取命令链接的合并类型。默认值是Add。
Merge order	Mergeorder NumericUpDown框获取一个值，表示当菜单项和另一个菜单项合并时，二者的相对位置。
Delimiter	选择Delimiter复选框在选中的命令链接之前绘制一条分隔符。
New column	选择Newcolumn复选框启用C1CommandLink.NewColumn属性。
Always recent	选择Alwaysrecent复选框启用C1CommandLink.AlwaysRecent属性。
Reset properties to default	选择重置属性为默认值项目将已经修改过的C1MainMenu的外观属性修改回其默认值。

C1Command属性

单击C1Command属性按钮打开C1CommandMenu命令属性对话框，在此您可以应用设置到选中的命令。

下表定义了包含在C1CommandMenu命令属性对话框中的项目：

字段	描述
General	
Text	Text文本框显示出现在选中命令上的文本名称。为重命名命令的文本名称，请选中Text文本框中的文本，并键入期望的文本名称。
Show text As ToolTip	Show text as Tooltip复选框显示当复选框选中时，C1Command.Text 属性的值将用作工具提示。
Tooltip	工具提示文本框显示出现在选中命令上的工具提示。如果没有定义工具提示，则工具提示文本框为空。为修改或创建一个工具提示，可以在工具提示文本框中输入文本。注意，在此C1CommandLink.ToolTipText属性的优先级要高于

	C1Command.ToolTipText 属性。
Shortcut	Shortcut下拉框显示一个按键的列表，您可以从中选择关联到当前选中命令的键盘快捷方式。
Show shortcut	Showshortcut复选框指示选中的命令是否显示快捷方式。默认情况下，C1Command.ShowShortCut为启用状态。
Category	Category下拉列表框显示选中命令的分类。
C1ContextMenu	C1ContextMenu 下拉列表框显示多个C1ContextMenu的名称，因此您可以容易地为选中的命令关联到一个C1ContextMenu。
Image	
Icon	Icon下拉列表框打开一个Open对话框，您可以查找关联到选中命令的图标。
Image	Image下拉列表框显示当前关联到选定的命令的图像。单击下拉箭头，打开Open对话框，在此您可以查找到希望关联到选中命令的图像。
Image index	Imageindex下拉列表框中显示命令的图像索引值。
Behavior	
Visible	Visible复选框指示是否选定的命令将在运行时显示。默认情况下，C1Command.Visible属性设置为True。
Enabled	Enabled复选框指示是否选中的命令在运行时启用。默认情况下，C1Command.Enabled属性设置为True。
Pressed	Pressed复选框指示是否选中的命令为按下状态。默认情况下，C1Command.Pressed属性设置为True。
Show drop down arrow	Showdropdownarrow复选框指示当选定的命令位于工具栏上时，是否显示下拉箭头。
Checked	Checked复选框表示是否命令为复选状态。
Auto-toggle checked state	自动切换选中状态的复选框指示是否当该命令被调用时C1Command.Checked属性自动切换。
Reset properties to default	选择重置属性为默认值项目重置修改的C1Command属性修改回其默认值。

改变链接的命令

单击Changelinkedcommand按钮打开LinktoCommand设计器，在此您可以在link to existing command列表框中改变到命令的链接。

C1CommandMenu工具栏

C1CommandMenu工具栏将出现在C1MainMenu以及C1ToolBar控件的子菜单项目上。

显示 C1CommandMenu工具栏

为显示C1CommandMenu工具栏，选中位于C1ToolBar或者C1MainMenu控件中的C1CommandMenu项目以及其子菜单项目，并单击打开按钮 以打开浮动的C1CommandMenu工具栏，如下图所示：

C1CommandMenu 工具栏按钮

C1CommandMenu工具栏包含以下命令按钮：

工具栏按钮	描述
	**添加新的命令链接/命令：**添加一个空白的命令链接，将打开LinktoCommand对话框为此空白命令创建一个新的命令/命令链接。
	**编辑命令链接：**打开C1CommandLink集合编辑器，该编辑器允许您在Members列表框中编辑当前的命令链接。
	**编辑命令属性：**打开C1CommandMenu命令属性对话框，在此您可以修改选定的命令的属性。
	**编辑菜单的外观：**打开C1CommandMenu外观对话框，在此您可以修改选中命令的外观属性。
	**编辑副标题属性：**打开C1CommandMenuFormCaption属性对话框，在此您可以修改选中菜单以及其子菜单的副标题属性。

插入命令链接

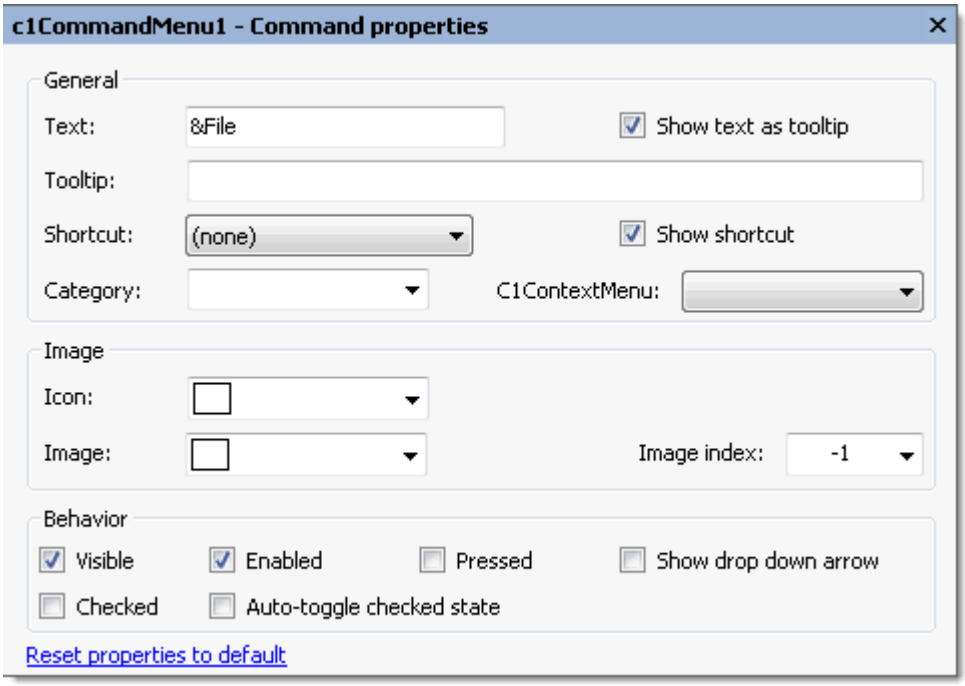
单击Addnewcommandlink/command按钮添加一个空白的命令链接，并打开LinktoCommand对话框，您可以为此空白命令创建一个新的命令/命令链接。

编辑命令链接

单击Editcommandlinks按钮打开C1CommandLink集合编辑器，该编辑器允许您在Members列表框中编辑当前的命令链接。

编辑命令属性

点击Editcommandproperties按钮打开C1CommandMenu命令属性对话框，在此您可以修改选定的命令的属性。



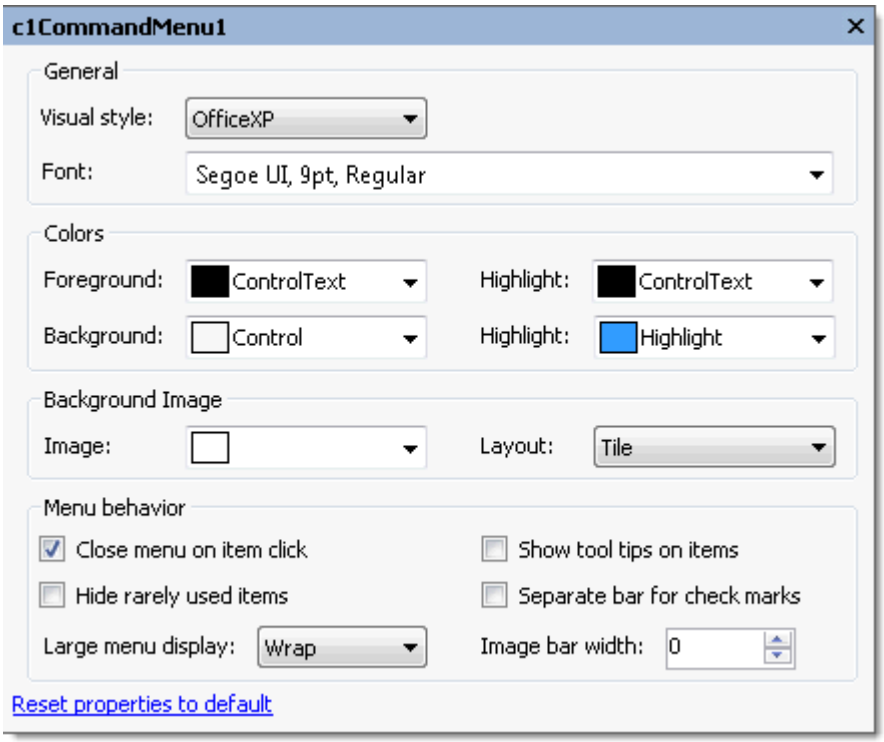
下表定义了包含在C1CommandMenuCommand属性对话框中的项目：

项目	描述
General	
Text	Text文本框显示出现在选中命令上的文本名称。为了重命名命令的文本名称，选中Text文本框中的文本，并键入所期望的文本名称。
Show text as tooltip	将文本显示为工具提示复选框指示当复选框选中时，将C1Command.Text属性的值作为工具提示的内容。
Tooltip	ToolTip文本框显示出现在选中命令的工具提示。如果没有定义ToolTip，则ToolTip文本框为空。为修改或创建一个提示，请在ToolTip文本框中输入文本。
Shortcut	Shortcut下拉框显示一个按键列表，您可以在此选择关联到选中命令的键盘快方式。
Show shortcut	Showshortcut复选框指示是否选中的命令显示快捷方式。 默认情况下C1Command.ShowShortCut 是启用状态。
Category	Category下拉列表框显示选中命令的分类。
C1ContextMenu	下拉列表框显示多个C1ContextMenu的名称，因此您可以容易地位选中的命令关联到一个C1ContextMenu。
Image	
Icon	Icon下拉列表框打开一个Open对话框，您可以查找关联到选中命令的图标。
Image	Image下拉列表框显示当前关联到选定的命令的图像。点击下拉箭头，打开Open对话框，在此您可以查找希望关联到选中命令的图片。
Image Index	Imageindex下拉列表框中显示命令的图像索引值。
Behavior	
Visible	Visible复选框指示是否选定的命令在运行时显示。默认情况下，C1Command.Visible属性的值设置为True。

Enabled	Enabled复选框指示是否选定的命令将在运行时启用。默认情况下，C1Command.Enabled属性设置为True。
Pressed	Pressed复选框指示是否选中的命令为按下状态。默认情况下，C1Command.Pressed属性设置为True。
Show drop down arrow	显示下拉箭头复选框指示当选定的命令位于工具栏上时，是否显示下拉箭头。
Checked	Checked复选框表示是否命令为复选状态。
Auto-toggle checked state	自动切换选中状态的复选框指示是否当该命令被调用时C1Command.Checked属性自动切换。
Reset properties to default	选择重置属性为默认值项目将重置已经被修改的C1CommandMenu命令的属性为其默认值。

编辑菜单的外观

单击Editmenuappearance按钮打开C1CommandMenu外观对话框，在此您可以修改选中命令的外观属性。



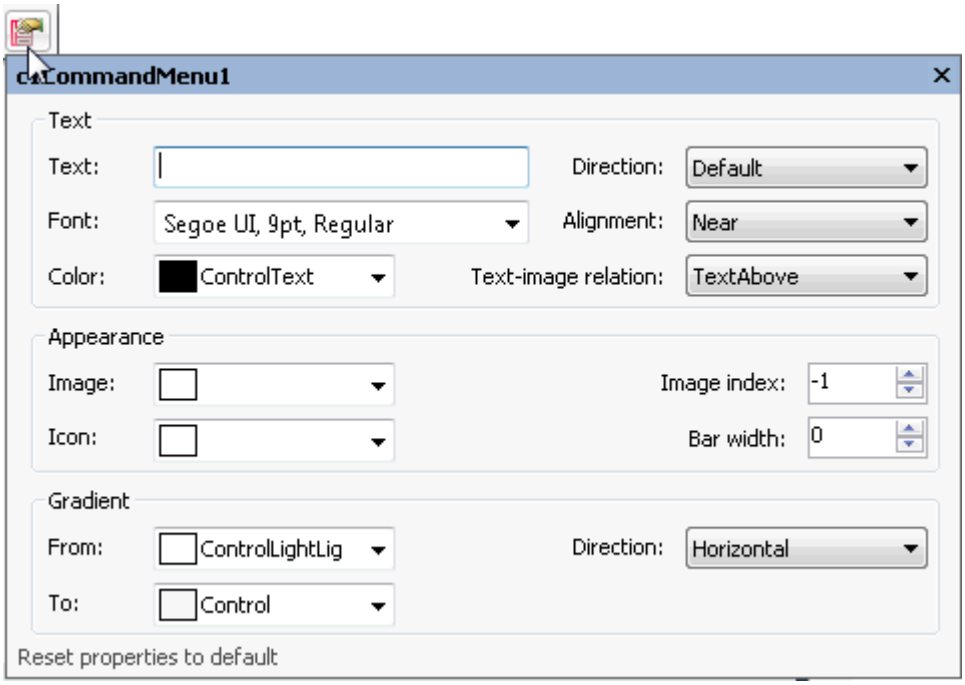
下表定义了包含在C1CommandMenu外观属性对话框中的项目：

项目	描述
General	
Visual style	Visualstyle下拉框包含以下项目，您可以从中选择以改变菜单控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP。
Font	Font下拉框打开字体对话框，您可以修改菜单字体样式属性。

Colors	
Foreground	Foreground下拉列表包含Custom，System，以及 Web 分类的颜色选项，您可以从中为菜单控件选择前景色。
Highlight (Foreground)	Highlight下拉列表包含Custom，System，以及 Web 分类的颜色选项，您可以从中为高亮显示的项目选择文本颜色。
Background	Background下拉列表包含Custom，System，以及 Web 分类的颜色选项，您可以从中为菜单选择底部的背景色。
Highlight (Background)	Highlight下拉列表包含Custom，System，以及 Web 分类的颜色选项，您可以从中为菜单中高亮显示的项目选择背景色。
Background Image	
Image	Image下拉框打开Open对话框，在此您可以选择命令所使用的背景图像。
Layout	Layout下拉框打开一个布局项目的列表（None，Tile，Center，Stretch，以及Zoom），您可以从中选择菜单的背景图片的布局方式。
Menu behavior	
Close menu on item click	当Close menu on item复选框被选中时，运行时将在单击子菜单项时将其关闭。
Show tooltips on itemss	当Show tooltips on items复选框被选中时，将在运行时在菜单项上显示工具提示。
Hide rarely used items	当Hide rarely used items复选框被选中时，将在运行时隐藏较少使用的菜单项。
Separate bar for check marks	当Separate bar for check marks复选框被选中时，将在一个分隔栏上显示复选标记而不是图片。
Large menu display	当Large menu display复选框被选中时，它将决定使用大菜单方式显示（当所有项目无法在一列显示时）。
Image bar width	Image bar width复选框被选中时，将获取或设置的图像/复选框栏的宽度。如果设置为0，则宽度为自动计算。
Reset properties to default	选择重置属性为默认值项重置修改过的C1CommandMenu外观属性为其默认值。

编辑副标题属性

单击Edit side caption properties按钮打开C1CommandMenu FormCaption属性对话框，在此您可以修改选中菜单以及其子菜单的副标题属性。



下表定义了包含在C1CommandMenu FormCaption对话框中的项目：

项目	描述
Text	
Text	Text文本框显示出现在选中的C1CommandMenu的副标题的文本名称。为重命名副标题的文本名称，请选中Text文本框中的文本并键入期望的文本名称。
Direction	Direction下拉框显示一个包含不同值（Default，Horizontal，VerticalLeft，VerticalRight）的列表框，您可以从中选择SideCaption中文本的方向。
Font	Font下拉框打开字体对话框，在此您可以修改SideCaption的字体样式属性。
Alignment	Alignment下拉框显示一个包含文本对齐选项（near，far，或者center）的列表框，您可以从中选择指定SideCaption的文本对齐方式。
Color	Color下拉框显示custom，web，以及system颜色，您可以从中选择设置SideCaption的文本颜色。
Text-image relation	Text-image relation下拉框显示一个包含不同值（TextAbove，TextBelow，TextOnLeft，TextOnRight）的列表框，您可以从中选择设置SideCaption的文本布局。
外观	
Image	Image获取或设置SideCaption的图像。
Image index	Image index获取或设置SideCaption所使用的位于C1CommandHolder.ImageList中间的图像索引值。
Icon	Icon下拉框打开Open对话框，从中您可以选择设置用于SideCaption的图标。
Bar width	Bar width获取或设置菜单中图像/复选框栏的宽度。如果设置为0，则宽度为自动计算。
Gradient	
From	From下拉框打开一个包含custom，web，以及system颜色分类的列表，您可以从中选择SideCaption渐变效果的起始颜色。

To	To下拉框打开一个包含custom，web，以及system颜色分类的列表，您可以从中选择SideCaption渐变效果的结束颜色。
Direction	Diection下拉框包含不同的渐变方向（horizontal，vertical，forward diagonal，或者backward diagonal），您可以从中选取SideCaption渐变效果的渐变类型。
Reset properties to default	选择重置属性为默认值项重置修改过的C1CommandMenu FormCaption属性的值为其默认值。

C1DockingTab工具栏

C1DockingTab工具栏以及C1DockingTabPage工具栏显示在C1DockingTab控件上。为显示C1DockingTab工具栏，请选中C1DockingTab控件并滑动您的光标至C1DockingTab控件上。为显示C1DockingTabPage工具栏，请滑动您的光标至C1DockingTab控件内部的C1DockingTabPage区域。

打开和关闭C1DockingTab工具栏

为打开C1DockingTab 工具栏，点击 按钮。为关闭C1DockingTab 工具栏，点击 按钮。

C1DockingTab工具栏由以下命令按钮组成：

工具栏按钮	描述
	**添加标签页：**添加标签页至C1DockingTab。
	**编辑标签页的集合：**通过C1DockingTabPage集合编辑器添加或删除标签页，同时也可以在此修改每一个标签页的属性。
	**编辑标签区域属性：**应用样式，布局，以及标签布局至C1DockingTab控件。
	**编辑停靠标签外观：**设置颜色和字体样式并应用图像至C1DockingTab控件。
	**编辑停靠标签行为：**应用行为设置至C1DockingTab控件。

添加新的页面

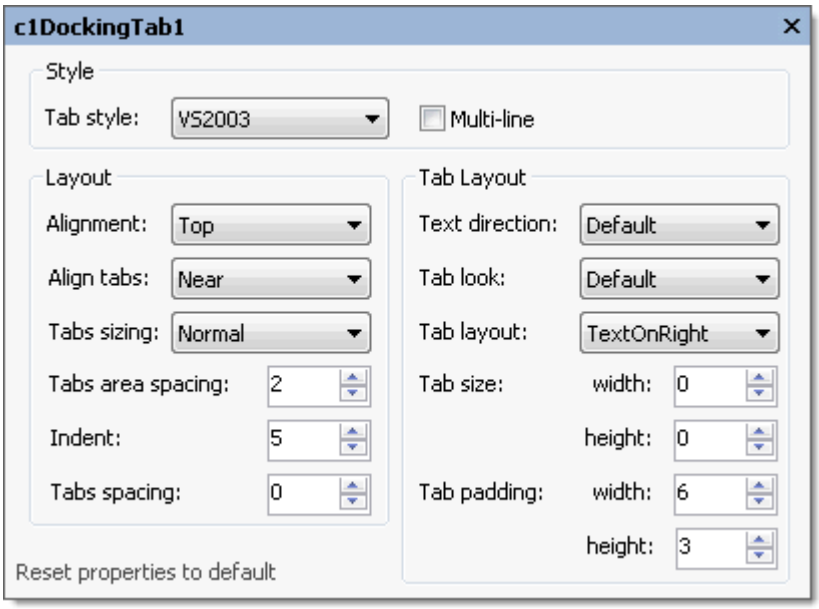
点击添加标签页按钮在现有的标签之后添加一个新的标签。

编辑标签页的集合

单击编辑标签页集合按钮打开C1DockingTabPage集合编辑器，在此您可以修改每一个标签页的设置，同时也可以添加或者删除标签页。关于C1DockingTabPage集合编辑器的更多信息，请参见C1DocingTabPage集合编辑器。

编辑标签区域属性

单击编辑标签区域按钮打开标签区域对话框，在此您可以修改C1DockingTab控件中标签区域的设置。



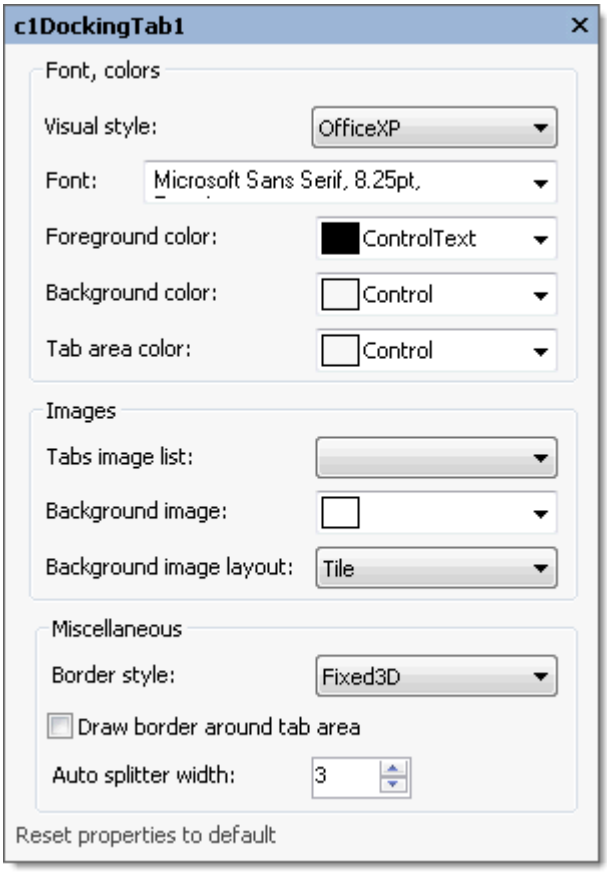
下表定义了包含在标签区域属性对话框中的项目：

字段	描述
Style	
Tab style	TabStyle下拉框显示一个标签样式选项（Default，WindowsXP，Classic，Sloping，Rounded，Office2003，以及Office2007）的列表，您可以从中选择C1DockingTab控件的标签样式。
MultiLine	多行复选框获取或设置一个值，（如果选中则为True，否则为False）指示是否可以显示多行标签。
Layout	
Alignment	Alignment下拉框中显示一个包含控件区域（顶部，底部，左侧或右侧）的列表框，您可以从中选取指定标签对齐的位置。
Align tabs	Alignment tabs下拉框包含以下选项（Near，Center，或者Far），您可以从中选择指定标签如何在页面内容区域的某一侧进行布局。
Tabs sizing	**Tabs sizing**下拉框包含以下项目（Normal，Fit，FillToEnd，以及User），您可以从中选择设置如何设置标签的大小。
Tabs area spacing	**Tabs area spacing**获取或设置标签区边缘和标签之间的间距。
Indent	**Indent**属性获取或设置第一个选项卡的缩进值。
Tabs spacing	**Tabs spacing**获取或设置标签之间的距离（距离可以为负值以便将标签页重叠）。
Tab Layout	
Text direction	Text direction项目获取或设置绘制在标签上文本的方向。获取或设置文本方向项目画在标签文本的方向。
Tab look	Tab look下拉框包含以下项目（Default，Text，Image，以及 TextAndImage），您可以从中选择设置标签的外观。
Tab layout	**Tab layout**下拉框包含以下项目（TextOnRight，TextOnLeft，TextBelow，TextAbove）

	您可以从中选择设置标签上文本和图像的相对布局。
Tab size	**Tab size**获取标签的宽度和高度。
Tab padding	**Tab padding**获取标签内边距的宽度和高度。
Reset properties to default	选择**Reset properties to default**项目重置修改的C1DockingTab属性为其默认值。

编辑停靠标签外观

点击 Edit docking tab appearance按钮打开外观对话框，在此您可以修改C1DockingTab的外观属性。



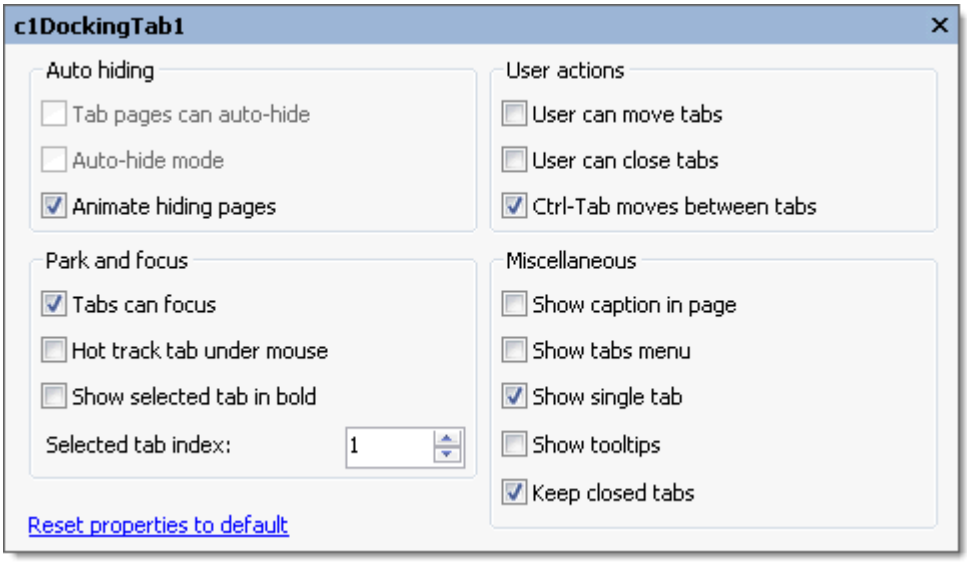
下表定义了包含在外观属性对话框中的项目：

字段	描述
Font, colors	
Visual style	**Visual style**下拉框包含以下项目，您可以从中选择改变C1DockingTab控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP。
Font	**Font**下拉框打开字体对话框，在此您可以修改C1DockingTab控件的字体样式属性。
Foreground color	**Foreground**下拉列表框包含Custom，System，以及 Web 分类的颜色，您可以从中选取颜色用于设置C1DockingTab控件的前景色。

Background color	**Background**下拉列表框包含Custom，System，以及 Web 分类的颜色，您可以从中选取颜色用于设置C1DockingTab控件的背景色。
Tab area color	**Tab area color**下拉列表框包含Custom，System，以及 Web 分类的颜色，您可以从中选取颜色用于设置C1DockingTab控件标签区域的颜色。
Images	
Tabs image list	管理C1DockingTab控件中标签的图像集合。
Background image	**Background image**的下拉框将打开一个对话框，在此您可以应用背景图片到C1DockingTab控件。
Background image layout	**Background image layout**下拉框打开一个布局项目（None，Tile，Center，Stretch，以及 Zoom）的列表，您可以从中选择C1DockingTab控件的背景图片布局方式。
Miscellaneous	
Border style	**Border style**下拉框中显示一个列表框包含不同的边框样式（None，FixedSingle，Fixed3D），您可以从中为C1DockingTab控件指定边框样式。
Draw border around tab area	在**Draw border around tab area**复选框指示是否在标签区域绘制一个边框。（如果选中则为True；否则为False）
Auto splitter width	**Auto splitter width**获取或设置当启用页面停靠时控件页面之间的自动分隔线的宽度。
Reset properties to default	选择重置属性为默认值项重置修改过的C1DockingTab属性为其默认值。

编辑停靠标签行为

单击编辑停靠标签行为按钮打开Behavior对话框，在此您可以为C1DockingTab启用特定的行为。



下表定义了包括在Behavior属性对话框中的项目：

项目	描述
Auto hiding	
Tab pages can auto-hide	标签页可以自动隐藏复选框指示该页面可以自动隐藏。（如果选中则为True；否则为False）
Auto-hide mode	**Auto-hide mode**复选框指示该C1DockingTab 是自动隐藏模式。（如果选中则为True；否则为False）
Animate hiding pages	**Animate hiding pages**复选框指示动画隐藏停靠的标签页。（如果选中则为True；否则为False）
Park and focus	
Tabs can focus	**Tabs can focus**复选框指示标签可以在鼠标单击时接收焦点。（如果选中则为True；否则为False）
Hot track tab under mouse	**Hot track tab under mouse**复选框指示是否控件的标签页当鼠标悬停经过时改变其外观。（如果选中则为True；否则为False）
Show selected tab in bold	**Show selected tab in bold**复选框指示是否在选中的标签上显示文本为粗体。（如果选中则为True；否则为False）
Selected tab index	**Selected tab index**获取或设置当前选定页面的索引。
User actions	
User can move tabs	**User can move tab**复选框指示是否最终用户可以在运行时通过拖拽重新排布标签。（如果选中则为True；否则为False）
User can close tabs	**User can close tabs**复选框指示是否用户可以单独关闭每一个标签页。如果C1DockingTab.CanCloseTabs属性的值为True，则在标题区域显示一个关闭图标（如果此时ShowCaption属性为True），或者在标签区域显示此关闭图标。（如果选中则为True；否则为False）
Ctrl-Tab moves between tabs	**Ctrl-Tab moves between tabs**复选框指示是否C1DockingTab控件处理 Ctrl-Tab 以及 Ctrl-Shift-Tab 按键。（如果选中则为True；否则为False）
Miscellaneous	
Show caption in a page	**Show caption in a page**复选框指示是否在页面上显示标题。（如果选中则为True；否则为False）
Show tabs menu	**Show tabs menu**复选框指示是否显示一个按钮，单击显示全部的标签列表。（如果选中则为True；否则为False）。在多行模式下，此属性被忽略。

Show single tab	**Show single tab**复选框指示是否在控件上仅有一个页面时，显示单个的标签。（如果选中则为True；否则为False）
Show tooltips	显示工具提示复选框指示是否当鼠标悬停经过时，显示工具提示。（如果选中则为True；否则为False）
Keep closed tabs	**Keep closed tabs**复选框指示是否保持关闭的标签。（如果选中则为True；否则为False）
Reset properties to default	选择重置属性为默认值项目重置修改的C1DockingTab属性为其默认值。

C1DockingTabPage 工具栏

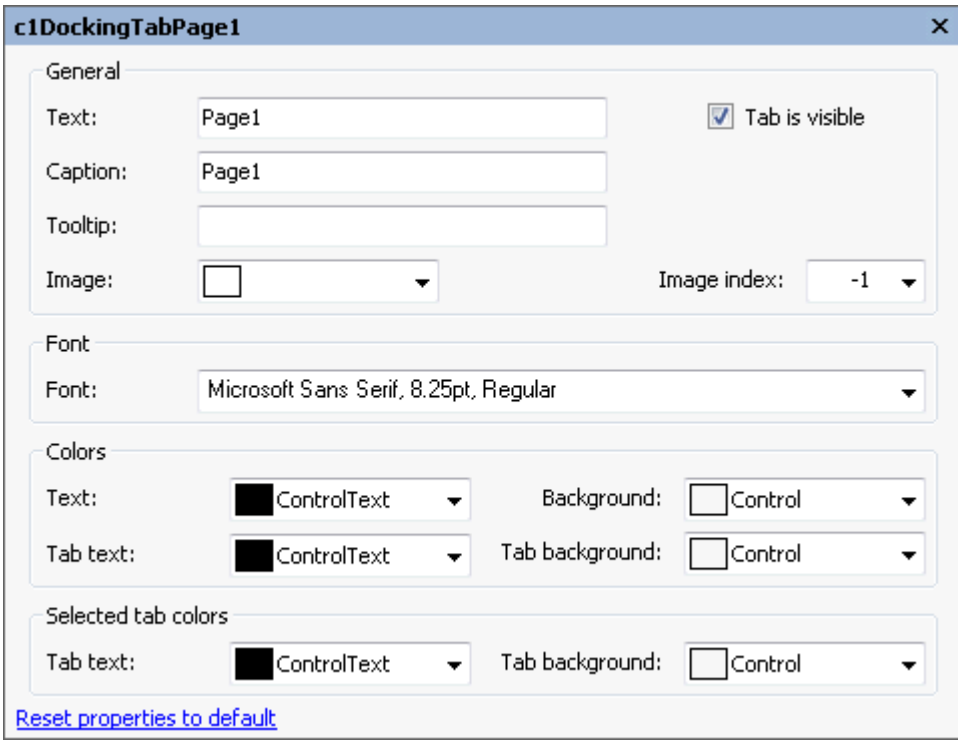
C1DockingTabPage工具栏将出现在C1DockingTabPage控件上。为了显示C1DockingTabPage工具栏，选中C1DockingTab控件并滑动光标至C1DockingTab控件的C1DockingTabPage区域内。

C1DockingTabPage工具栏由一个命令按钮组成：

工具栏按钮	描述
	**Edit tab page** 属性: 设置标签页的颜色和字体样式。

编辑标签页属性

单击编辑标签页属性按钮打开外观对话框，在此您可以修改C1DockingTabPages外观属性。



下表定义了包含在Tab Page 属性对话框的项目：

项目	描述
General	
Text	Text文本框显示出现在选中标签页的文本名称。为重命名标签页的文本名称，选中Text文本框的文本并键入期望的文本名称。
Caption	标题文本文本框显示出现在caption页面的文本名称。为重命名标签页的文本名称，选中Text文本框中的文本并键入期望的文本名称。
Tooltip	工具提示文本框设置标签页的工具提示文本。
Image	Image下拉列表框显示关联到选中的标签页当前的图像。单击下拉箭头打开Open对话框，在此您可以查找希望关联到选中标签页的图像。
Image index	Image index下拉列表框显示选中标签页图像的索引值。
Font	
Font	字体下拉框打开字体对话框，在此您可以修改C1DockingTab控件选中页面的字体样式。
Colors	
Text	Text下拉列表中包含Custom，System，以及Web分类的颜色，您可以从中选择标签以及标签页的文本颜色。
Background	Background下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择标签以及标签页的文本颜色
Tab background	标签背景下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置标签的背景色。
选中标签颜色	
Text	Text下拉列表中包含Custom，System，以及Web分类的颜色，您可以从中选择选中标签的文本颜色。
Background	背景下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择颜色设置选中的标签的背景色。
Reset properties to default	选择重置属性为默认值项重置修改过的C1DockingTabPage属性为其默认值。

C1NavBar工具栏

C1NavBar工具栏将出现在C1NavBar控件上。为显示C1NavBar工具栏，选中C1NavBar控件并将光标滑动到C1NavBar控件上。

打开和关闭C1NavBar工具栏

为打开C1NavBar 工具栏，点击 按钮。为关闭C1NavBar 工具栏，点击 按钮。

C1NavBar工具栏包含以下命令按钮：

工具栏按钮	描述

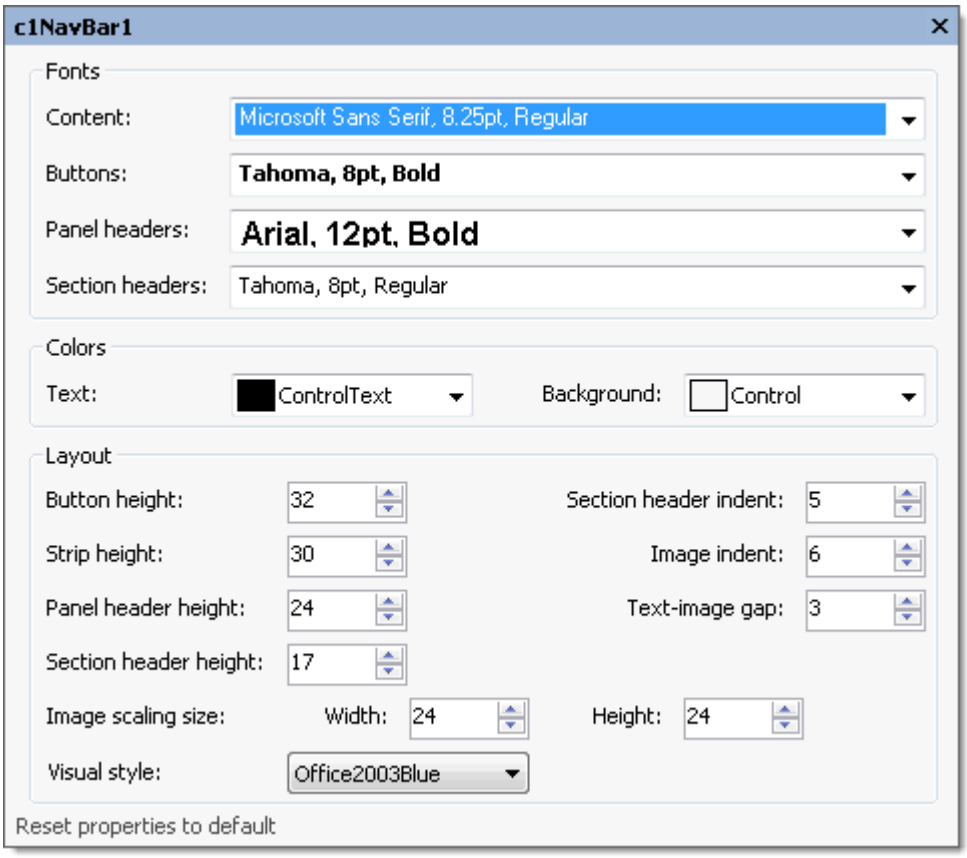
	**添加按钮以及相应的面板**：将按钮添加到C1NavBar。
	**编辑导航栏的外观和布局**：设置颜色和字体样式并应用图像至C1NavBar控件。
	**编辑导航栏杂项属性**：打开C1NavBar控件的C1NavBar对话框，在此您可以应用行为属性至C1NavBar控件。

添加按钮以及相应的面板

单击添加按钮以及相应面板按钮将在现有标签页之后添加一个标签。

编辑导航栏的外观和布局

单击"编辑导航栏的外观和布局"按钮将打开C1NavBar 属性对话框，在此您可以修改C1NavBar的外观属性。



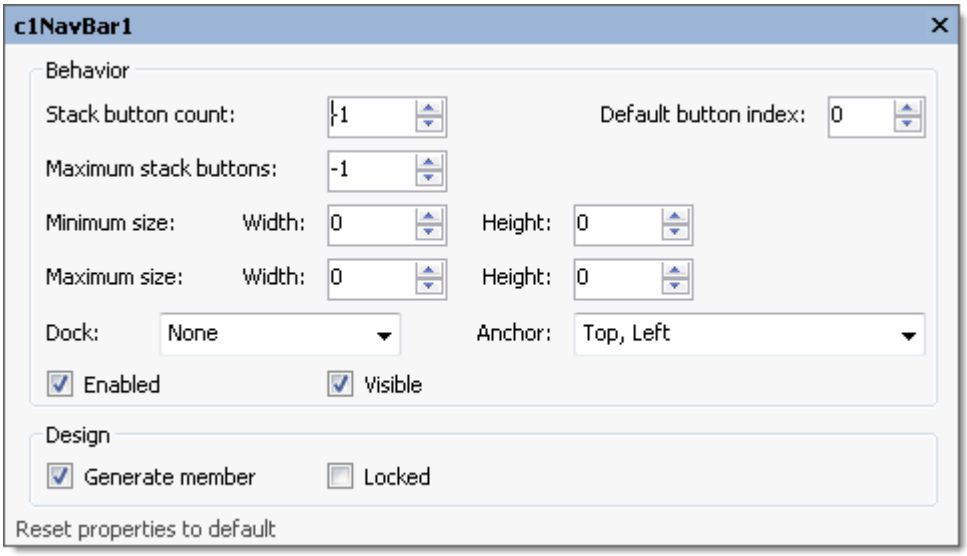
下表定义了包含在C1NavBar属性对话框中的项目：

项目	描述
字体	
Content	Content下拉框打开Font对话框，在此您可以修改C1NavBar控件内容的字体样式信息。
Buttons	Buttons下拉框打开字体对话框，在此您可以修改C1NavBar控件中按钮的字体样式属性。
Panel headers	Panel headers下拉框打开字体对话框，在此您可以修改C1NavBar控件面板标题的字体样式属性。

Section headers	Section headers下拉框打开字体对话框，在此您可以修改C1NavBar控件区域标题的字体样式属性。
Colors	
Text	Text下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置菜单控件的前景色。
Background	Background下拉列表框中包含Custom，System，以及Web分类的颜色，您可以从中选择设置C1NavBar控件的背景色。
Layout	
ButtonHeight	ButtonHeight设置一个整型值，表示按钮的高度。
Section header indent	Section header indent 决定了区域标题文本距离起始位置的间距。
StripHeight	StripHeight设置一个整型值，表示按钮的Strip高度。
ImageIndent	ImageIndent下拉框设置一个整型值，表示图像的缩进值。
PanelHeaderHeight	PanelHeaderHeight设置一个整型值，表示面板标题的高度。
Text-image gap	Text-image gap获取在一个堆叠按钮上，文本和图像之间的间距。
SectionHeaderHeight	SectionHeaderHeight设置一个整型值，表示区域标题的高度。
Image scaling size	Image scaling size设置显示在堆叠按钮上图像的尺寸大小，单位是像素。默认值是24 x 24像素。为设置图像的宽度，请使用表示宽度的NumericUpDown框，为设置图像的高度，请使用表示高度的NumericUpDown框。
Visual style	视觉样式下拉框包含以下的项目，您可以从中进行选择，改变C1NavBar控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP。
Reset properties to default	选择重置属性为默认值选项重置修改过的C1NavBar属性为其默认值。

编辑导航栏杂项属性

单击编辑导航栏杂项属性按钮打开C1NavBar对话框，在此您可以修改C1NavBar的各种其他属性。



下表定义了包含在C1NavBar属性对话框中的项目：

项目	描述
行为	
StackButtonCount	StackButtonCount获取或设置堆叠显示的按钮的数量。
DefaultButtonIndex	DefaultButtonIndex指定当窗体打开时，默认选中的按钮的索引。
Maximum stack buttons	Maximum stack buttons获取或设置堆叠的按钮的最大数量。如果设置为-1则表示最大数量是无限制的。
Minimum size	最小尺寸字段包括指定C1NavBar控件最小宽度和高度的，表示Width和Height属性的NumerUpDown控件组。
Maximum size	最大尺寸字段包括指定C1NavBar控件最大宽度和高度的，表示Width和Height属性的NumerUpDown控件组。
Dock	Dock下拉框包含一些项目，您可以从中选择定义C1NavBar控件停靠到容器中的位置。
Anchor	Anchor下拉框定义了C1NavBar控件可以绑定的容器边缘。当其锚定到某一个边缘时，C1NavBar和指定边缘最接近的边之间的间距将保持不变。
Enabled	Enabled复选框指示是否在运行时启用该C1NavBar控件。
Visible	Visible复选框指示是否该C1NavBar在运行时显示。
Design	
Generate member	Generate member复选框指示是否为C1NavBar控件的成员生成代码。（如果选中则为True；否则为False）
Locked	Locked复选框指示是否C1NavBar为锁定状态。（如果选中则为True；否则为False）
Reset properties to default	选择重置属性为默认值项目重置修改过的C1NavBar其他属性为其默认值。

C1OutBar 工具栏

C1OutBar工具栏出现在C1OutBar控件上。为显示C1OutBar工具栏，选中C1OutBar控件并移动光标至C1OutBar控件上。

打开和关闭C1OutBar工具栏

打开C1OutBar 工具栏，点击 按钮。关闭C1OutBar 工具栏，点击 按钮。

C1OutBar工具栏包含以下命令按钮：

工具栏按钮	描述
	**添加空白页：**添加一个带有工具栏的页面。
	**添加空页：**向C1OutBar添加不带工具栏的空白页。
	**编辑outbar的外观和布局：**设置颜色和字体样式并应用图像至C1OutBar控件
	**编辑outbar杂项属性：**打开C1OutBar控件的Miscellaneous对话框，在此您可以应用各种其他属性至C1OutBar控件。

添加空白页

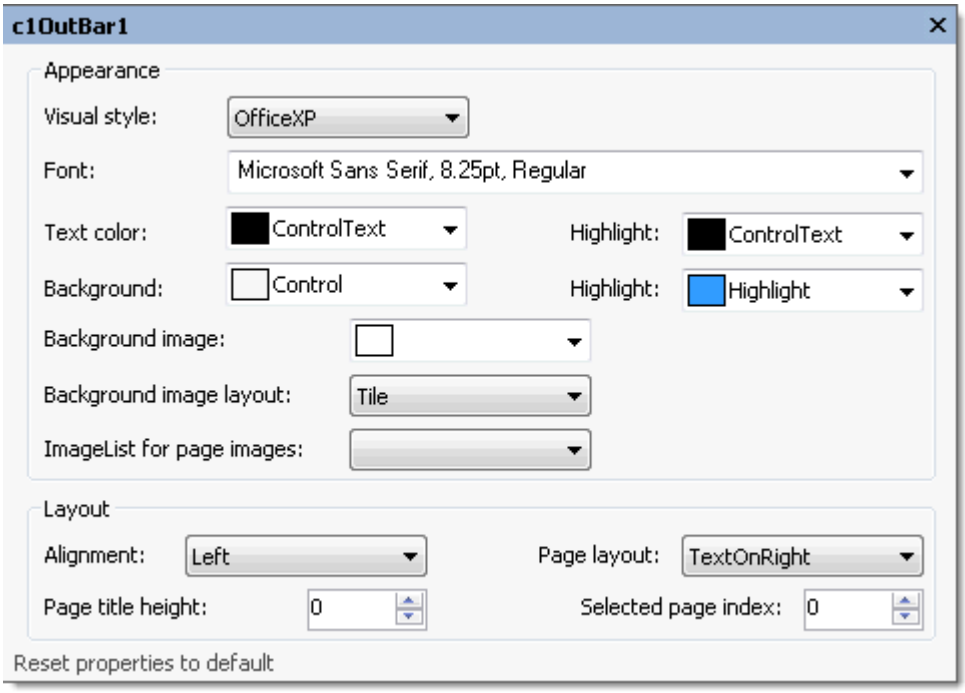
单击Add blank page按钮添加一个新的带有工具栏的页面至C1OutBar控件。

添加空页面

单击Add empty page按钮添加一个空页面至C1OutBar控件。

编辑outbar的外观和布局

单击编辑outbar外观和布局按钮打开属性对话框，在此您可以修改C1OutBar控件的各种属性。



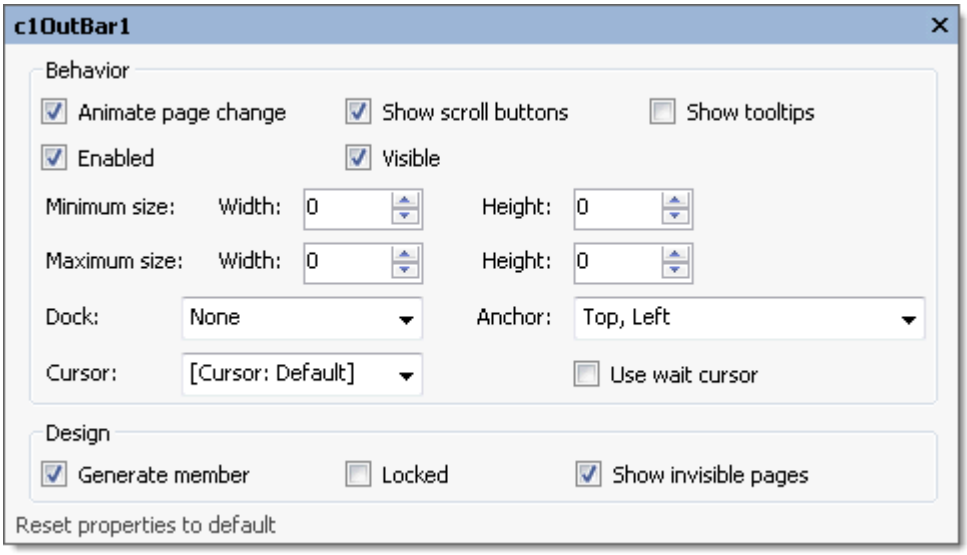
下表定义了包含在C1OutBar属性对话框中的项目：

项目	描述
Appearance	
Visual style	视觉样式下拉框包含以下项目，您可以从中选择一个改变C1OutBar控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP， Classic，以及WindowsXP。
Font	Font下拉框打开字体对话框，在此您可以修改C1OutBar控件的字体样式属性。
Text color	Forground下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置C1OutBar控件的文本颜色。
Highlight (ForeHiColor)	Highlight下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置C1OutBar控件高亮显示项目的前景色。
Background	Background下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置C1OutBar控件的背景色。
Highlight (BackHiColor)	Highlight下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选择设置C1OutBar控件突出显示项目的背景色。
Background image	Background image下拉框打开一个Open对话框，在此您可以选择应用到C1OutBar控件的背景图。
Background image layout	Layout下拉框打开布局项目的一个列表（None，Tile，Center，Stretch，以及 Zoom），您可以从中选取一个值，用于设置C1OutBar控件的背景图片的布局方式。
ImageList for page images	页面图像的ImageList下拉框获取用来提供显示在页面标题栏上图像的ImageList。
布局	
Alignment	Alignment下拉框获取页面标题栏区域文本和图像的对齐方式（Left，Right，Center）。

Page layout	页面布局下拉框包含一组项目，您可以从中选择设置文本相对于页面标题图像的上方，下方，左侧或者右侧。C1OutBar.PageLayout属性的默认值为TextOnRight。
Page title height	页面标题高度NumericUpDown框设置每一个页面标题的高度。
Selected page index	选定页面的索引设置选定页面的索引。
Reset properties to default	选择重置属性为默认值选项重置修改过的C1OutBar的属性为其默认值。

编辑杂项属性

单击编辑杂项属性按钮打开Miscellaneous对话框，在此您可以修改C1OutBar的杂项属性。



下表定义了包含在C1OutBar Miscellaneous对话框中的项目：

项目	描述
Behavior	
Animate page change	动画显示页面切换获取或设置一个值（如果选中则为True；否则为False），该值表示是否在改变选中的页面时显示动画。
Show scroll buttons	显示滚动按钮获取或设置一个值（如果选中则为True；否则为False），该值表示是否滚动工具栏的滚动按钮可见。
Show Tooltips	显示工具提示复选框指示是否当鼠标光标悬停在页面项上方时，显示工具提示。选择复选框启用此属性，取消选择该复选框禁用此属性。
Minimum size	最小尺寸的字段包括一个指定C1OutBar控件的最小宽度和高度尺寸的宽度和高度的NumericUpDown控件。
Maximum size	最大尺寸的字段包括一个指定C1OutBar控件的最大宽度和高度尺寸的宽度和高度的NumericUpDown控件。
Dock	Dock下拉框包含一些选项，您可以从中选择定义C1OutBar控件的哪一边绑定到容器边界。

Cursor	Cursor下拉框打开一个不同光标项目的列表，在此您可以设置当鼠标指针移动到C1OutBar控件上方时，显示的光标类型。
Use Wait cursor	Use wait cursor复选框指示是否使用表示等待状态的光标。
Anchor	Anchor下拉框定义了C1OutBar控件绑定的容器边缘。当锚定到某一边时，C1OutBar和指定的边缘最近的边之间的间距保持恒定。
Design	
Generate member	Generate member复选框指示是否为C1OutBar控件的成员生成代码。（如果选中则为True；否则为False）
Locked	Locked复选框指示C1OutBar控件是否为锁定状态。（如果选中则为True；否则为False）
Reset properties to default	选择重置属性为默认值选项重置修改过的C1OutBar的其他属性为其默认值。

C1TopicBar 工具栏

C1TopicBar控件的智能设计器为topic bar，topic page，以及topic link提供了以下工具栏：

- C1TopicBar工具栏
- C1TopicPage工具栏
- C1TopicLink工具栏

TopicBar，C1TopicPage，以及C1TopicLink工具栏出现在C1TopicBar控件上。

C1TopicBar 工具栏

C1TopicBar工具栏出现在C1TopicBar控件上。为显示C1TopicBar工具栏，请选中C1TopicBar控件并将光标移动到C1TopicBar控件的任意位置。

打开和关闭C1TopicBar工具栏

打开C1TopicBar工具栏，点击  按钮。关闭C1TopicBar工具栏，点击按钮。

C1TopicBar工具栏包含以下命令按钮：

工具栏按钮	描述
	**添加topic page：**添加一个新的topic page。
	**编辑页面：**打开C1TopicPage集合编辑器，在此您可以添加，删除C1TopicPage，或者修改其属性设置。
	**编辑topicbar外观和布局：**打开C1TopicBar 外观对话框，在此您可以修改C1TopicBar控件的外观，行为以及布局样式。

	编辑杂项属性:打开C1TopicBar的Miscellaneous对话框，在此您可以应用各种杂项属性至C1TopicBar控件。

添加新的页面

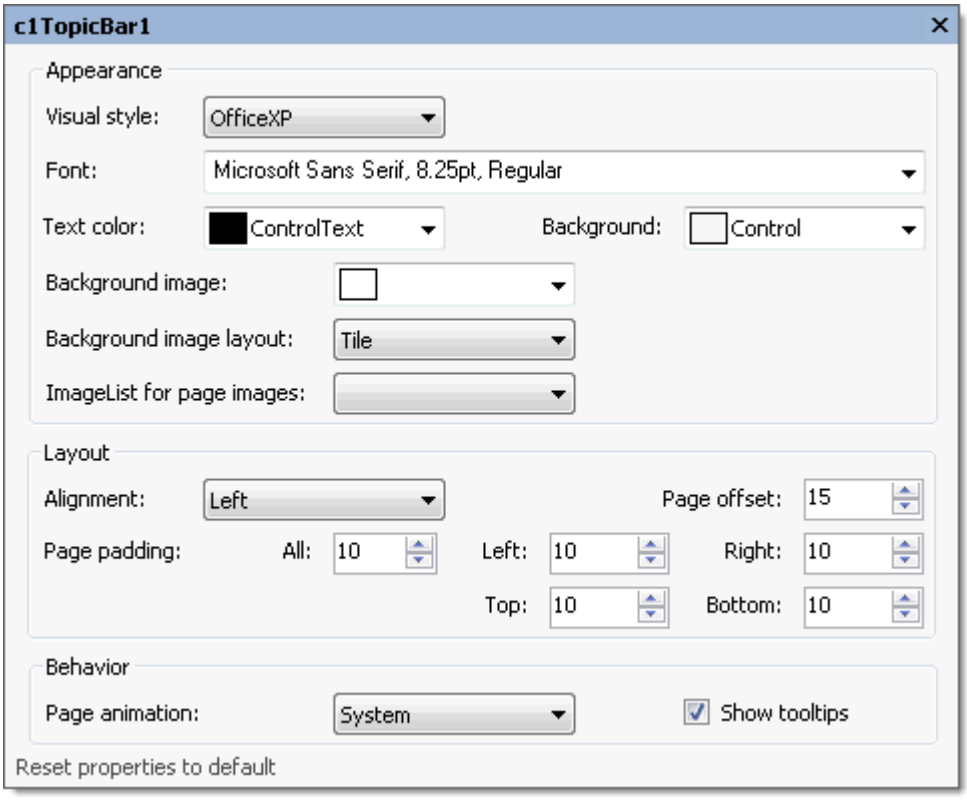
单击Add New Page按钮添加一个新的topic页面至当前已存在的topic page之后。

编辑页面

单击编辑页面按钮打开C1TopicPage集合编辑器，在此您可以添加或移除C1TopicPage，再或者修改其属性设置。

编辑topicbar外观和布局

单击编辑topicbar外观和布局打开C1TopicBar属性对话框，在此您可以修改C1TopicBar控件的外观，行为以及布局样式。



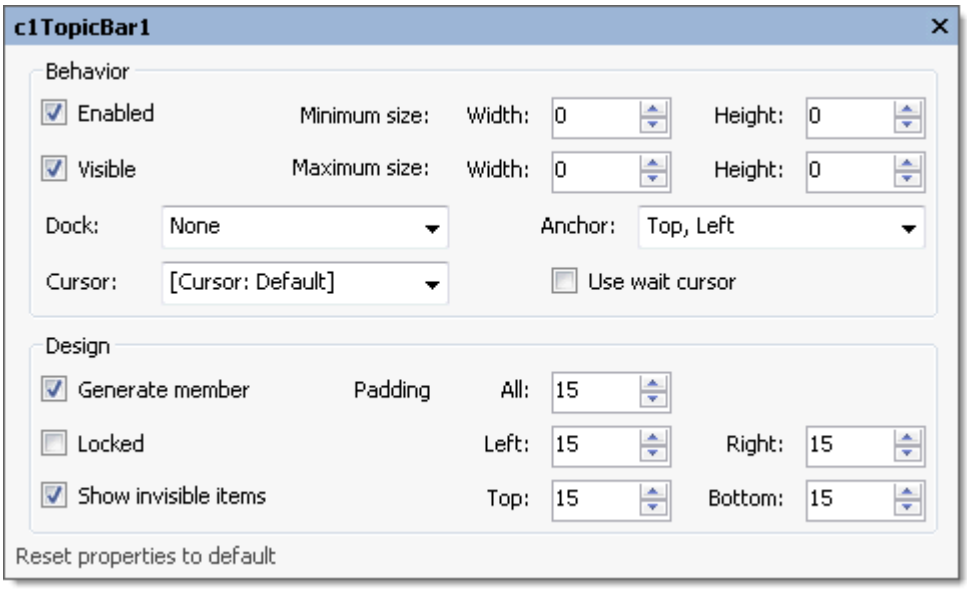
下表定义了包含在C1TopicBar属性对话框中的项目：

项目	描述
外观	
Visual style	视觉样式下拉框包含以下项目，您可以从中进行选择，以改变C1TopicBar控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP。
Font	Font下拉框打开字体对话框，在此您可以修改C1TopicBar控件的字体样式属性。

Text color	Foreground下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1TopicBar控件的前景色。
Background	Background下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1TopicBar控件高亮显示项目的背景色。
Background image	背景图像的下拉框将打开一个Open对话框，在此您可以选取图像用做C1TopicBar控件背景图。
Background image layout	背景图像布局下拉框打开一个包含布局项目（None，Tile，Center，Stretch，以及Zoom）的列表，在此您可以选择C1TopicBar控件的背景图片的布局方式。
ImageList for page images	页面图像的ImageList下拉框获取用来给页面标题栏提供图像的ImageList。
布局	
Alignment	Alignment下拉框包含Left，Right，以及Center选项，您可以用来设置topicbar页面标题的的对齐方式。
Page offset	页内偏移设置不同页面之间的间距，为一整型值。
Page padding	页边距设置页面边框和link之间的间距。
行为	
Page animation	页面动画下拉框包含所有可用的动画选项，您可以从中选择用作页面收起/展开时的动画效果。
ShowToolTips	ShowToolTips复选框指示是否当鼠标悬停在页面标题上方时，显示工具提示。
Reset properties to default	选择重置属性为默认值项目重置修改过的C1TopicBar 属性为其默认值。

编辑topicbar其他属性

单击Edit miscellaneous topicbar properties按钮打开C1TopicBar对话框，在此您可以修改C1TopicBar的杂项属性。



下表定义了包含在C1TopicBar对话框中的项目：

项目	描述
行为	
Enabled	Enabled复选框指示是否在运行时启用C1TopicBar。
Visible	Visible复选框指示是否在运行时C1TopicBar控件为可见。
Minimum size	最小尺寸字段包括一组表示Width和Height的NumericUpDown控件，用来指定C1TopicBar控件的最小宽度和高度的值。
Maximum size	最大尺寸字段包括一组表示Width和Height的NumericUpDown控件，用来指定C1TopicBar控件的最大宽度和高度的值。
Dock	Dock下拉框包含一些选项，您可以从中进行选择定义C1TopicBar控件的哪一边绑定到容器。
Cursor	Cursor下拉框打开一个包含不同的光标项目的列表，您可以从中选择设置当鼠标指针移动到C1TopicBar控件上时，显示的光标类型。
Use wait cursor	Use wait cursor复选框指示是否使用表示等待状态的光标。
Anchor	Anchor下拉框定义了C1TopicBar控件绑定到容器的哪些边。当其锚定到某一边时，C1TopicBar距离该边最近的边之间的间距保持不变。
Design	
Generate member	Generate member复选框指示是否为C1TopicBar的成员生成代码。（如果选中则为True；否则为False）
Locked	Locked复选框指示是否C1TopicBar为锁定状态。（如果选中则为True；否则为False）
Padding	Padding指定C1TopicBar控件的内部间距。
Reset properties to default	选择重置属性为默认值选项重置C1TopicBar修改过的属性为其默认值。

C1TopicPage工具栏

C1TopicPage工具栏将出现在C1TopicBar控件上。为显示C1TopicPage工具栏，请选中C1TopicBar控件，之后选中C1TopicBar中间的topic page，并移动鼠标光标至C1TopicPage的任意区域。

C1TopicPage工具栏包含一个命令按钮：

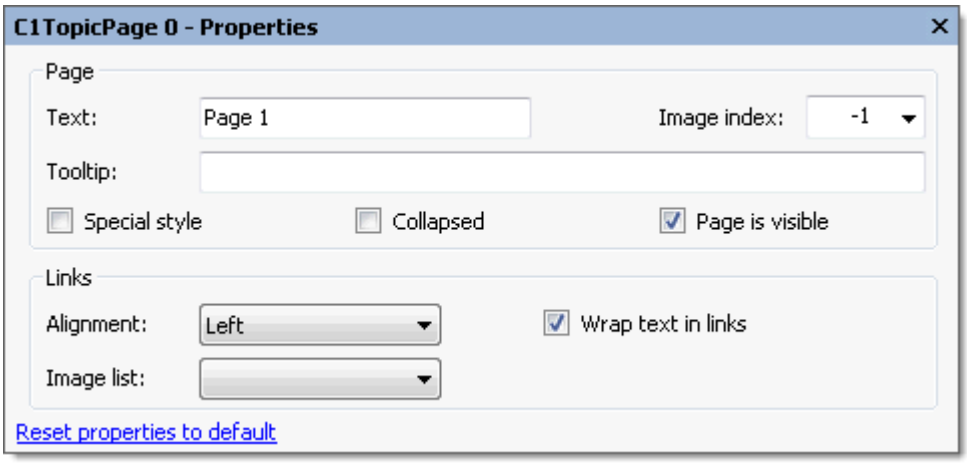
工具栏按钮	描述
	**添加主题链接：**在活动的主题页添加一个新的主题链接。
	**编辑主题页面的外观：**打开选中的C1TopicPage的属性对话框。
	**删除主题页：**从C1TopicBar中移除选中的C1TopicPage。

添加新的主题链接

单击Add topic link按钮向C1TopicBar控件中的主题页添加一个新的主题链接。

编辑主题页面的外观

单击编辑主题页面外观按钮打开选中的C1TopicPage的属性对话框。



下表定义了包含在在C1TopicBar属性对话框中的项目：

项目	描述
Page	
Text	Text文本框显示出现在标题上的文本名称。为了重命名标题的文本名称，请选中Text文本框中的文本并键入期望的文本名称。
Tooltip	工具提示文本框为选中的主题页设置工具提示文本。为添加一个工具提示，请在工具提示文本框中输入文本。
Image index	Image index下拉列表框中显示命令的图像索引值。
Special style	特殊样式复选框指示该页有一个特殊的深色标题。（如果选中则为True；否则为False）。
Collapsed	Collapsed复选框指示是否页面为收起状态。（如果选中则为True；否则为False）。
Page is visible	页面可见复选框指示该页是否可见。（如果选中则为True；否则为False）。
Links	
Alignment	Alignment下拉框包含Left，Right，以及Center对齐方式选项，以设置页面链接的对齐方式。
Wrap text in links	链接中文本折行复选框指示是否页面链接文本可以在其长度超出页面宽度时折行显示。（如果选中则为True；否则为False）。
Image list	图像列表设置用于提供页面链接显示的图像的图像列表。
Reset properties to default	选择重置属性为默认值项目将重置已经修改的C1TopicPage属性为其默认值。

删除主题页

单击删除主题页按钮从C1TopicBar移除选中的主题页。

C1ToolBar 工具栏

C1ToolBar工具栏将出现在C1ToolBar控件上。为了显示C1ToolBar工具栏，滑动您的鼠标光标至C1ToolBar控件的左上角区域。Open按钮将出现，用来打开C1ToolBar工具栏。

打开和关闭C1ToolBar工具栏

为打开C1ToolBar工具栏，点击 按钮。为关闭C1ToolBar工具栏，点击 按钮。

C1ToolBar工具栏包含以下命令按钮：

工具栏按钮	描述
	**添加新的命令链接/命令：**向工具栏当前的命令之后添加一个新的命令。
	**编辑命令链接：**打开C1CommandLink 集合编辑器来编辑命令链接。
	**编辑工具栏外观：**打开C1ToolBar 外观对话框，在此您可以设置C1ToolBar控件的通用外观属性。
	**编辑工具栏布局：**打开布局对话框，在此您可以为CToolbar控件设置布局属性。
	**编辑工具栏杂项属性：**打开C1ToolBar控件的Miscellaneous对话框，在此您可以应用行为相关的设置至C1ToolBar控件。

添加命令链接

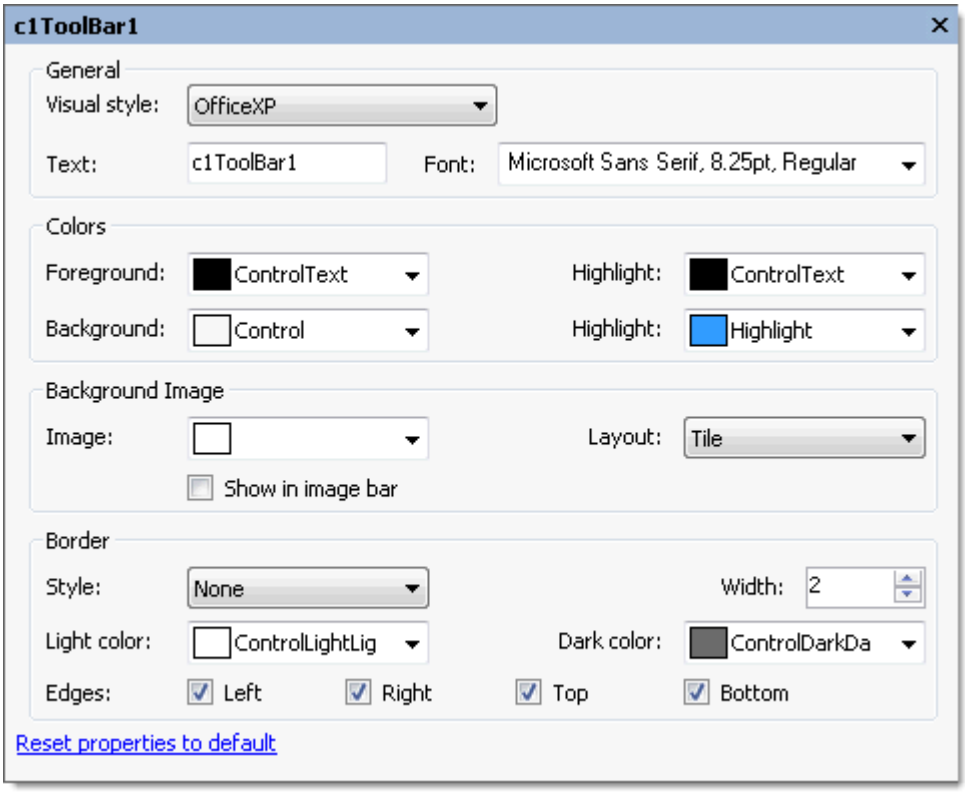
点击添加命令链接按钮在当前命令之后添加一个新的命令。它将在新的命令下方显示Link to Command设计器，因此您可以容易地编辑新的命令，而不需要离开当前设计界面。

编辑命令链接

点击编辑命令链接按钮打开C1CommandLink 集合编辑器，在此您可以添加或删除命令链接并编辑commandlink的属性。

编辑工具栏的外观

点击编辑工具栏外观按钮打开工具栏外观对话框，在此您可以修改C1ToolBar控件的外观属性。



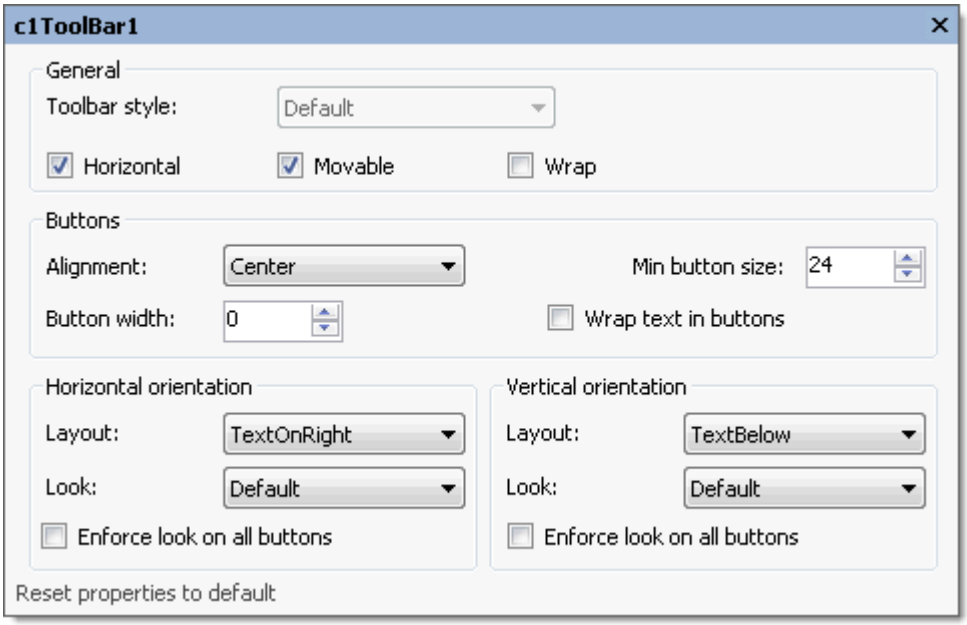
下表定义了包含在C1ToolBar控件外观对话框中的项目：

项目	描述
General	
Visual style	视觉样式下拉框包含以下选项，您可以从中选择以改变C1ToolBar控件的样式：Custom，System，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP。
Text	Text文本框显示出现在选中的工具栏按钮上的文本名称。为重命名工具栏按钮的文本，请选中Text文本框中的文本，并键入期望的文本名称。
Font	字体下拉框打开字体对话框，在此您可以修改C1ToolBar控件的字体样式属性。
颜色	
Foreground	Forground下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1Toolbar控件的前景色。
Highlight (ForeHiColor)	Highlight下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置设置C1ToolBar高亮显示的按钮的前景色。
Background	Background下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1ToolBar控件高亮显示按钮的背景色。
Highlight (BackHiColor)	Background下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1TopicBar控件中高亮显示项目的背景色。
背景图像	

Image	图像下拉框打开Open对话框，在此您可以选取应用为C1ToolBar控件背景图片的图像。
Layout	布局下拉框打开一个布局选项（None，Tile，Center，Stretch，以及Zoom）的列表，您可以从中选择设置C1ToolBar控件的背景图像布局。
Show in image bar	在图像栏中显示复选框指示当工具栏的样式为Drop-downMenu时，是否在图像栏显示背景图像。（如果选中则为True；否则为False）
边框	
Style	样式下拉框包含边框样式的列表，您可以从中选择设置C1ToolBar的边框样式。
Width	Width NumericUpDown控件设置C1ToolBar周围边框的宽度。
Light color	Light color下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1ToolBar控件周围边框的亮色颜色。
Dark color	Dark color下拉列表框包含Custom，System，以及Web分类的颜色，您可以从中选取颜色设置C1ToolBar控件周围边框的暗色颜色。
Edges	Edges复选框（Left，Right，Top，以及Bottom）允许您指定将此边框样式应用到哪一边。
Reset properties to default	选择重置属性为默认值选项重置修改过的C1ToolBar的属性为其默认值。

编辑工具栏布局

点击编辑工具栏布局按钮打开"布局"对话框，在此您可以修改C1ToolBar控件的布局属性。



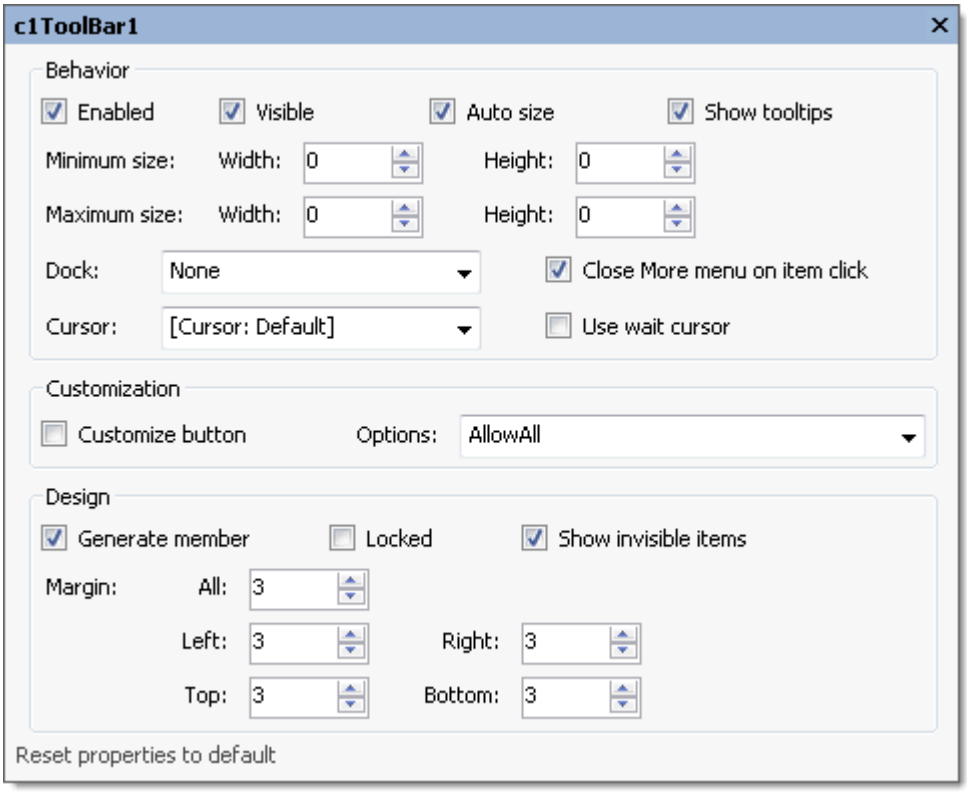
下表定义了包含在C1ToolBar控件布局对话框中的项目：

项目	描述

General	
Toolbar style	工具栏样式下拉框包含ToolBarStyle 属性的两个选项（Default以及Drop-downMenu），您可以从中选取设置工具栏的样式。
Horizontal	水平复选框指示设置C1ToolBar的方向为水平方向。（如果选中则为True；否则为False）。
Movable	可移动复选框指示是否用户可以移动工具栏。（如果选中则为True；否则为False）
Wrap	Wrap复选框指示当一行无法显示下全部的项目时是对工具栏进行折行还是显示一个\"More...\" 按钮。（如果选中则为True；否则为False）
按钮	
Alignment	Alignment下拉框获取垂直工具栏按钮的对齐方式，可用的选项包括Left，Right，或者Center。
Minimum button size	最小按钮尺寸获取或设置工具栏上按钮的最小尺寸（宽度和高度）。
Button width	按钮宽度获取或设置所有按钮的宽度（仅应用到水平工具栏；如果设置为0，则每一个按钮单独指定尺寸）。
Wrap text in buttons	按钮中文本折行复选框获取或设置一个值，该值指定是否当C1ToolBar.ButtonWidth的值大于零，且宽度无法放下全部文本时，链接的文本是否折行显示。
水平方向	
Layout	布局下拉框包含以下成员，用作设置C1ToolBar类型实例的C1ToolBar.ButtonLayoutHorz属性：Default，Image，Text，以及TextAndImage。
Look	ook下拉框包含以下成员，用作设置C1ToolBar类型实例的C1ToolBar.ButtonLookHorz属性：TextAbove，TextBelow，TextOnLeft，以及TextOnRight。
Enforce look on all buttons	Enforce look on all buttons复选框获取水平方向全部按钮的指定外观。
竖直方向	
Layout	布局下拉框包含以下成员，用作设置C1ToolBar类型实例的C1ToolBar.ButtonLookVert属性：Default，Image，Text，以及TextAndImage。
Look	Look下拉框包含以下成员，用作设置C1ToolBar类型实例的C1ToolBar.ButtonLookVert属性：TextAbove，TextBelow，TextOnLeft，以及TextOnRight。
Enforce look on all buttons	Enforce look on all buttons复选框获取垂直方向全部按钮的指定外观。
Reset properties to default	选择重置属性为默认值选项重置修改过的C1ToolBar的布局属性为其默认值。

编辑杂项属性

单击编辑Miscellaneous属性按钮打开C1ToolBar对话框，在此您可以编辑C1ToolBar控件的各种杂项属性。



下表定义了包含在C1ToolBar控件的C1ToolBar对话框中的字段：

项目	描述
Behavior	
Enabled	Enabled复选框指示该C1ToolBar将在运行时启用。（如果选中则为True；否则为False）
Visible	Visible复选框指示是否C1Toolbar控件在运行时可见。（如果选中则为True；否则为False）
AutoSize	AutoSize复选框指示是否工具栏自动调整其尺寸以适应显示全部项目。（如果选中则为True；否则为False）
Show tooltips	显示工具提示复选框获取或设置一个值，该值指示当鼠标移动到工具栏按钮时是否显示工具提示文本。
Minimum size	最小尺寸的字段包括一个指定C1ToolBar最小宽度和高度尺寸的表示宽度和高度的NumericUpDown控件。
Maximum size	最大尺寸的字段包括一个指定C1ToolBar最大宽度和高度尺寸的表示宽度和高度的NumericUpDown控件。
Dock	Dock下拉框包含一些项目，您可以从中选择定义C1ToolBar控件的哪一边绑定到容器。
Close More menu on item click	单击项目关闭额外菜单复选框获取或设置一个值，该值表示当一些项目无法在工具栏上显示时自动创建的菜单，在其中某个项目单击时是否关闭此菜单。
Cursor	Cursor下拉框打开一个不同类型的光标项目列表，您可以从中选择设置当鼠标指针移动到C1ToolBar控件上方时，出现的光标类型。

Use wait cursor	Use wait cursor复选框指示是否使用表示等待状态的光标。
自定义	
customize button	自定义按钮复选框指示是否显示自定义按钮。（如果选中则为True；否则为False）
Options	Options下拉框包含以下成员，用于设置C1ToolBar类型的实例上的C1ToolBar.CustomizeOptions属性的值：AllowAddItem，AllowAll，AllowDelete，AllowNone，AllowRemoveItem，以及AllowToggleCustomizeButton。
Design	
Generate member	Generate member复选框指示是否为C1ToolBar控件的成员生成代码。（如果选中则为True；否则为False）
Locked	Locked复选框指示是否C1ToolBar为锁定状态。（如果选中则为True；否则为False）
Show invisible items	Show invisible items复选框指示是否在C1ToolBar控件上显示不可见的按钮。（如果选中则为True；否则为False）
Margin	Margin指定C1ToolBar控件和另一个控件之间的间距。
Reset properties to default	选择重置属性为默认值项目重置C1ToolBar控件修改过的属性为其默认值。

菜单和工具条概述

C1Command将一套完整的菜单和工具条集成到简单的系统中，这样将允许你通过简单的对象和代码来复用菜单条目和工具条按钮。
C1Command提供五种主要的对象来生成菜单系统，包括C1MainMenu, C1CommandMenu, C1CommandControl, C1CommandMdiList, 和C1ContextMenu。同样的，用于生成工具条系统的几种主要对象包括C1ToolBar，C1CommandMenu，C1CommandControl，C1CommandMdiList，和C1ContextMenu。两类对象唯一的不同在于两个私有控制器：C1MainMenu和C1ToolBar。

C1MainMenu

C1MainMenu是一类主要用于在Windows表单中显示主要菜单的控制器。当你将这个对象放置到你的表单中，它将显示在表单的上部横跨整个窗口，就像常规Windows主菜单一样。除了在表单上部的主菜单之外，一个C1CommandHolder将自动的显示在组件托盘处。C1CommandHolder作为一个单独连接，将存储所有的菜单命令。想要了解更多关于如何使用C1CommandHolder，请参阅C1CommandHolder Component。

C1CommandLink类型的命令链接主要用于代表菜单中的命令。

C1ToolBar

C1ToolBar是一款代表工具条的控制器。就像C1MainMenu一样，它包括一个存储在C1CommandHolder组件中的命令链接。命令链接代表着主菜单的菜单条目，同理，C1ToolBar 的命令链接主要代表工具条上的按钮。

下面的章节主要介绍菜单和工具条的详细功能，并且介绍用于创建菜单和工具条系统的命令和一系列的对象。

菜单和工具条功能

菜单条目和工具条按钮的常用功能与C1Command中的功能类似。菜单条目或者工具条按钮通过两个组件进行区分，分别是：command 和command link。

C1Command 组件的功能

command （一个C1Command类型对象或者或者此对象派生的其他对象，更多详细信息，请参阅ClassHierarchy）主要使用属性以及事件句柄来连接命令代表的实际动作。命令并不包含在C1Commands菜单和工具条中。取而代之的是，所有的表单中的命令都作为一个单独连接存储在C1CommandHolder组件中，这是一个单例对象，它将在你添加第一个C1Commands菜单和工具条时自动创建在表单中。

C1CommandLink 组件功能

command link是一个小巧而简单的组件。Command是它最重要的属性，主要用于指向命令链接中实际的命令对象。除了这一点，命令链接允许你重载部分已经连接的命令属性。例如文本等。命令链接的外观主要取决于两个因素：命令连接到何处，以及连接内是否包含主菜单，弹出式菜单或者工具条。

属性主要用于显示从命令中取出的命令链接。例如，文本或者图片，同理，它们的显示方式将取决于容器。例如，在一个主菜单中，只有命令的文本才能够显示，在一个弹出式菜单中，图片和快捷键同样显示，等等。多个命令链接可以指向同一个命令，这也是命令和命令链接进行条目区分的主要原因之一。

命令，命令链接，菜单和工具条以及命令持有者的关系。

简单概括后，下面的关系存在于表单中的命令，命令链接，菜单和工具条以及命令持有者之间：

- 命令（C1Command类和其子类）自动存储在表单中的命令持有者（一个C1CommandHolder类型对象）中菜单和工具条（一个C1MainMenu, C1CommandMenu, C1ContextMenu,C1ToolBar类型对象），包括命令链接（C1CommandLink类型），命令链接代表菜单条目或是工具条按钮。每个命令链接都代表一个命令持有者中的实际命令。命令链接存储在菜单或是工具条中的CommandLinks 连接中。
- 多个命令链接能够指向同一个命令。并且命令链接可以指向不同容器中的同一个命令。例如，文件菜单或者其他文件选项工具条中的链接可以指向同一个文件打开命令。
- 用户可以看到大部分命令链接中的属性（文本，图片或是其他等等）。命令链接的显示状态（可见/不可见，选中/未选中，以及其他等等）都由已经连接的命令（命令链接并不含有状态属性）的一致状态决定。
- 更重要的是，事件句柄将展示用户定义的实际动作（例如，打开文件，或是复制到剪贴板），通常使用commands 而不是command links。当一个菜单条目被选中或是一个工具条按钮被点击之后，已连接命令的单击事件句柄将被激活。

想要列举所有报表中的命令，可以使用Commands 收集所有的命令持有者（显示在表单组件工具栏中）。你同样可以使用Connection编辑器来添加或是删除命令（虽然使用菜单或是工具条设计器是一种更简单的方式来访问命令的链接）。

创建菜单和工具条常用对象

C1Command的菜单和工具条使用下面的对象来创建菜单和工具条系统：

- C1CommandHolder component
- C1CommandMenu command
- C1ContextMenu control
- C1CommandControl command
- C1CommandMdiList command

下面的章节主要介绍创建菜单和工具条过程中使用到的命令和组件。

C1CommandHolder组件

C1CommandHolder是一个命令容器。它同样包含了一个命令的图片列表，一部分其他常用设置。每个表单中只能安放一个C1CommandHolder 。

当你创建了一个C1Command后，它会自动添加到表单的命令持有者中。如果之前没有命令存在，它会自动由命令的设计器创建一个新的命令持有者。

命令持有者提供下述特性：

- 它同时还是一个IExtenderProvider ，主要用于向表单中的控制器提供C1ContextMenu 属性。
- 提供空闲时自动更新命令状态，如可见性，可用性，是否选中等。

C1CommandMenu 命令

C1CommandMenu组件是一个菜单式的命令（从C1Command基类中派生）。除了其他的C1Command属性，它还包含了一个命令链聚集，其中包含了本菜单中的菜单条目。C1CommandMenu可以作为子菜单包含在其他的菜单中。

当你创建一个新的C1CommandMenu时，系统将会添加一个空的命令链接到其中。这个过程与添加空命令链接到一个新的主菜单中过程类似。

想要了解更多的关于如何使用C1CommandMenu 命令的信息，请参阅Menu任务。

C1ContextMenu 控制器

C1ContextMenu 组件是一个菜单（从C1CommandMenu基类中派生得出），可以用于连接到任何一个控制器中当作上下文菜单来使用。为了使这一过程变得更加容易，C1CommandHolder（在使用C1菜单的表单中始终存在）同时还是一个IExtenderProvider，用于向表单中的所有控制器提供C1ContextMenu类型的C1ContextMenu属性。

需要注意的是，C1ContextMenu与它的基类C1ContextMenu类似，同样能够在其他菜单中使用。因此，如果你想要在主菜单系统使用同样的菜单作为子菜单或者作为上下文菜单，只需要将C1CommandLink连接到相同的C1ContextMenu的同一个位置。

想要了解更多的关于如何使用C1ContextMenu命令的信息，请参阅Context Menu Tasks。

C1CommandControl命令

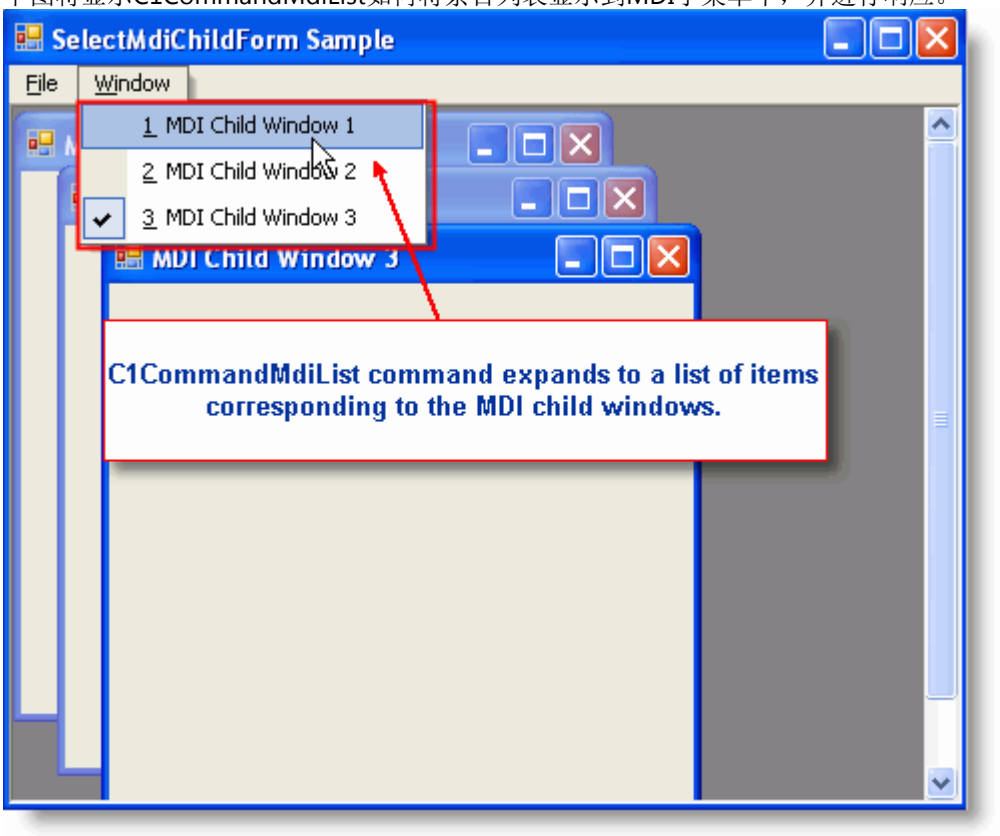
C1CommandControl是一个可以连接到任何控制器中的命令。这一特性由C1CommandControl类提供，它派生于C1Command类。控制器能够从Visual Studio的工具栏中拖拽出来，将其放置到C1MainMenu或者C1ToolBar中。它将自动创建一个C1CommandControl，并将其连接到放置的控制器上，然后将新的命令链接添加到工具条中。

> **注意：**一个小型的控制器将添加到它的空间中，但是它并不能处理大型控制器，例如容器。

C1CommandMdiList 命令

C1CommandMdiList组件是一个命令，它在运行时将会扩展到一个条目列表中，主要用于响应当前窗口的MDI 子窗口。需要注意的是，它不是一个子菜单。你可以通过它将命令加入一个子菜单中，或者在创建它之前或之后添加其他的菜单条目。

下图将显示C1CommandMdiList如何将条目列表显示到MDI子菜单中，并进行响应。



通过设置MaxItems属性，你可以限制显示在它的菜单列表中的C1CommandMdiList命令条目为你想要显示的数量。这

个属性缺省值为10。

你还可以在菜单列表中显示隐藏的MDI窗口，只需要将ListHidden属性设置为True。

菜单和工具条中的唯一对象

菜单和工具条共享许多对象，然而，他们中有两个独特的组件。菜单中包含一个C1MainMenu控制器，它是菜单中的主菜单。工具条中包含一个C1ToolBar控制器，它代表整个工具条系统。

下面的章节将为你介绍C1MainMenu和C1ToolBar控制器，提供更多的关于它们外观和行为属性的详细信息。

C1MainMenu 控制器

C1MainMenu是一个在窗口表单中显示主菜单的控制器。当你将这个对象放置到你的表单中时，它将横跨整个表单的上不，就像常规的Windows主菜单做的那样。除了表单顶部的主菜单，它将自动添加一个C1CommandHolder 到组件托盘中。C1CommandHolder作为一个独立对象，将会存储所有的菜单命令。每个表单中只能存在一个C1MainMenu控制器。

设计时添加C1MainMenu 控制器：

在Visual Studio工具栏中，双击C1MainMenu组件，或者将其拖拽、放置到表单中。

编程时添加C1MainMenu控制器：

Visual Basic

Visual Basic

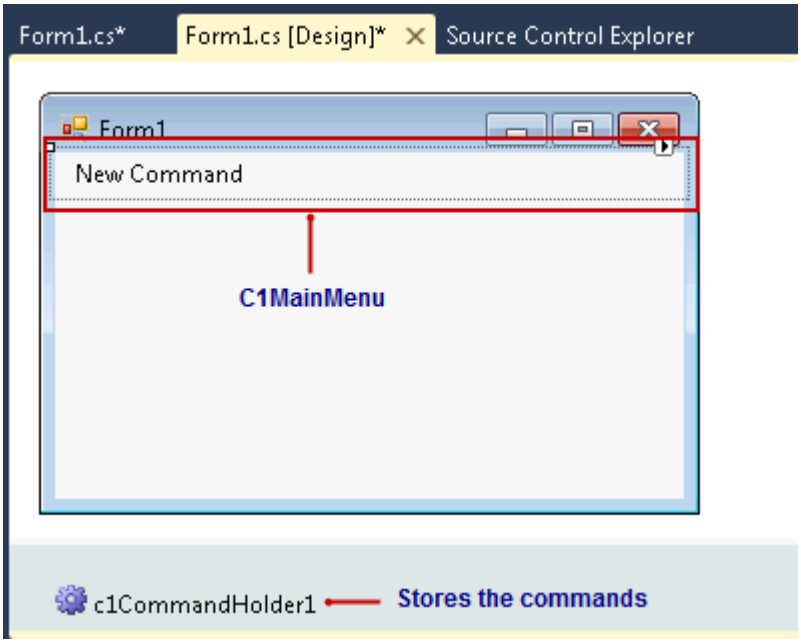
```
Imports C1.Win.C1Command
Dim ch As C1CommandHolder(Me)
Dim mm As New C1MainMenu
Me.Controls.Add(mm)
```

C#

C#

```
using C1.Win.C1Command
C1CommandHolder.CreateCommandHolder(this);
C1MainMenu mm = new C1MainMenu();
this.Controls.Add(mm);
```

下面的屏幕截图显示一个C1MainMenu控制器被添加到表单后的效果：



C1MainMenu控制器包含一个Link to Command设计器，将会允许你很方便的配置菜单。

注意: 这个编辑器对所有的C1CommandLinks都有效；因此，你可以很简单的为任何对象编辑他们的所有命令：C1ContextMenu, C1ToolBar, 和C1OutBar。

C1ToolBar控制器

C1ToolBar控制器在表单中作为工具条使用。当你将这个对象放置到你的表单中时，就像C1MainMenu一样，它会自动创建一个C1CommandHolder，并将其添加到组件托盘中。C1CommandHolder作为一个单独聚集，将会存储所有的命令链接。这些命令链接代表着主菜单中的菜单条目，同理，C1ToolBar中的命令链接代表着工具条上的按钮。

一旦C1ToolBar组件加入到表单中，Link to Command设计器将允许你设置工具条系统。C1ToolBar和C1MainMenu使用同样的Link to Command设计器。想要了解更多关于Link to Command 界面的信息，请参阅LinktoCommandDesigner。

C1ToolBar提供两种不同类型的工具条：缺省工具条和一个下拉式的工具条。工具条按钮为下拉菜单提供下拉按钮。这些按钮根据工具条的方向，可以调整为竖向排列以及横向排列。

菜单的外观和行为

菜单提供一系列非常有用的属性来控制主菜单和菜单条目的行为与外观。

C1Command的菜单包含一整套外观属性来实现提升外观和定制控制器。菜单的风格，大小以及布局都可以使用C1MainMenu的外观属性来很轻易的实现定制。这些属性可以在设计时通过属性窗口进行设置，或是通过编码实现。

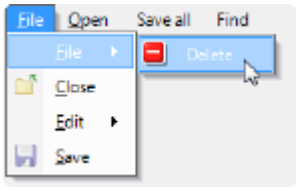
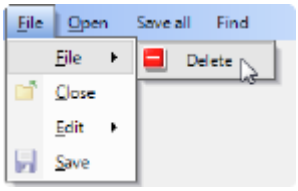
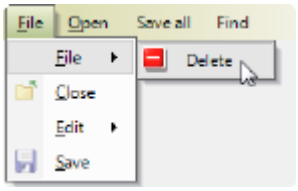
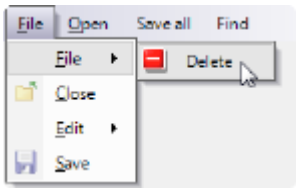
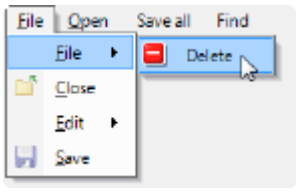
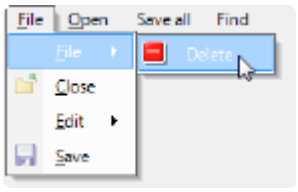
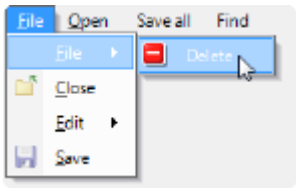
C1Command的菜单同样还包含了若干有用的行为属性，用于在菜单条目中包装，合并，以及显示工具提示。

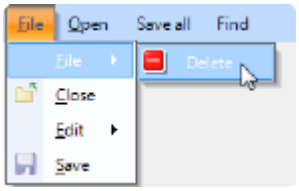
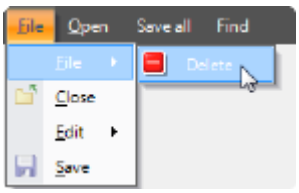
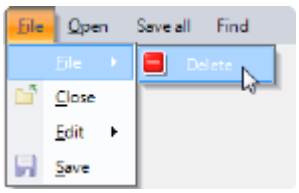
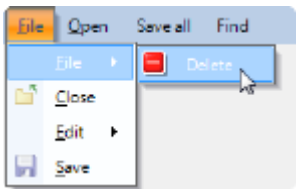
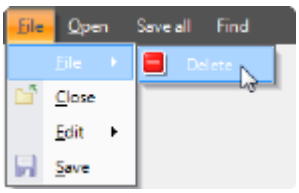
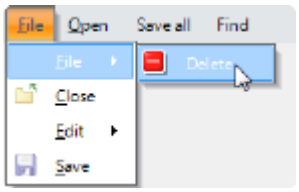
下面的章节主要介绍一些C1MainMenu菜单视觉风格中常用的外观和行为属性。

菜单视觉风格

C1MainMenu和C1ContextMenu控制器提供若干种内置的风格，例如：Custom, System, Office2010Blue, Office2010Black, Office2010Silver, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, and WindowsXP。使用控制器的VisualStyle 属性可以很轻松的将这些风格应用到你的菜单中。

下面表格中将介绍C1MainMenu控制器的每一种风格内容。C1ContextMenu控制的视觉风格和C1MainMenu控制器的完全相同，只是C1ContextMenu控制器不包含菜单条而已。

属性设置	图像
VisualStyle.Custom	[Custom风格允许你定制自己的控制器视觉风格]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	

VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

菜单条目的外观和感觉

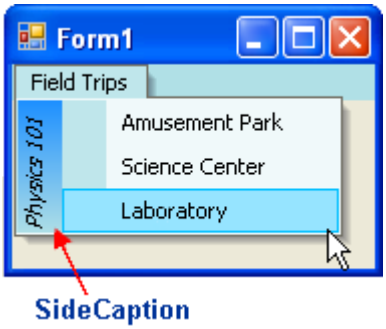
C1Command 支持各种菜单条目的设置，包括字体风格，边框风格，背景色以及鼠标移动的风格。

你可以在命令对象中，或者它的命令链接中设置工具条按钮/命令的文本名称。然而，C1CommandLink.Text 属性重载了C1Command.Text属性。

特殊侧标题栏风格的菜单

C1Command 在它的C1CommandMenu常用风格中包含了一种特殊的侧标题栏属性。使用这个属性，你可以在子菜单条目为一个特殊的C1CommandMenu显示一个侧标题栏。你可以在标题栏中显示文字或是图片。侧标题栏中除了包含一个文本或者图片外，你还可以定制标题栏的外观和布局。

下面这张图显示了一个Field Trips菜单纵向侧标题栏效果：



想要了解更多关于如何应用侧标题栏到你的菜单中，请参阅 Creating a Side Caption for a Command Menu.

菜单条目中鼠标移动风格

你可以在菜单条目中应用鼠标移动技术来提高菜单的用户交互设计。

C1MainMenu组件有两个特殊的属性，它们主要用于应用鼠标移动技术。
BackHiColor属性设置后，当你将鼠标移动到菜单条目上，菜单条目将显示当前条目的背景色。ForeHiColor 设置后，当你将鼠标移动到菜单条目上时，菜单条目将显示前景色。

想要了解更多关于如何使用这些属性，请参阅 Modifying the Appearance of the Menus.

合并菜单

某些情况下，当你需要将MDI子菜单与MDI父菜单合并时，你可以使用CanMerge属性。同时，你还可以通过MergeType属性来制定合并菜单的行为类型。使用这一属性，你可以决定是否添加，替换，删除或者合并菜单条目。MergeItems 会将菜单上的命令链接合并到一起。

无论是菜单条目还是工具条按钮的命令链接都包含一个MergeOrder属性，这条属性主要用于决定合并菜单条目或是工具条按钮的顺序。

想要了解更多关于如何合并菜单，请参阅Merging Menu Items.

菜单布局和文字包装

C1MainMenu包含一个自动布局。菜单条目将会自动调整大小以适应显示区域。
C1MainMenu 的Wrap属性允许你调整主菜单条中的行列包装。如果主菜单中包含太多的菜单条目想要放置到同一行中，它将自动调整菜单条目的大小。

菜单中的工具提示

工具提示主要用于当鼠标移动到控制器上时，显示相应的文本信息。C1MainMenu提供一个ShowToolTips属性，用于将Text属性的值作为一个每个菜单条目的工具提示显示在界面上。这个属性缺省情况下是打开状态。

如果你想要为每个菜单条目输入自定义文本信息，你可以通过设置ShowTextAsToolTip为False，然后在ToolTipText属性中设置自定义文本来实现这一需求。

想要了解更多关于如何使用工具提示，请参阅Displaying ToolTips for Menus and Toolbars。

工具条的外观和行为

C1ToolBar提供大量有用的属性来控制工具条和工具条按钮的行为和外观。

C1ToolBar包含各种外观属性来实现视觉的提升以及控制器的定制。工具条的风格，大小以及布局都可以通过C1ToolBar的外观属性来定制。这些属性可以在设计时通过属性窗口设置以及通过编码实现。

除了通过属性设置工具条的外观，C1ToolBar 还提供若干有用的行为属性来锁定或是浮动工具条，移动工具条按钮，在工具条中植入任意控制器，运行时定制工具条，设置工具条按钮布局是横向或是纵向，在工具条上显示工具提示以及它的命令按钮，以及在工具条按钮上包装文字。

下面的章节将为你介绍一些C1ToolBar控制器中常用的外观和行为属性。

工具条视觉风格

C1ToolBar控制器提供若干内置的风格，例如Custom, System, Office2010Blue, Office2010Black, Office2010Silver, Office2007Blue, Office2007Black, Office2007Silver, Office2003Blue, Office2003Olive, Office2003Silver, OfficeXP, Classic, 和 WindowsXP。这些风格都可以通过设置VisualStyle 属性来实现。

下面的表格中介绍了每一个C1ToolBar控制器的视觉风格

Property Setting	Image
VisualStyle.Custom	[Custom允许你定制属于你自己的视觉风格]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

工具条外观和感觉

C1Command 支持各种工具条和工具条按钮的设置，包括字体风格，边框风格，背景色以及鼠标移动的风格。

你可以在命令对象中，或者它的命令链接中设置工具条按钮/命令的文本名称。然而，C1CommandLink.Text 属性重载了C1Command.Text属性。

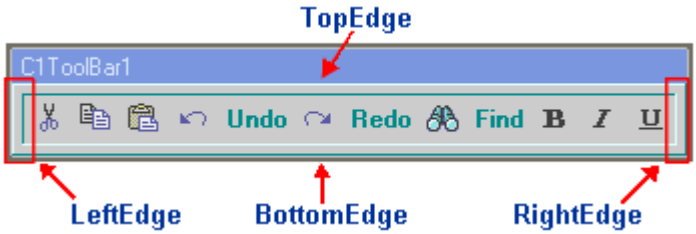
想要了解更多关于如何使用常用外观属性，请参阅Modifying the Appearance of the Toolbar。

工具条中的特殊边框风格

C1ToolBar 有一个特殊的类，C1Border，这个类允许你添加各种边框风格到工具条中。C1Border包含以下几个成员：

名称	描述
BottomEdge	决定边框是否存在底边
DarkColor	获取或者设置分组的颜色。在Style的结构设置中，这个颜色将应用于工具条的顶部，底部，左侧和右侧边框。
LeftEdge	决定边框是否存在左侧边线。
LightColor	获取或者设置边框颜色。该颜色并不应用于C1Border的结构化设置中。
RightEdge	决定边框是否存在右侧边线。
Style	获取或者设置边框风格
TopEdge	决定边框是否存在顶部边线。
Width	按照像素单位，获取或者设置边框宽度

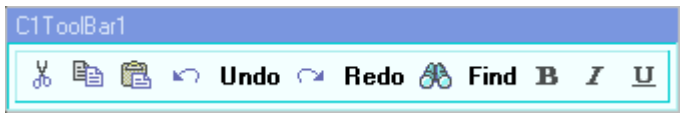
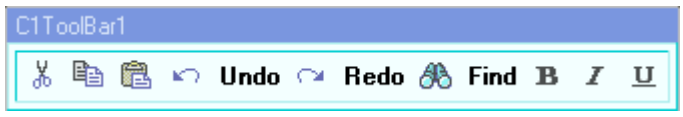
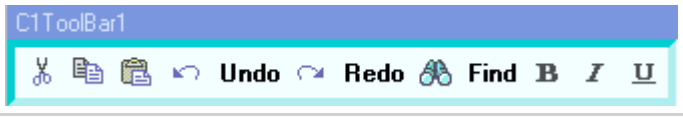
下面的图片介绍了C1Border 类的TopEdge, LeftEdge, BottomEdge, 和RightEdge属性。



TopEdge, LeftEdge, BottomEdge, 和 RightEdge属性在将边框应用到C1ToolBar的指定区域例如顶部，底部，左侧或者右侧时，是非常有效的。这些属性缺省情况下默认为True。

下面的表格介绍了Style属性的每个属性设置。除了各种显示在下方的边框风格外，表格中还介绍了Width, DarkColor, LeftEdge, RightEdge, BottomEdge, 和TopEdge属性。Width属性设置为5px，DarkColor属性设置为DarkTurquoise，LeftEdge属性设置为PaleTurquoise，以及 LeftEdge, RightEdge, BottomEdge, 和TopEdge属性都被设置为True。

属性设置	图像
Style.None	
Style.Flat	

Style.Groove	
Style.Ridge	
Style.Inset	
Style.Outset	

下面的表格中将介绍当LeftEdge, RightEdge, BottomEdge, and TopEdge 属性被设置为disabled时的效果：

属性设置	图像
BottomEdge.False	
LeftEdge.False	
RightEdge.False	
TopEdge.False	

想要了解更多关于如何使用这些属性，请参阅 Modifying the Appearance of the Toolbar。

工具条按钮的鼠标移动风格

你可以在工具条按钮中应用鼠标移动技术来提高工具条的用户交互设计。

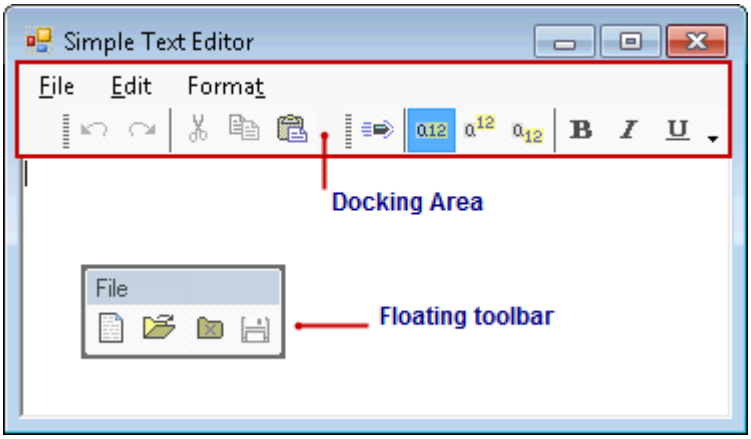
C1ToolBar组件有两个特殊的属性，它们主要用于应用鼠标移动技术。
BackHiColor设置后，当你将鼠标移动到菜单条目上，菜单条目将显示当前条目的背景色。ForeHiColor设置后，当你将鼠标移动到菜单条目上时，菜单条目将显示前景色。

想要了解更多关于如何使用这些属性，请参阅 Modifying the Appearance of the Toolbar。

锁定和浮动工具条

在C1CommandDock在容器内分配的区域内，工具条可以被锁定到顶部，底部，左侧或者右侧区域。

每一个C1ToolBar被锁定时，都将按照锁定区域大小重新调整尺寸。使用一个下拉选项，工具条可以在不同的锁定区域内移动，它们同样会自动调整大小。



如果你在编程过程中创建一个C1ToolBar，你将会非常喜欢使用C1CommandDock来设置你添加到C1CommandDock 中的工具条的锁定或者浮动行为。具体代码如下所示：

Visual Basic

```
Visual Basic
Me.C1CommandDock = New C1.Win.C1Command.C1CommandDock()
Me.C1CommandDock.Controls.Add(Me.C1ToolBar1)
Me.Controls.Add(Me.C1CommandDock)
```

C#

```
C#
this.c1CommandDock = new C1.Win.C1Command.C1CommandDock();
this.c1CommandDock.Controls.Add(this.c1ToolBar1);
this.Controls.Add(this.c1CommandDock);
```

添加控制器到工具条

C1CommandControl允许你将任意控制器添加到工具条中。

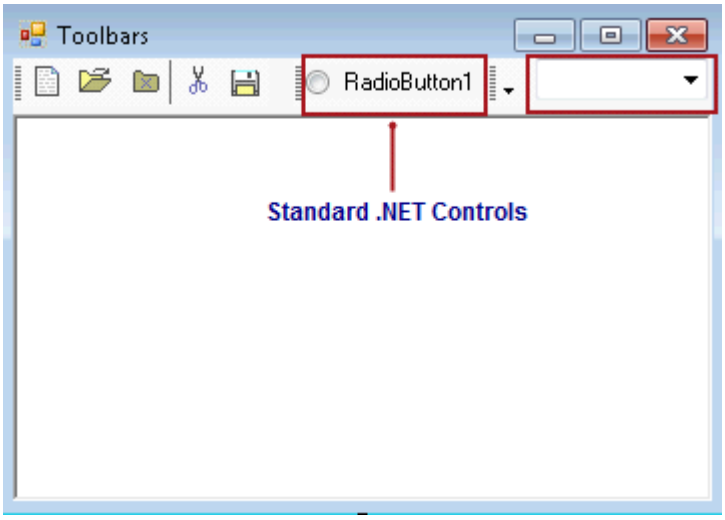
任意的控制器，如文本框，都可以通过C1CommandControl添加到C1ToolBar中。

你可以很简单的通过拖拽一个任意控制器到工具条中，然后在设计时添加一个C1CommandControl命令类型，或者通过编码添加一个C1CommandControl命令类型。

想要了解更多关于如何添加任意控制器到C1ToolBar对象中的信息，请参阅AddinganArbitraryControltotheToolbar。

当任意一个控制器被拖拽到工具条中，它将自动创建一个叫做C1CommandControl的新命令类型。C1CommandControl包含一个Control属性，允许任意控制器与命令进行连接。

下图中将显示将RadioButton, CheckBox, 和一个ComboBox控制器添加到C1ToolBar中。

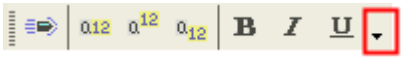


运行时定制工具条

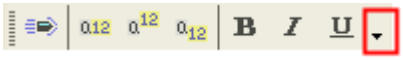
C1ToolBars 通过在设计时设置CustomizeButton属性为True，可以实现运行时定制工具条功能。

> **注意：工具条需要提前添加到C1CommandDock 中，如果你想要在设计时将它的 CustomizeButton属性设置为True。**

设计时，当定制生效将在工具条上显示一个下拉箭头。



当你单击这个下拉箭头时，运行时将会出现一个弹出菜单。



定制工具条菜单选项如下所示：

添加或者删除按钮

单击菜单中的命令条目，从而将该命令按钮从工具条中删除。

重置

单击Reset菜单条目，重置工具条为初始设置。

定制

单击Cusomize菜单条目，打开Customize toolbars对话框。

TheCustomizeDialogcontainsthreetabsformodifyingtheC1ToolBarcomponent:
定制对话框包含三个选项卡，主要用于修改C1ToolBar 组件：

- Toolbars -这一选项卡包含C1ToolBar 组件的创建，重命名，删除，修改选项。
- Commands -这一选项卡包含添加已存在的命令到工具条的选项。

- Options -这一选项卡包含修改C1ToolBars 一般外观属性的选项，例如工具条的外观和感觉，字体和颜色。

在Customize toolbars 对话框每个选项卡的底部，都包含一个Save, Restore, Reset, OK, 和Cancel命令按钮，这些按钮能够实现保存工具条更新后的设置，恢复更新后的设置，重置缺省设置，接收新设置，以及取消定制工具条等功能。

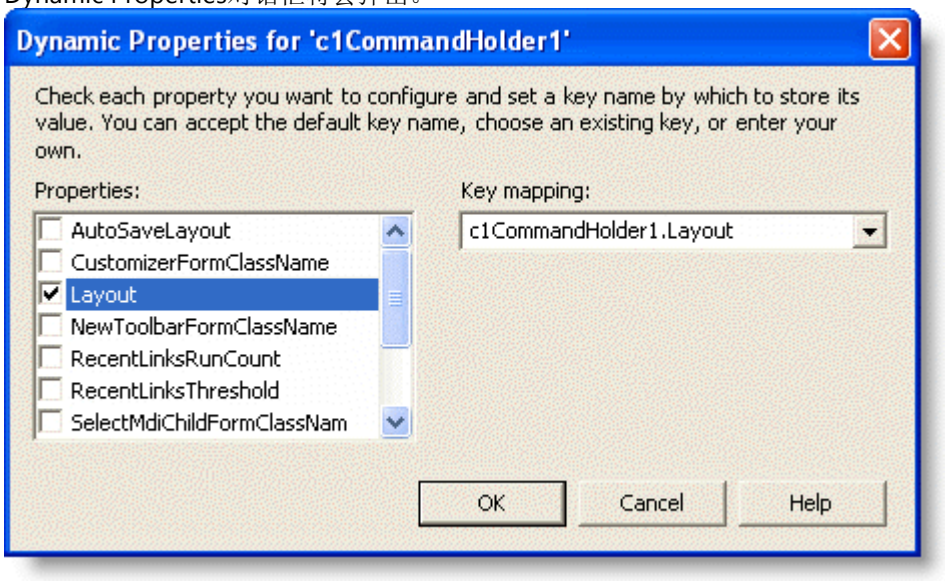
用户最后的定制配置信息将保存在应用的配置文件中，命令持有者的环境属性同样添加到动态属性中。

注意：动态属性的用户界面在Visual Studio 2005中将移除掉。它仍然支持动态属性。想要了解 更多关于如何使用动态属性的信息，请参阅Microsoft Visual Studio 2005 文档:Introduction to Dynamic Properties (Visual Studio).

想要将布局保存到应用的配置文件中，完成以下步骤：

1. 单击表单组件托盘中的C1CommandHolder 。
2. 展开DynamicProperties 节点，然后单击Advanced 属性旁边的ellipsis按钮。

Dynamic Properties对话框将会弹出。



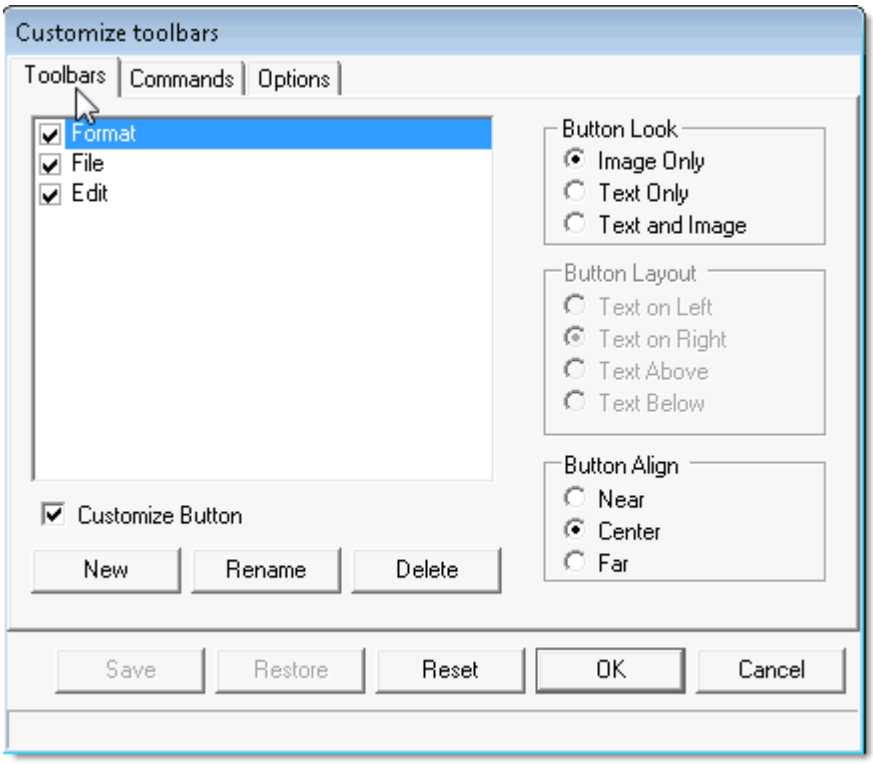
3. 单击Layout勾选框，然后单击OK按钮。布局将保存在应用的配置文件中，而不是保存在表单的代码中。

注意：当你使用Visual Studio设计器编程时，Visual Studio将会创建一个app.config文件在项 目路径下，然后当每次运行时将会用实际的应用配置文件（位于bin路径下）替换这个 app.config文件内容。因此，如果你在Visual Studio中运行该项目，改变工具条布局，关闭 它，然后再次运行这个项目，你将看不到上次存储的布局。这不是一个系统错误，当你不用 Visual Studio运行该应用时，一切工作将会很好的完成。

除了使用表单的动态属性保存你的工具条布局，你还可以使用你自己的框架保存以及恢复工具条布局。想要实现更好的控制效果，你可以使用编码实现保存和设置Layout属性。

工具条

Toolbars 包含了一些用于创建和操作工具条的选项。



缺省情况下，ButtonLook和CustomizeButton属性是失效的。

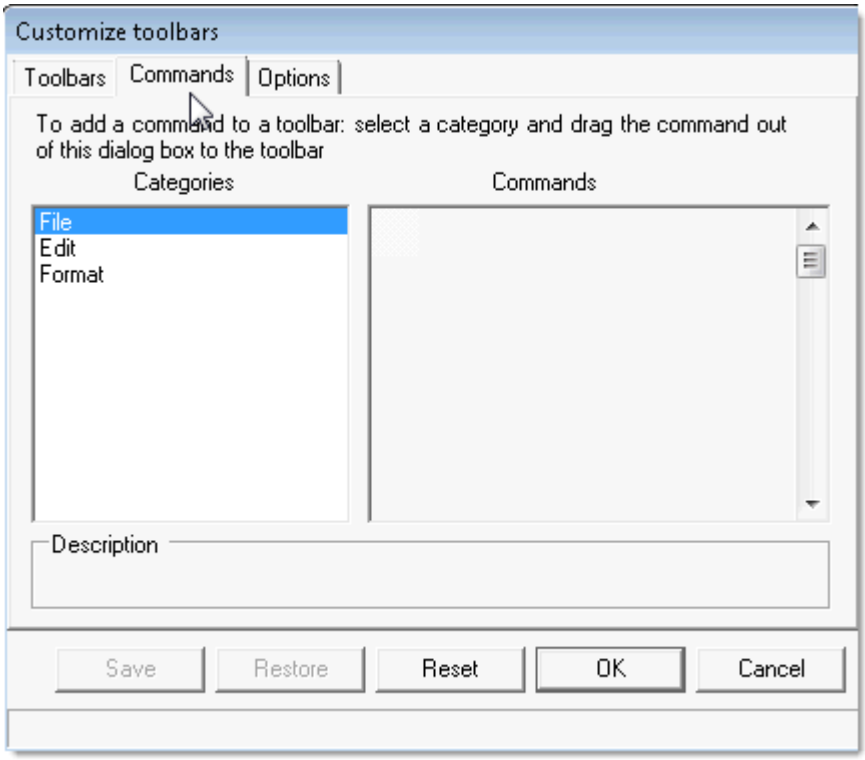
按钮布局属性只有在ButtonLook属性设置为Text and Image（当Text and Image单选按钮被选中）时才能生效。这是因为ButtonLayoutHorz属性将决定文本在图片中如何放置（位于图片之上，之下，左侧，或者是右侧）。

当C1ToolBar 添加到对话框中后，CustomizeButton 即时生效。

命令

Commands 选项包含两个列表框：Categories 和Commands。Categories 列表框包含所有命令的分类，Commands 列表框包含了每个分类的所有命令。

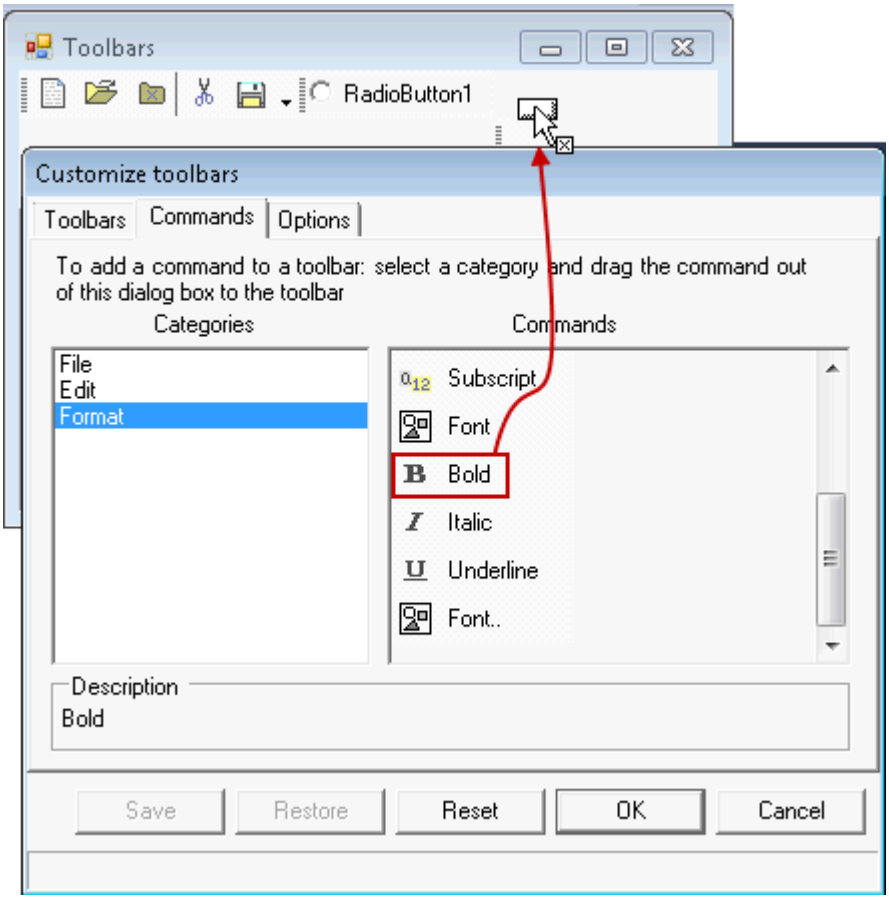
注意：如果命令的Category属性并没有设置，分类列表框将显示为空。



完成以下操作，命令可以很简单的添加到工具条：

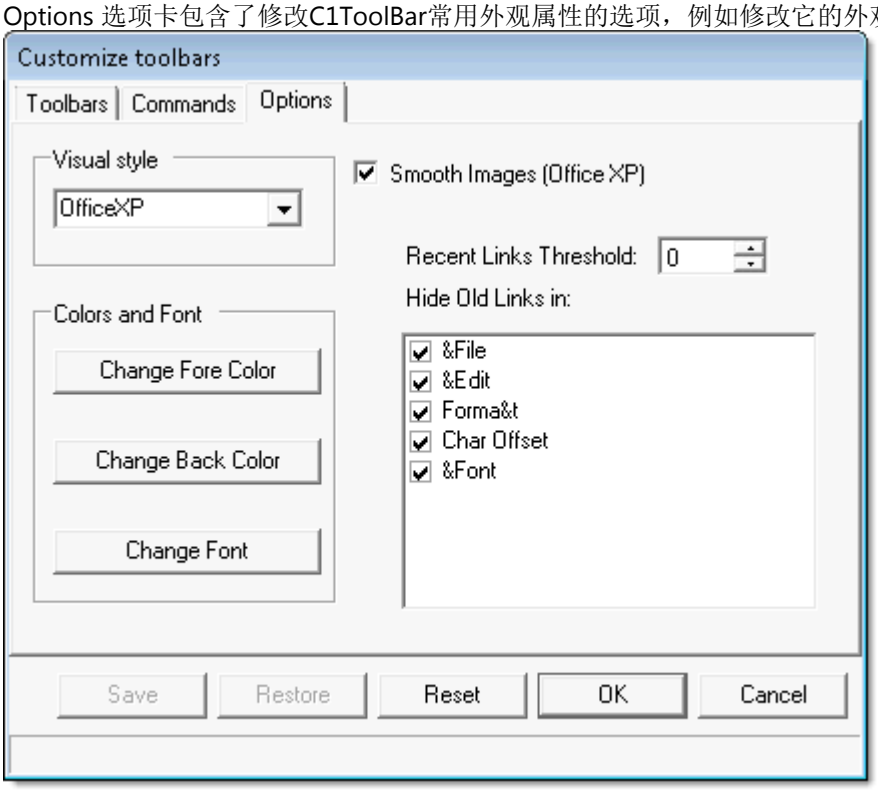
- 在Categories 列表框中选择一个分类
- 从Commands 列表框中选择一个命令，然后将其拖拽到想要放置的工具条上。

下图展示了运行时一个命令从命令列表框拖拽到表单中的格式工具条中。



选项

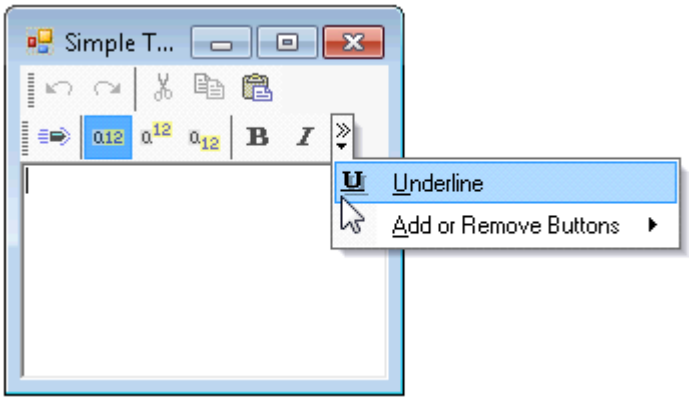
Options 选项卡包含了修改C1ToolBar常用外观属性的选项，例如修改它的外观和感觉，字体和颜色。



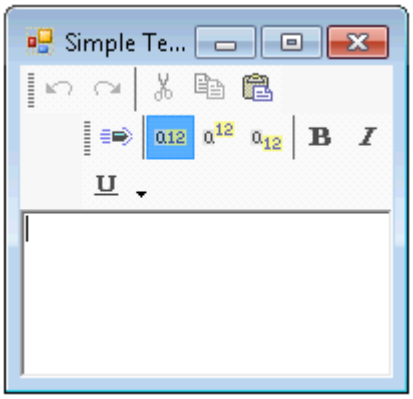
包装工具条按钮和文本

C1ToolBar为工具条的按钮提供包装功能，同样还可以为工具条中的按钮包装文本信息。Wrap 属性能够将工具条覆盖到其他行，从而使所有的工具条按钮都显示出来。缺省情况下，这一属性是生效的。

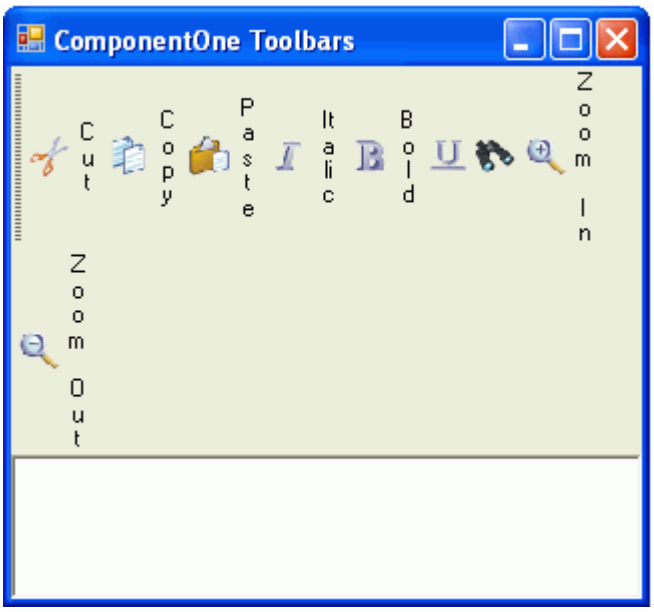
下图展示工具条按钮是如何显示的，当Wrap属性设置为False时。



下图展示工具条按钮是如何显示的，当Wrap属性设置为True时。



下图展示工具条按钮是如何显示的，当Wrap属性设置为True并且WrapText属性设置为True时。



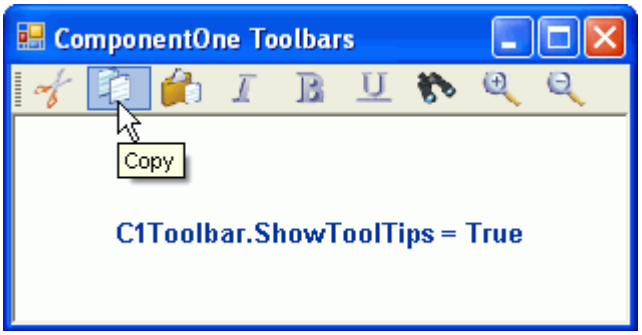
工具条中的工具提示

工具提示主要用于当鼠标移动到控制器上时，显示文本信息。C1ToolBar提供ShowToolTips属性来为每一个工具条按钮将Text属性的值作为工具提示显示出来。缺省情况下，这一属性是生效的。

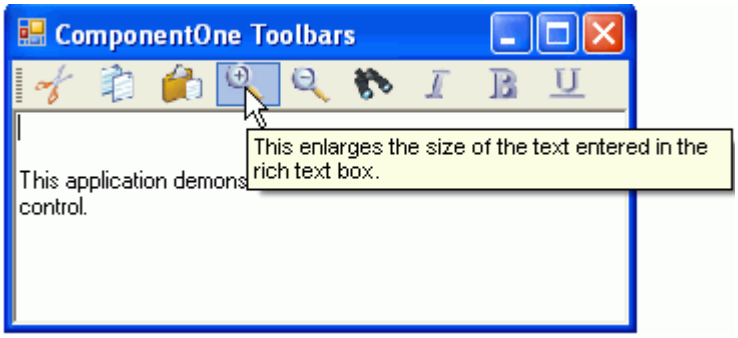
> **注意：**如果你设置了按钮的C1CommandLink.Text属性，而不是它的C1Command.Text属 性。工具提示将获取C1Command.Text的缺省名称进行显示。例如，如果它是第一个按 钮，工具提示以及C1Command.Text的名字都将变成Button1.

如果你想要为每个工具条按钮设置自定义文本的工具提示，你可以通过设置ToolTipText属性来实现。

下图将展示一个ShowToolTips属性设置为True的工具条。



下图将展示一个ShowToolTips属性设置为True的工具条，并且通过C1CommandLink的第四个ToolTipText属性输入自定义文本。



想要了解更多关于如何使用工具提示的信息，请参阅 Displaying ToolTips for Menus and Toolbars。

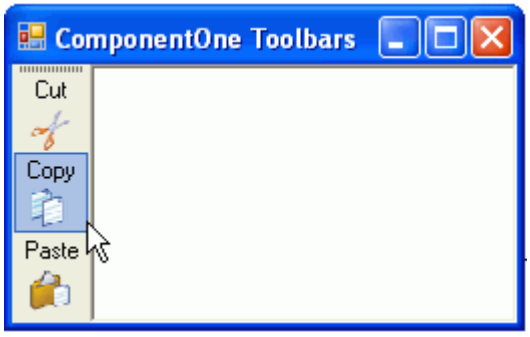
工具条和按钮的布局行为

C1ToolBar的布局是非常自由的。它们可以横向或者是纵向排列，同时还可以锁定在表单中的指定区域。工具条的Movable属性缺省情况下是生效的。这将允许用户在表单中的任何位置移动工具条。缺省的工具条布局是横向的，你可以通过将Horizontal属性设置为False，从而改变工具条的布局。

> **注意：**当你设置ToolBarStyle属性为DropDownMenu时，菜单将向一个下拉菜单那样进行工 作。因此，工具条是固定的。

除了工具条的方向，C1ToolBar还为纵向的工具条提供按钮校准。使用ButtonAlign属性，你可以设置图片或者文本靠近，居中或者是远离按钮。

下面的表格中将展示ButtonAlign属性的值：

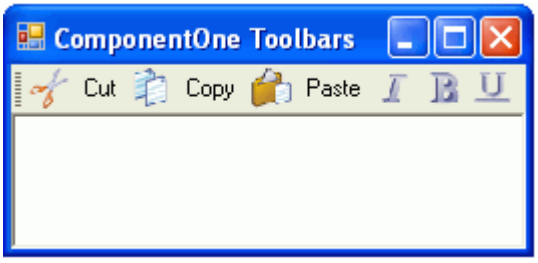
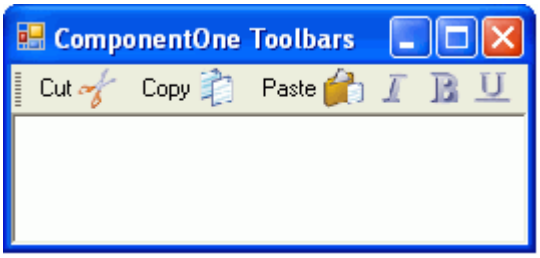
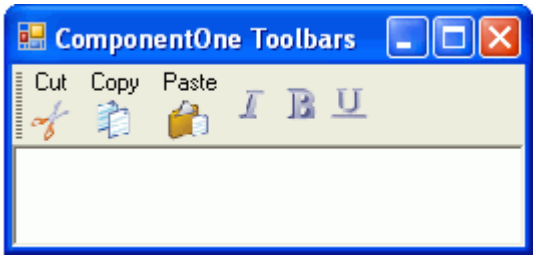
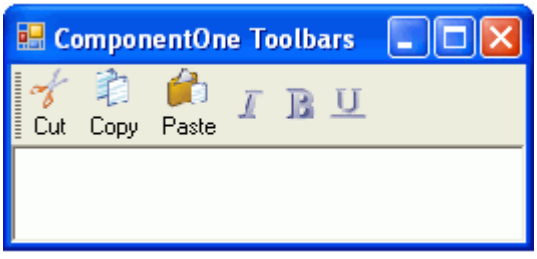
属性设置	图像
ButtonAlign.Near	
ButtonAlign.Center	

ButtonAlign.Far	

你可以使用ButtonLayoutHorz和ButtonLayoutVert属性来决定横向以及纵向工具条中的工具条按钮文本和图片的相对位置。
当工具条是横向时，ButtonLayoutHorz属性将获取按钮的布局。这是工具条的缺省方向。当工具条是纵向时，ButtonLayoutVert属性将获取按钮的布局。将Horizontal属性设置为False，你可以使工具条变为纵向。

注意: ButtonLayoutHorz属性的缺省配置为TextOnRight。

C1ToolBar 提供若干选项来定制横向或者纵向的工具条的工具条按钮。

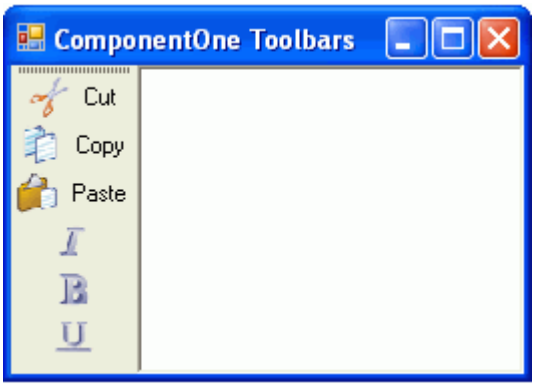
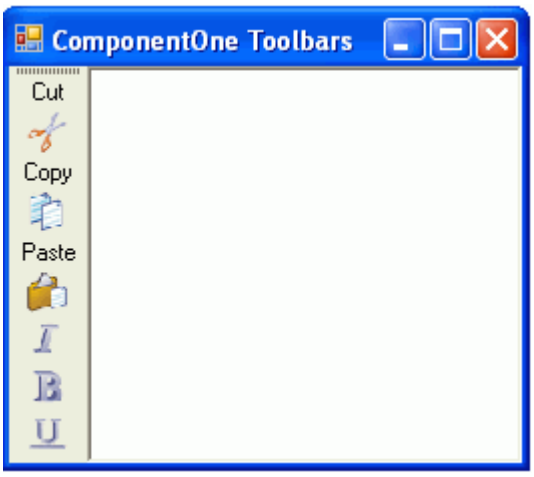
属性设置	图像
ButtonLayoutHorz.TextOnRight (default)	
ButtonLayoutHorz.TextOnLeft	
ButtonLayoutHorz.TextAbove	
ButtonLayoutHorz.TextBelow	

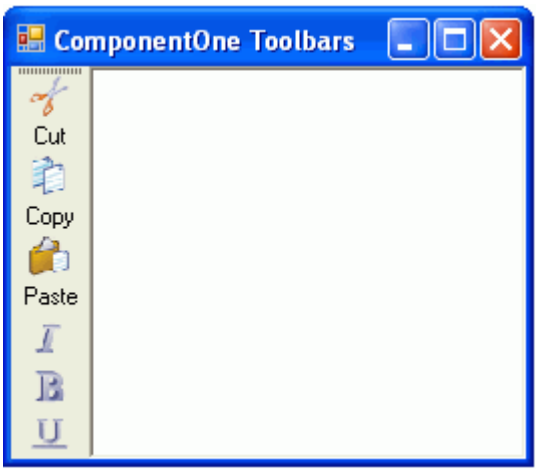
除了控制工具条按钮上的文本和图片的相对位置，你还可以使用ButtonLookHorz属性来显示文本，图片，或者同时在

横向工具条中显示两者。ButtonLookVert属性用于在纵向工具条中显示文本，图片，或者两者同时显示。

> **注意：**ButtonLook的ButtonLookText, Image, 和TextAndImage 值重载了ButtonLookHorz和ButtonLookVert属性的值。如果你设置了ButtonLookHorz或者ButtonLookVert属性的值，那么ButtonLook属性的应该设置为缺省。

下面的表格中显示了ButtonLayoutVert属性的值：

属性设置	图像
ButtonLayoutVert.TextOnRight	
ButtonLayoutVert.TextOnLeft	
ButtonLayoutVert.TextAbove	

ButtonLayoutVert.TextBelow (default)	

DockingTab概述

C1DockingTab控制器实现了用户熟悉的选项卡控制界面，用户可以通过控制器侧边的选项卡来控制若干个页面（每个页面中都包含了任意的控制器）。

除了这一标准操作，C1DockingTab控制器还提供了对接和浮动功能。使用这一功能，整个控制器页面，或者是个别页面（选项卡）能够分割开来，并且自动的对接到表单的另一边，或者是对接到其他的C1DockingTab控制器中，亦或是在一个单独的工具窗口中浮动。想要使用C1DockingTab控制器的对接以及浮动功能，用户必须将C1DockingTab添加到C1CommandDock中。

DockingTab外观和行为属性

C1DockingTab提供了一些有用的属性，用来控制选项卡条目的行为以及外观。

C1DockingTab包含了各种外观属性，从而实现提升视觉效果以及定制控制器。控制器的选项卡风格，大小以及布局都能够很简单的通过C1DockingTab的外观属性实现定制。这些属性能够在设计过程中通过属性窗口进行设置，或是通过编码实现。

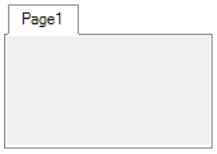
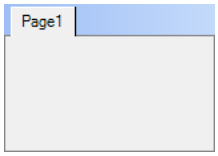
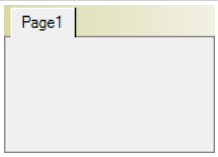
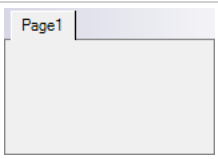
除了这些外观属性，C1DockingTab还提供了若干有用的行为属性，用于关闭选项卡页面，重新排列选项卡，以及制定选项卡页面上的鼠标悬停效果。

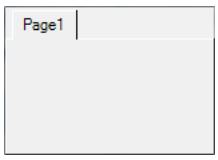
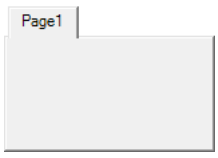
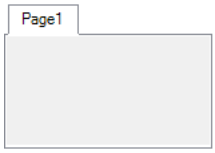
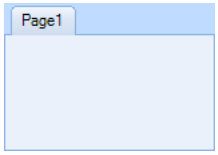
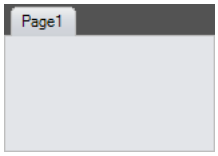
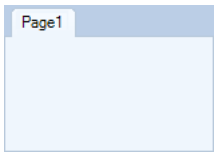
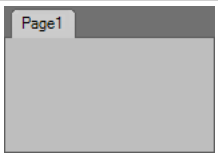
下面的章节包含了一些C1DockingTab控制器常用的外观属性和行为的介绍。

DockingTab视觉样式

C1DockingTab控制器提供若干个内置的风格，例如Custom,System,Office2010Blue,Office2010Black,Office2010Silver,Office2007Blue,Office2007Black,Office2007Silver,Office2003Blue,Office2003Olive,Office2003Silver,OfficeXP,Classic和WindowsXP，使用VisualStyle属性，用户可以很轻松的应用这些风格。

下面的表格中介绍了每一个C1DockingTab控制器的视觉样式。

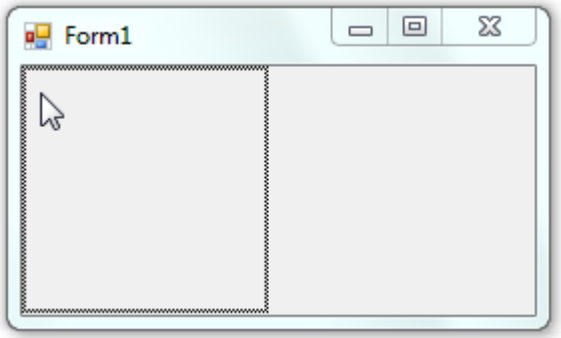
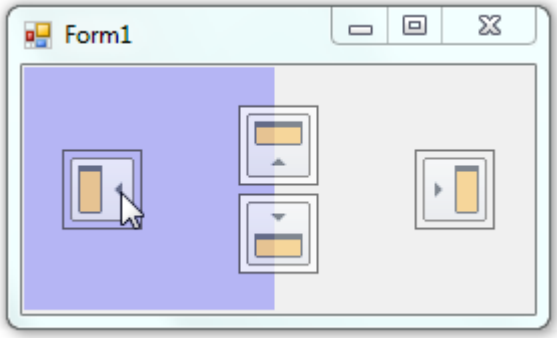
属性设置	图像
VisualStyle.Custom	[Custom允许你定制属于你自己的视觉样式]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	

VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

Docking 样式

当用户将C1DockingTab控制器加入到C1DockingTab组件中（请参阅 Enabling DockingTab Docking and Floating）后，它将变成一个对接控制器。你可以通过设置C1CommandDock组件的DockingStyle属性来为对接方式指定实现样式。DockingStyle属性包括：Default,VS2005,VS2008,和VS2010。Default设置显示一个灰色阴影轮廓用于指示当你释放光标时，控制器将对接的区域位置。然而VS2005,VS2008和VS2010将分别模仿 VisualStudio2005,VisualStudio2008,andVisualStudio2010中的对接样式。下面的表格中将为每一种对接样式提供一个示例。

Docking样式	示例

Default	
VS2005	
VS2008	
VS2010	

Tab样式

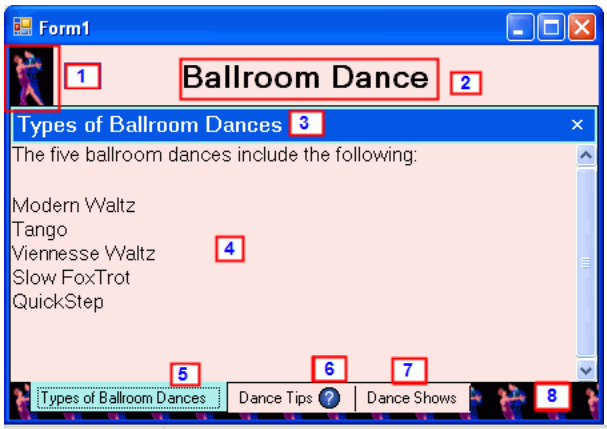
C1DockingTab控制器提供若干个内置的风格，例如Custom,System,Office2010Blue,Office2010Black,Office2010Silver,Office2007Blue,Office2007Black,Office2007Silver,Office2003Blue,Office2003Olive,Office2003Silver,OfficeXP,Classic和WindowsXP，使用VisualStyle属性，用户可以很轻松的应用这些风格。

C1DockingTab提供若干种内置风格，如Classic,Default,Office2003,Office2007,Office,2010,Rounded,Sloping,VS2003,和WindowsXP。使用TabStyle属性，用户可以很轻松的应用这些风格。你还可以为你的选项卡页面选择你喜欢的外观类型。例如，你也许想要选项卡页面显示图片，文本或者两者都包括。你可以通过TabStyle设置属性来完成这一工作。这些TabStyle属性包括Default, Text, Image和TextandImage。

你可能还想使用常用的C1DockingTab外观属性来定制你的选项卡样式，例如：C1DockingTab.BackgroundImage,C1DockingTab.BackgroundImageLayout,TabAreaBackColor,TabBackColor,TabBackColorSelected,和TabForeColorSelected。

下面的图片将介绍一部分C1DockingTab的常用外观属性：

图片1

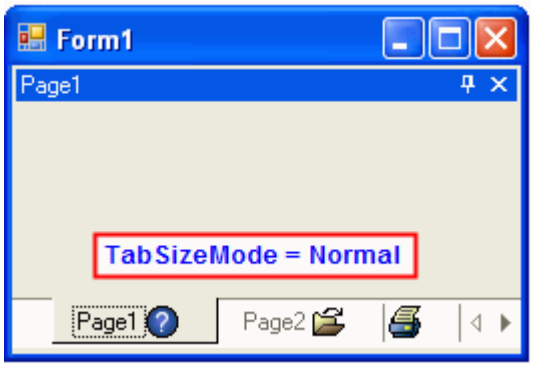


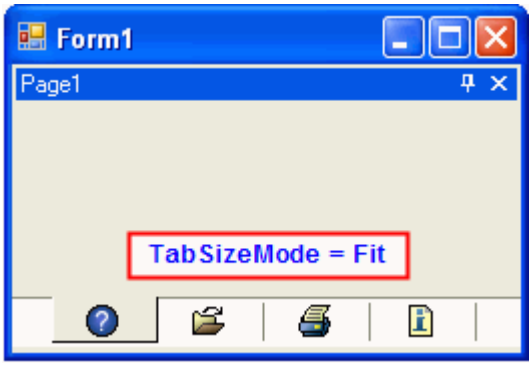
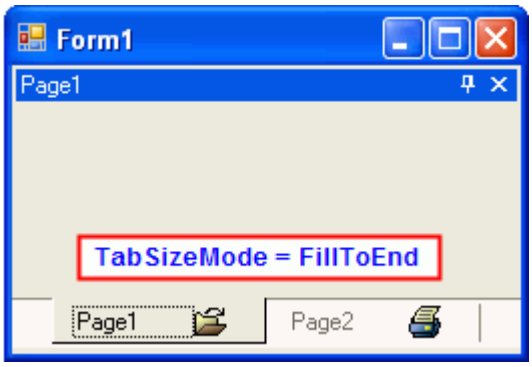
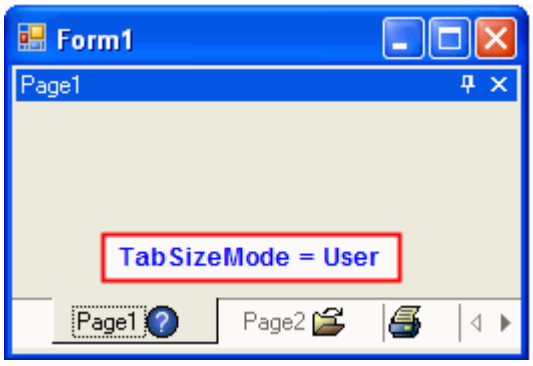
图片1标签	描述
1	PictureBox控件
2	Label 控件
3	C1DockingTabPage.CaptionText
4	RichTextBox 控件
5	C1DockingTabPage.TabBackColorSelected = PaleTurquoise
6	C1DockingTabPage.Image
7	C1DockingTabPage.TabBackColor = MistyRose
8	C1DockingTab.BackGroundImage

选项卡大小

你可以通过设置选项卡页面的ItemSize属性来控制页面的宽度和高度。这个属性将应用到所有的选项卡页面，无论页面中的内容包括什么。该属性的缺省设置根据页面内文本和图片的大小决定的。除了定制页面的高度和宽度，你还可以设置选项卡的模式。TabSizeMode属性包括四个参数类型：FillToEnd,Fit,Normal,和User。FillToEnd参数类型将延伸选项卡，使选项卡的宽度和整个选项卡控制器的宽度一致。Normal参数类型将使用缺省大小模式。User参数类型允许用户在MeasureTab事件中指定选项卡的大小。下面的表格将为你介绍每种TabSizeMode参数类型的实际效果。each TabSizeMode member.

注意:下面表格中的第三幅图片仅包括两个选项卡，用于更好的介绍FillToEnd参数类型的效果。

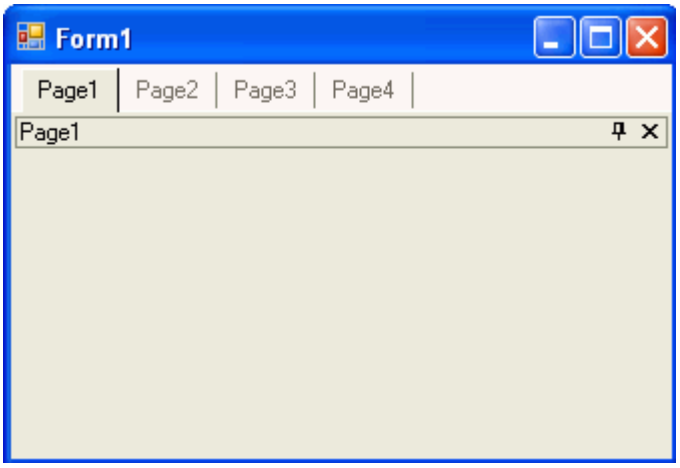
属性设置	图片
TabSizeMode.Normal	

TabSizeMode.Fit	
TabSizeMode.FillToEnd	
TabSizeMode.User	

选项卡方向

C1DockingTab提供各种选项卡方向的选项。用户可以将选项卡设置为顶部对齐，底部对齐，左侧对齐或者右侧对齐。用户可以使用Alignment属性来控制C1DockingTab控制器的选项卡方向。

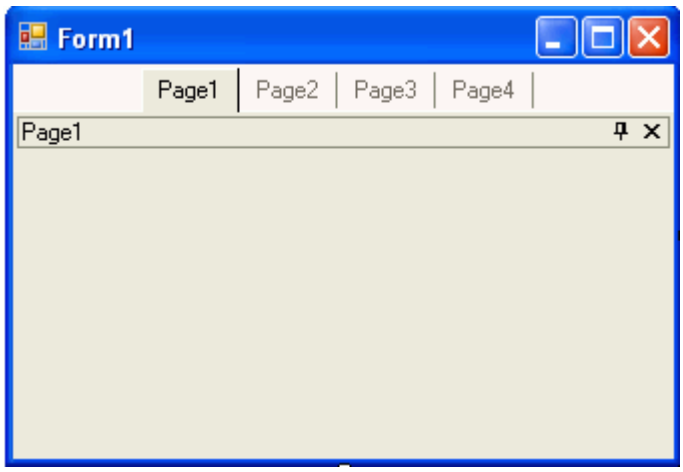
下面的图片显示C1DockingTab控制器中选项卡顶部对齐的效果。



除了选择选项卡的布局，你还可以指定选项卡如何对齐到页面内容区域的侧边。用户可以通过设置AlignTabs属性实现这一效果。AlignTabs属性设置选项包括：Near,Far,和Center。

注意:如果TabSizeMode属性设置为Fit，AlignTabs属性将不会起到任何效果。

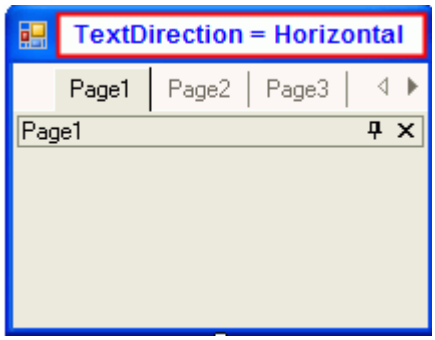
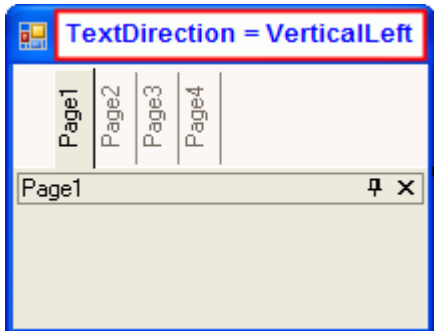
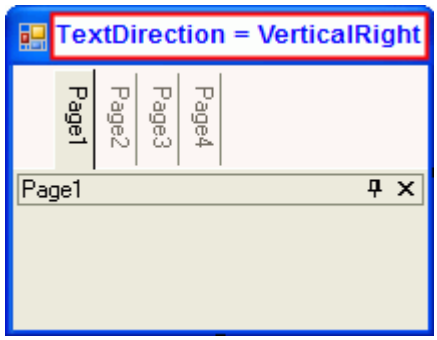
下面的图片显示C1DockingTab控制器中选项卡设置为顶部对齐并且居中的效果：



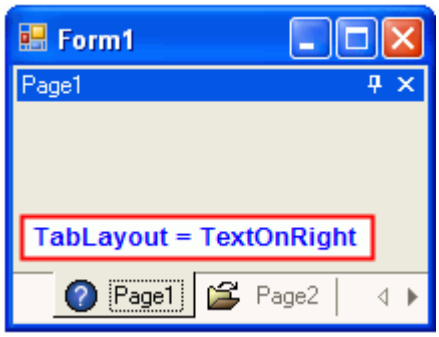
如果你希望你的选项卡放到一个指定位置，而不是通过控制器提供的Near，Far或者Center选项来控制。你可以通过设定Indent属性为一个指定像素数值来实现这一效果。
例如，下图中指定选项卡的缩进距离为30像素。

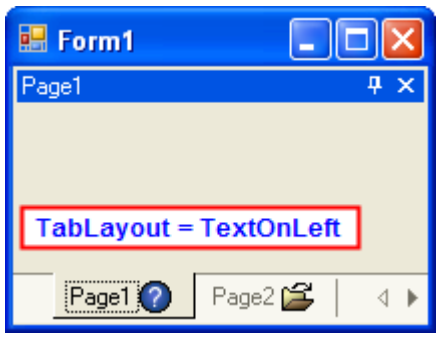
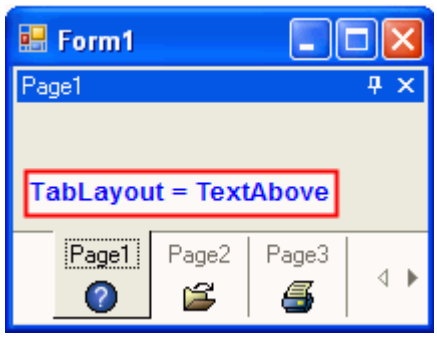
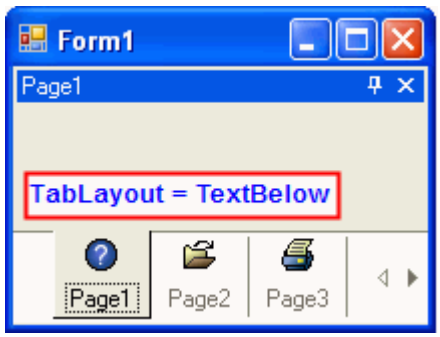
文字方向

除了决定选项卡的布局，你还可以定制选项卡中文本信息的布局。你可以使用TextDirection属性来横向，纵向，纵向从右到左，或者纵向从左到右的显示文本信息。下面的表格将为你介绍当TextDirection属性设置为Horizontal,VerticalLeft或者VerticalRight时，选项卡是如果显示的。

属性设置	图片
TextDirection.Horizontal	
TextDirection.VerticalLeft	
TextDirection.VerticalRight	

当你的选项卡中包含图片或者文字信息需要显示时，你也许希望控制文本显示在图片的上方，下方，左侧或者右侧。你可以使用TabLayout属性来设置选项卡中的文本显示在图片的上方，下方，左侧或者右侧。

属性设置	图片
TabLayout.TextOnRight	

TabLayout.TextonLeft	
TabLayout.TextAbove	
TabLayout.TextBelow	

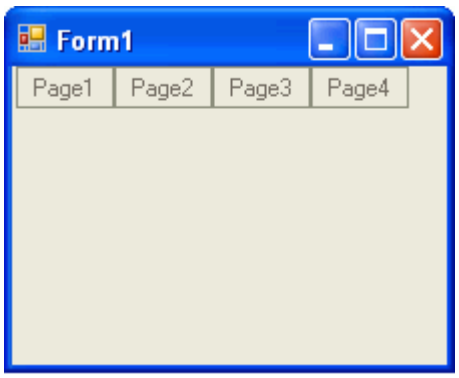
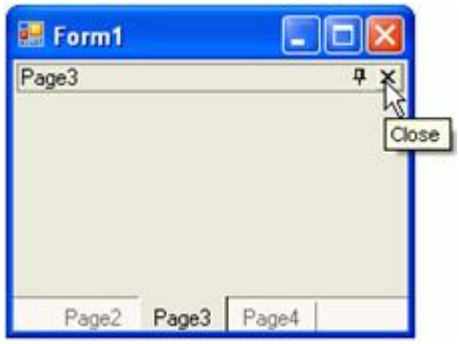
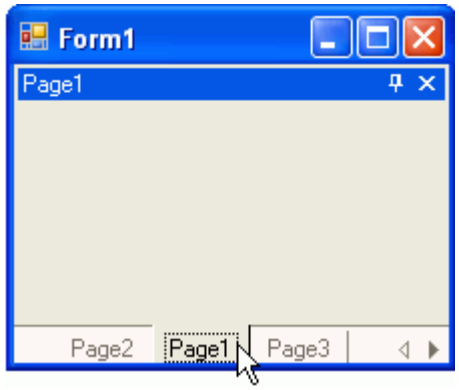
个别选项卡页面的隐藏，关闭以及移动

C1DockingTab控制器包括下面的几种行为属性，例如：隐藏，关闭以及移动选项卡页面。

- CanAutoHide
- CanCloseTabs
- CanMoveTabs

为了方便，你可以将这些属性都设置为缺省的True。如果你不希望让这些属性生效，你也可以将它们设置为False。下面的表格中将描述当CanAutoHide,CanCloseTabs,和CanMoveTabs属性设置为True时，C1DockingTab控制器外观是如何显示的。

动作	结果

AutoHide icon clicked	
Page1 Close button clicked	
Page1 tab moved	

选项卡页面中的鼠标悬停风格

C1DockingTab控制器提供一个简单属性，用于实现选项卡内鼠标悬停的效果。如果用户将HotTrack属性在设计或者编码阶段设置为True，当鼠标停留在选项卡上后，选项卡的外观将发生改变。

NavBar概述

C1NavBar用作将信息分为不同类型的分组，用来帮助快速地组织和浏览信息。它由一些预设的按钮表示的分组组成。每一个按钮具有一个标题，包含文本和图像，同时也具有一个用来向按钮分类添加信息的面板。同一时刻只能展开显示一个分类，其余分类仅按钮为可见，其关联的面板被隐藏。

导航栏外观和行为的属性

C1NavBar提供了许多有用的属性，用来控制C1NavBar的行为和外观。

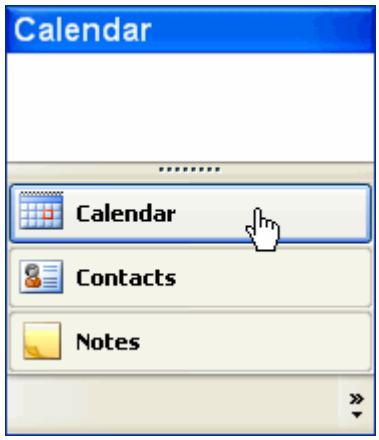
C1NavBar包括多种外观属性，用来可视地增强并定制化控件。控件的样式，大小，以及布局，可以很容易地通过C1NavBar的外观属性进行定制。这些属性可以在设计时通过属性窗口，或通过C1NavBar任务进行设置。此外，导航栏的外观可以通过编程方式使用C1NavBar 类进行设置。

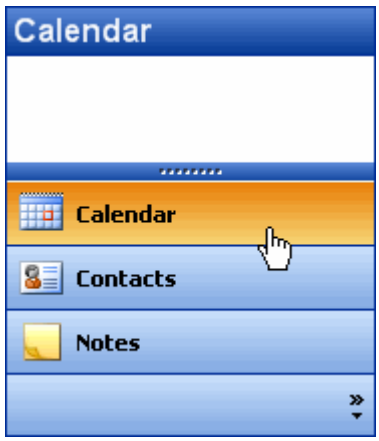
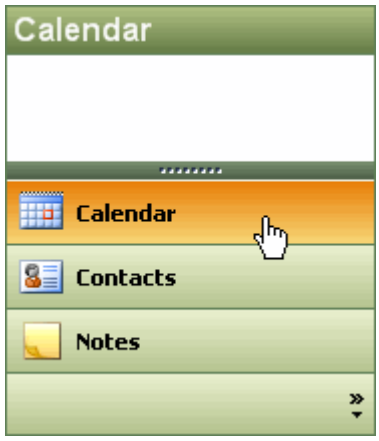
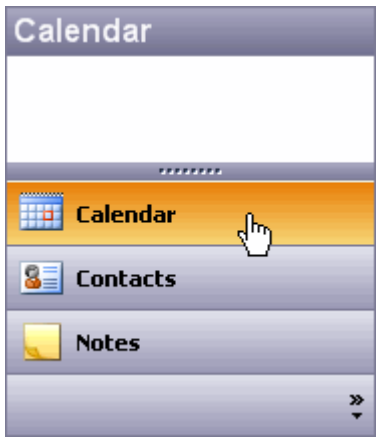
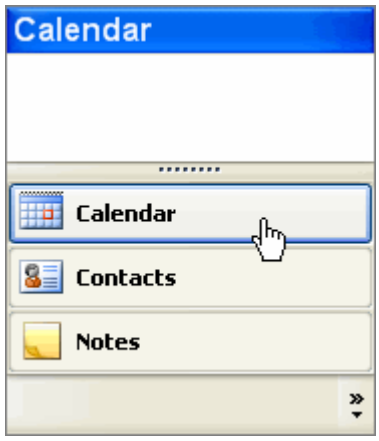
下面介绍一些C1NavBar控件的一些常见的外观和行为属性。

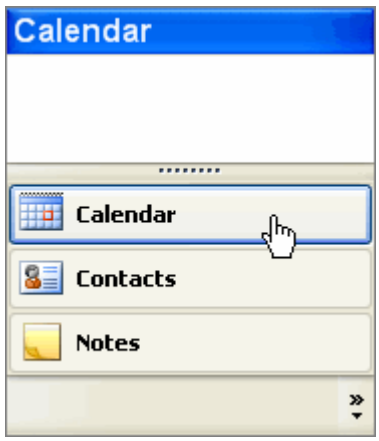
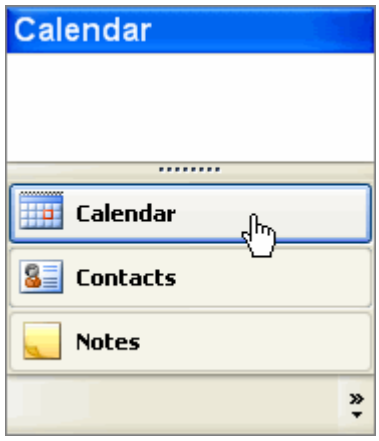
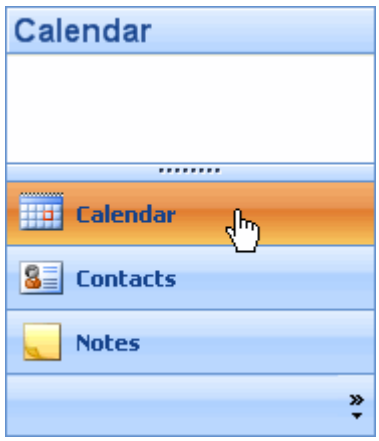
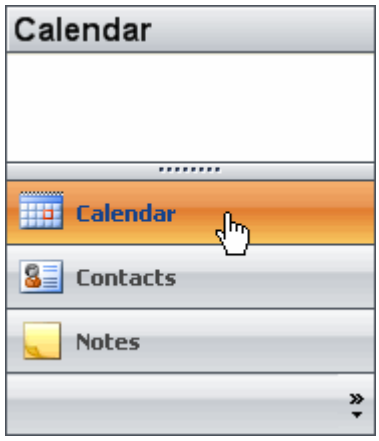
导航栏视觉样式

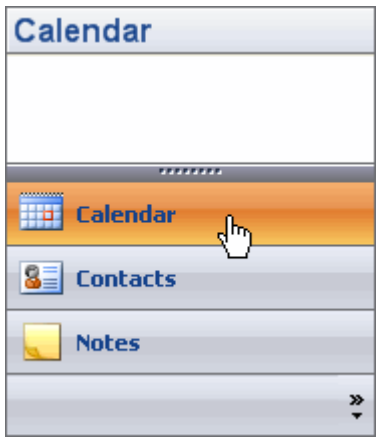
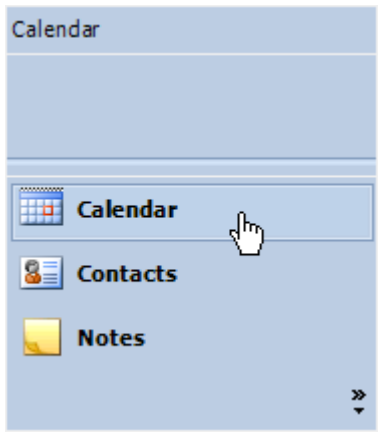
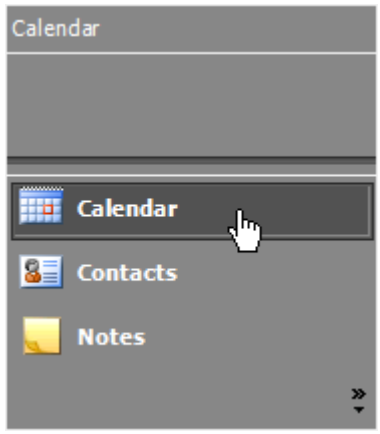
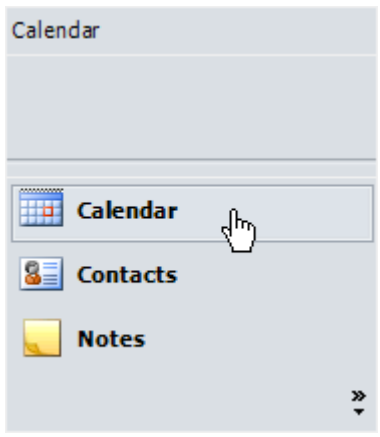
C1NavBar控件提供了一些内置的样式，比如说Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及 WindowsXP，这些样式可以容易地通过VisualStyle属性进行应用。

下表说明了C1NavBar 控件的每一种视觉样式。

属性设置	图像
VisualStyle.Custom	[Custom允许您自定义视觉样式。]
VisualStyle.System	

VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	

VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	

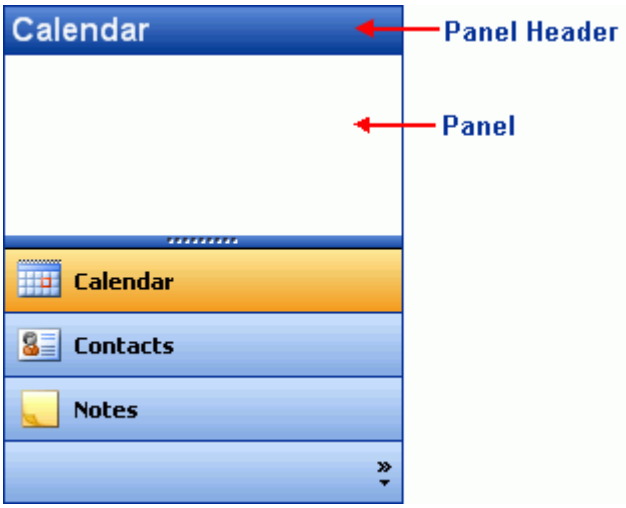
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

导航栏面板样式

导航栏面板标题

您可以在C1NavBarButton集合编辑器中使用PanelHeader属性以改变显示在预置按钮上的标题区域的默认文本。为了自定义标题的样式，您可以使用PanelHeaderFont属性。

面板标题的高度可以通过PanelHeaderHeight属性更改。



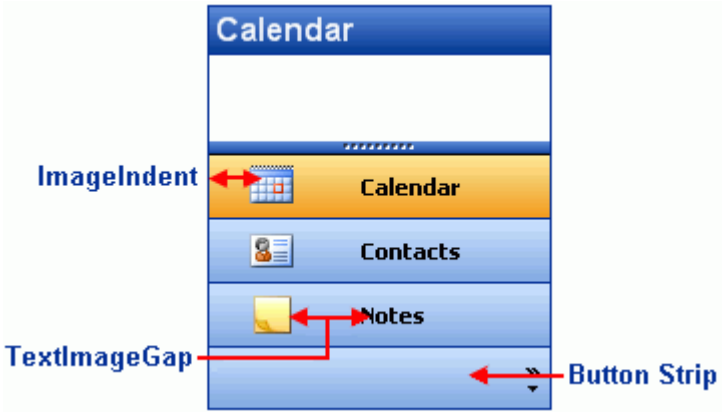
导航栏面板

你可以通过C1NavBarPanel 类定制C1NavBar 面板的外观属性，比如如背景色以及BackGroundImage。

导航栏按钮样式

C1NavBar由垂直堆叠的预置按钮或者自定义按钮组成。预置按钮包括以下类型：Custom，Mail，Calendar，Contacts，Tasks，Notes，Folder，Shortcut，以及Journal。每一个预置按钮包含图像和文本。默认情况下，图像显示在文字的左边。您可以设置将按钮的图像在C1NavBar底部的按钮栏水平显示。如果您打算在按钮栏显示很多按钮，那么您应当考虑增加其高度。您可以使用StripHeight属性增加按钮栏的高度。按钮栏的默认高度为30像素。

C1NavBar提供了多种外观属性用于自定义按钮的样式。下图说明了ImageIndent和TextImageGap属性：



按钮上的文字和图像的布局

您可以通过ImageIndent以及TextImageGap属性自定义每一个按钮文本和图像的布局，以获取期望的外观。如果您希望增加图像的缩进，可以使用ImageIndent属性。如果您希望文本和图像之家的间距更大，则可以使用TextImageGap属性增加二者的间距值，单位是以像素表示的宽度。默认情况下，ImageIndent属性的值为6个像素，而TextImageGap属性的值为3个像素。

导航栏文字竖排

如果其VisualStyle 属性设置为Office2010Black，Office2010Blue，或者 Office2010Silver文本可以在收起的C1NavBar 控件上垂直显示。下面的图像显示

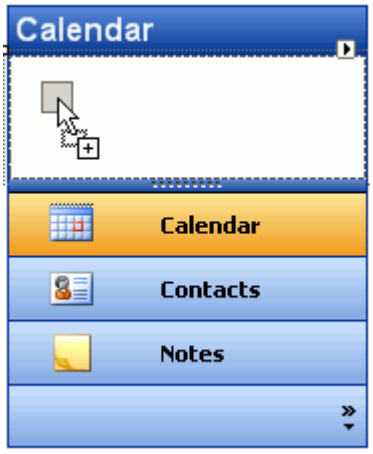
文本在C1NavBar控件垂直显示的示例。

为了达到这种效果，设置UIStrings CollapsedBarText 用户界面字符串枚举的文字为您希望当用户收起控件时，出现的文本。请注意，如果您希望使得控件在运行时可以收起，则您也必须设置控件的AllowCollapse 属性为True。

在导航栏面板嵌入控件

任意控件，比如说标签可以嵌入到C1NavBarPanel中。

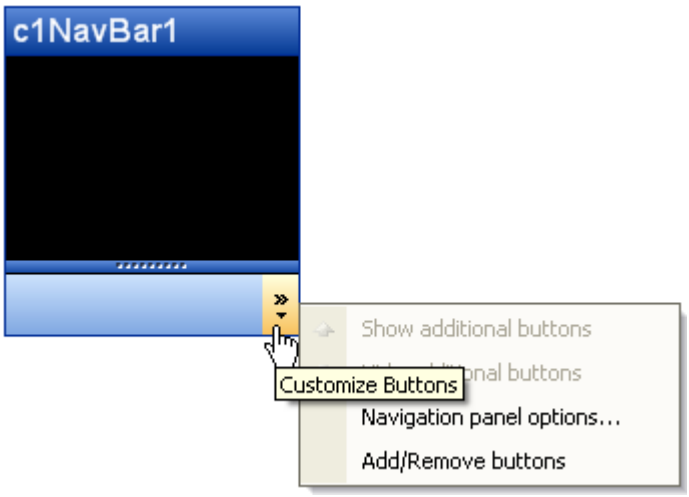
这可以简单地通过拖动任意一个控件到该面板上实现。下面的图像显示了一个正在被拖到面板上的标签控件：



运行时定制导航栏

C1NavBar的按钮可以在运行时通过单击下拉箭头按钮进行定制。

下面的弹出式菜单在运行时，当您单击下拉箭头时出现：



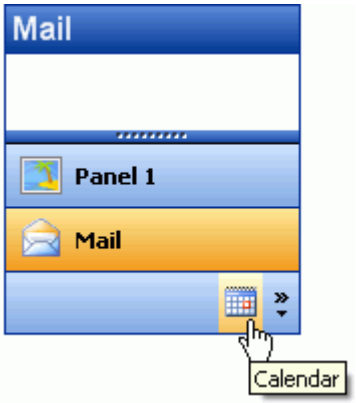
自定义按钮的菜单操作如下：

显示额外的按钮

从菜单中单击显示额外的按钮命令项将从任务栏区域移除该按钮，并将其放置在C1NavBar的面板区域。

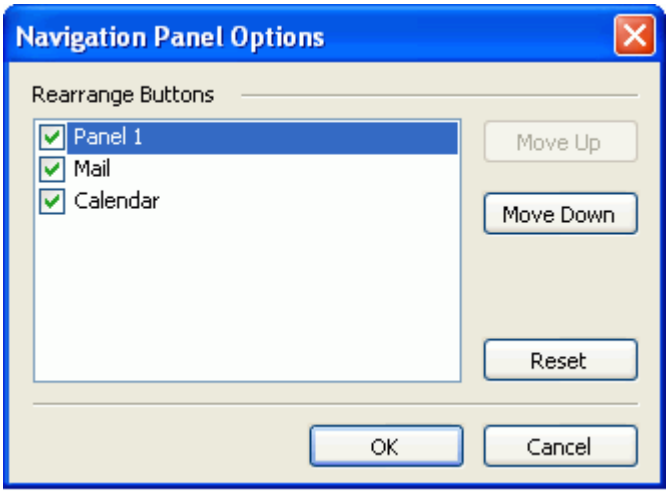
隐藏额外的按钮

单击隐藏额外的按钮从面板区域移除按钮并放置在任务栏区域。



导航面板选项

点击导航面板选项打开导航面板选项对话框。

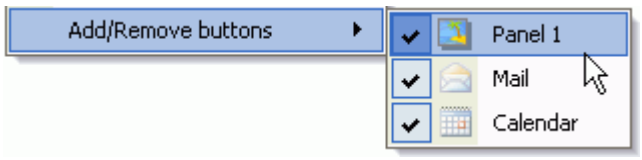


导航面板选项对话框包含以下命令按钮：

名称	描述
上移	将选定的按钮向上移动一个位置。
下移	将选定的按钮向下移动一个位置。
重置	重置按钮到其默认位置。
确定	保存更改并关闭导航面板选项对话框。
取消	取消对按钮所做的更该并关闭导航面板选项对话框。

添加/删除按钮

点击添加/删除按钮，打开一个具有可以显示在C1NavBar上的预置的按钮名称的菜单，并在每一个按钮的名称前伴随一个复选框，选中的状态表示显示在C1NavBar上的预置按钮。删除一个特定的按钮，点击您希望删除的按钮的名称旁边的复选框。



OutBar概述

C1OutBar是一个Outlook样式的容器/标签控件。它提供了一个页面的集合（页面为C1OutPage类型）。每一个页面包含一个标题栏和空白页，或者一个具有命令，比如说工具栏按钮的页面。在页面标题或者标题栏上单击将显示此页面，同时隐藏其余页面。每一页都以垂直堆叠的方式组织在一起。页面可以用来放置任意控件，但是当为每一页放置一个C1ToolBar时，将提供特殊处理（比如说智能滚动等等）。

以下主题提供关于C1OutBar控件外观和行为的更进一步的细节。

OutBar外观和行为属性

C1OutBar提供了许多有用的属性，用来控制标签项目的行为和外观。

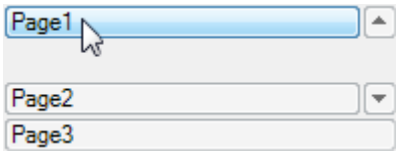
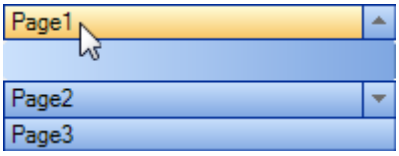
C1OutBar包括多种外观属性，用来可视地增强和定制控件。控件的页面样式，尺寸，以及布局可以容易地通过C1OutBar的外观属性进行定制。这些属性可以在设计时通过属性窗体或者编程方式进行修改。

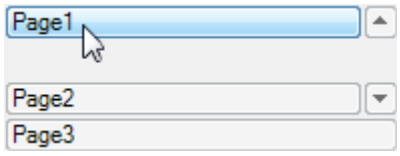
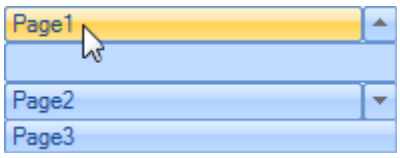
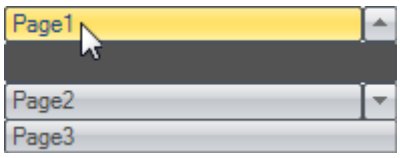
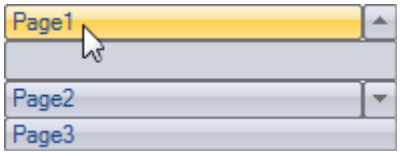
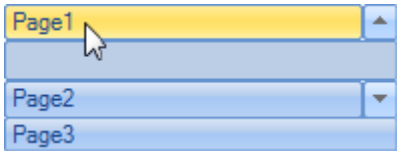
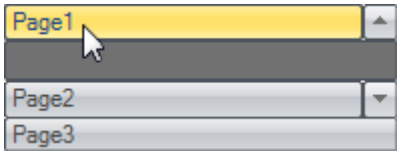
下面章节介绍一些C1OutBar控件常用的外观和行为属性。

OutBar视觉样式

C1OutBar控件提供了若干内置的样式，比如说Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP，可以通过VisualStyle属性容易地应用这些设置。

下表说明了C1OutBar控件的每一种视觉样式。

属性设置	图像
VisualStyle.Custom	[自定义允许您自定义视觉样式。]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	
VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	

VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	
VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

页面样式

C1OutBar提供了多种内置样式，比如说Custom，System，Office2007Blue，Office2007Black，Office2010Blue，Office2010Black，Office2010Silver，Office2007Silver，

Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP，您可以容易地通过VisualStyle属性应用这些设置。注意VisualStyle属性以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，它们现在已经被废弃不用。

您同样也可以选择您期望的标签页的外观。例如，您可能希望该按钮显示一个图像，文本或者二者组合显示。您可以通过设置ButtonLook属性的值为Default，Text，Image或者TextandImage实现不同的选择。

您同样也可以通过使用C1OutBar的通用外观属性，比如说BackColor，C1OutBar.BackgroundImage，C1OutBar.BackgroundImageLayout，以及BackHiColor自定义outbar的样式。

可以通过使用每一个C1OutPage上的C1OutPage.BackColor属性为每一个C1OutBar设置一个不同的背景颜色。

关于使用这些属性修改C1OutBar页面样式的更多信息，请参见Modifying the Appearance of the C1OutBar。

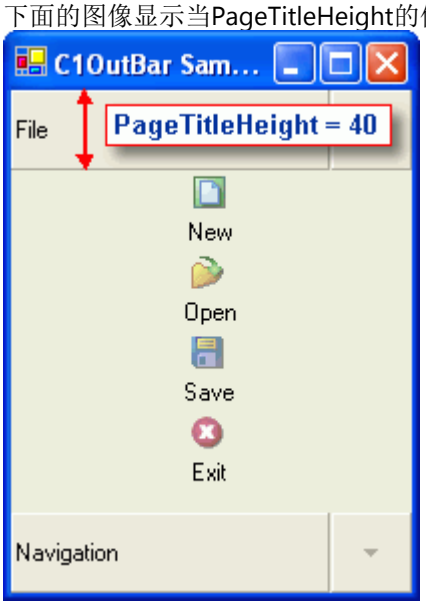
在页面中嵌入控件

任意控件，比如说文本框，可以嵌入到一个空白页（一个不包含工具栏的页面），或者一个带有工具栏的页面

页面尺寸

您可以通过设置PageTitleHeight属性为一个合适的值，以自定义C1OutPage的高度。这将使得全部的页面标题具有相同的高度。其默认值是零。

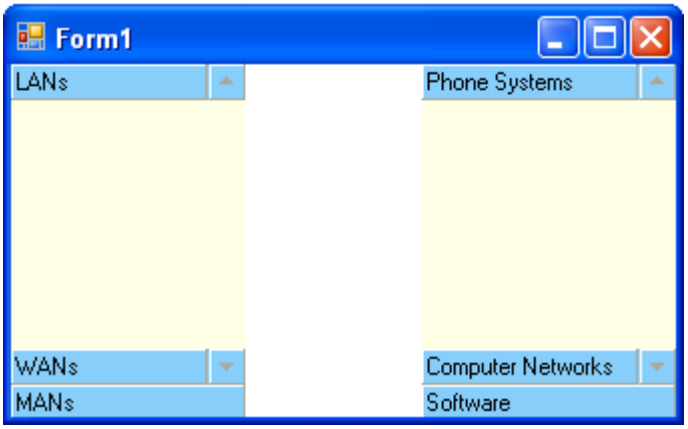
下面的图像显示当PageTitleHeight的值设置为40像素的结果。



页面布局和对齐

C1OutBar可以停靠到指定的C1CommandDock容器的顶部，左侧，底部或者右侧。

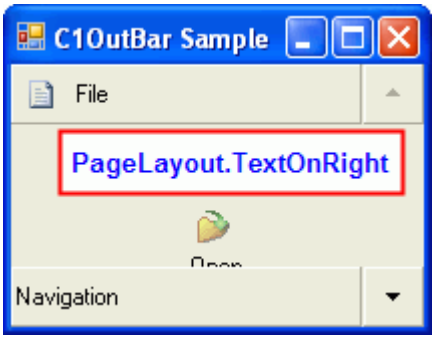
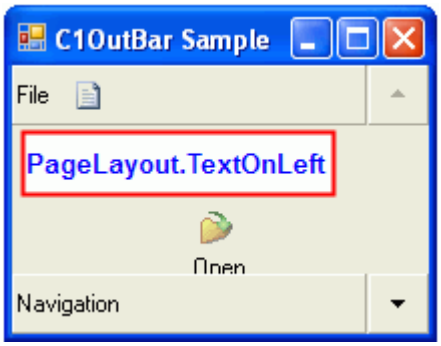
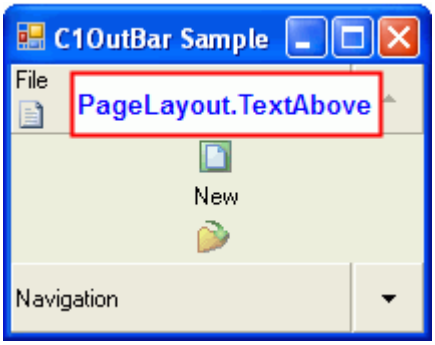
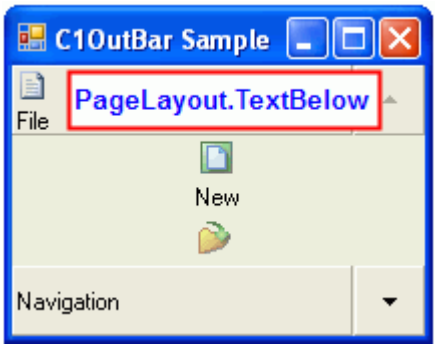
下面的图像显示了两个C1OutBar分别停靠在窗体的左右两侧的情况。每个C1OutBar停靠在C1CommandDock的内部。



C1OutBar提供了文本和图像在页面对齐的各种不同的选项。通过Align属性，页面中的图像或文本可以靠左侧，右侧或者居中在C1OutBar控件的title page/outpage上对齐显示。

当您同时在页面标题显示文本和图像时，您可能希望控制文本和图像显示的相对位置，选项可以为文本位于图像的上方，下方，左侧或者右侧。您可以使用PageLayout属性设置文本在页面标题中显示在图像的上方，下方，左侧或右侧。PageLayout属性的默认值为TextOnRight。

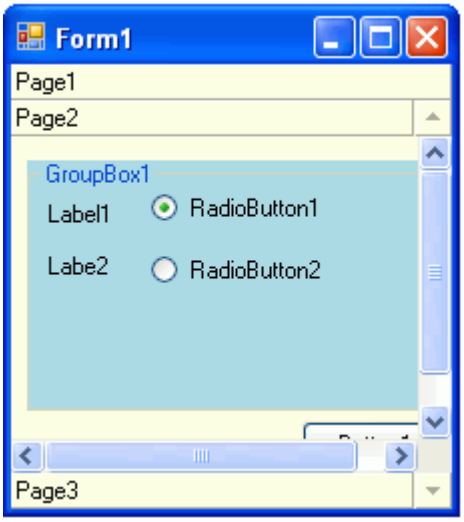
下表显示PageLayout属性的值。

属性设置	图像
PageLayout.TextOnRight	
PageLayout.TextonLeft	
PageLayout.TextAbove	
PageLayout.TextBelow	

滚动条

C1OutBar提供了一个C1OutPage.AutoScroll属性，以便在C1OutBar的特定页面启用滚动条。默认情况下，C1OutPage.AutoScroll属性为禁用状态。为启用滚动条，您可以设置C1OutPage.AutoScroll属性的值为True。

下面的图像显示C1OutBar的第二页显示滚动条的情况。



除了滚动条之外，C1OutBar还提供了一个ShowScrollButtons属性，用来启用显示在C1OutBar侧边的滚动条上面的按钮。默认情况下，该属性为启用状态。为禁用此属性，请设置ShowScrollButtons属性的值为False。

TopicBar概述

C1TopicBar表示一个主题栏。在C1OutBar中，控件提供了一个单页面的集合，组织为单一的分组，而C1TopicBar包含一组页面，组织为不同的分组。包含在C1TopicBar页面集合中的页面类型为C1TopicPage。每一页包括一个带有收起/展开图像以及由链接组成的内容区域的标题栏。每一页/分组可以有一个或多个链接。在分组上单击一次可以收起选中分组的链接项，再次单击则展开选中分组的链接项目。每个主题页面组织为垂直方向堆叠形式。

以下主题提供了关于C1TopicBar控件外观以及行为方面的进一步的细节。

TopicBar外观和行为属性

C1TopicBar提供了许多有用的属性来控制标签项目的行为和外观。

C1TopicBar包含多种外观属性，以可视地增强以及自定义控件。控件的页面样式，大小，以及布局，可以很容易地通过使用C1TopicBar的外观属性进行自定义。这些属性可以在设计时通过属性窗口或程序进行设置。

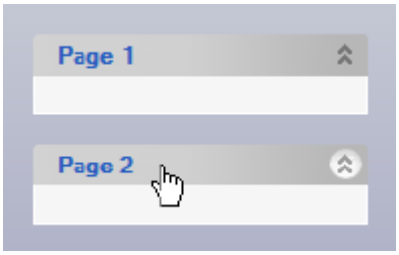
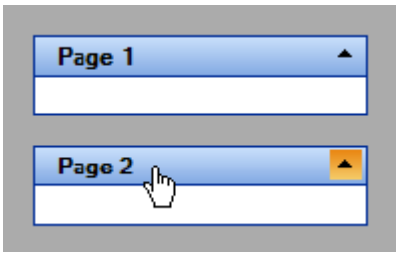
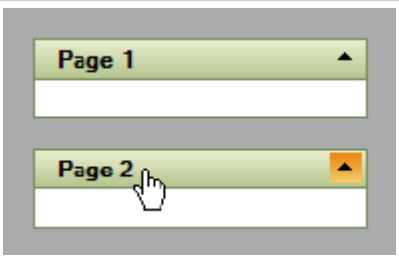
C1TopicBar包括若干有用的行为方面的属性，用来关闭标签页，重排标签，以及标签页的鼠标悬停效果，等等。

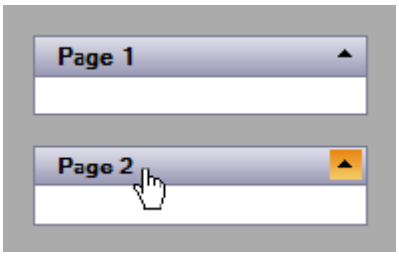
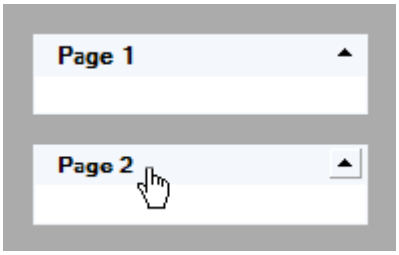
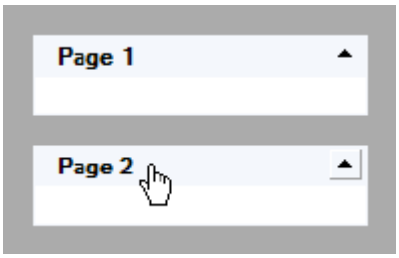
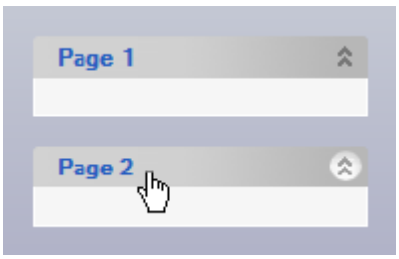
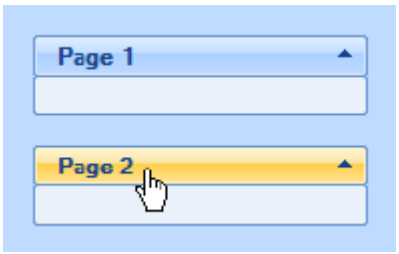
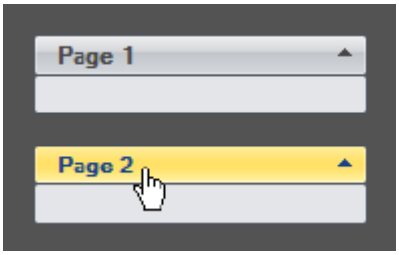
以下章节介绍一些常见C1TopicBar使用的外观和行为属性。

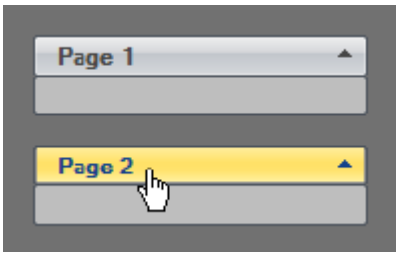
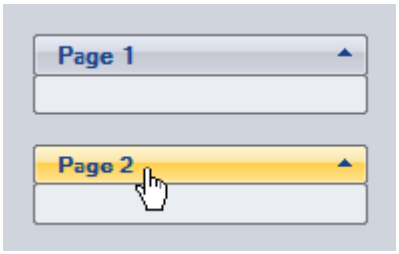
TopicBar视觉样式

C1TopicBar控件提供了一组内置的样式，比如Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP，可以非常容易地通过使用VisualStyle属性应用样式设置。

下表说明了C1TopicBar控件的每一种视觉样式：

属性设置	图像
VisualStyle.Custom	[**Custom** allows you to customize the visual style.]
VisualStyle.System	
VisualStyle.Office2003Blue	
VisualStyle.Office2003Olive	

VisualStyle.Office2003Silver	
VisualStyle.OfficeXP	
VisualStyle.Classic	
VisualStyle.WindowsXP	
VisualStyle.Office2007Blue	
VisualStyle.Office2007Black	
VisualStyle.Office2007Silver	

VisualStyle.Office2010Blue	
VisualStyle.Office2010Black	
VisualStyle.Office2010Silver	

可折叠以及可展开的主题页

您可以通过Collapsed属性指定页面为收起或展开状态。特定的页面可以收起或者展开，同时全部的页面都可以收起或者展开。

在设计时收起一个特定的页面

为了在设计时收起一个特定的页面，在C1TopicBar控件上选择期望的页面，并打开其C1TopicBar任务菜单以设置其Collapsed属性的值为True。

编程方式收起或展开全部的页面

为通过编程方式收起全部的页面，可以使用CollapseAll方法，或者也可以使用ExpandAll方法一次性展开全部页面。

TopicBar样式

C1TopicBar 提供了多种内置的样式，比如说Custom，System，Office2010Blue，Office2010Black，Office2010Silver，Office2007Blue，Office2007Black，Office2007Silver，Office2003Blue，Office2003Olive，Office2003Silver，OfficeXP，Classic，以及WindowsXP，可以容易地通过VisualStyle应用样式设置。注意VisualStyle属性以及VisualStyle枚举取代了LookAndFeel属性以及LookAndFeelEnum枚举，这两个接口已经被废弃不用。

您也可以应用一个特殊的样式至主题页，以便深色显示页面标题的背景色。为了应用特殊样式至主题页，请使用SpecialStyle属性。您也可以通过C1TopicBar的通用外观属性，比如说BackColor，BackgroundImage，BackgroundImageLayout，以及ForeColor，自定义C1TopicBar的外观。

TopicBar动画

主题页在收起或展开时可以具有动画效果。请使用Animation属性以添加动画效果至主题页。

TopicPage布局和对齐方式

C1TopicBar可以停靠到指定的C1CommandDock容器的
顶部，左侧，右侧或底部。

C1TopicBar提供了文本在页面中对齐的各种不同的选项。通过Align属性，页面中的文本可以在C1TopicBar控件标题栏中居左，居右或居中对齐。当您在主题页中显示一个图像时，默认情况下，文本出现在图像之后。

TopicBar中的工具提示

主题栏上为您提供了一个TooltipText属性，用来创建一个用户友好的应用程序。例如，您可以为每一个主题页添加工具提示，以便向最终用户提供关于此主题页的更多信息。工具提示用来当鼠标悬停在某个控件上方时，显示一段文本。

RadialMenu概述

C1RadialMenu是一个和C1ContextMenu相似的组件，唯一的不同是它的命令和按钮是环形排列，通过按钮访问，当你点击表单或者可触控的设备，轻敲某一项目的时候会弹出窗口 。C1RadialMenu支持鼠标，键盘，触屏触摸的交互。

C1RadialMenu显示为一组切片样式的环形菜单。径向菜单的中心是一个主要的按钮被称为内半径按钮。每一个切片都能连接另一个径向菜单。如果有很多子菜单项目的话，将会有箭头按钮显示在每一切片边框。点击箭头将会显示其他的径向菜单，并带有更多表示另外子菜单的切片，

因为径向菜单就像上下文菜单和弹出菜单，仅当使用C1RadialMenu.ShowMenu方法决定在菜单上怎样显示C1RadialMenu时才会显示。通过只在需要时显示C1RadialMenu能够防止视觉注意力和记忆过载。

下面的代码示例展示了在向表单添加组件后显示C1RadialMenu。你也能够调用参数来展开显示C1RadialMenu或者新建一个方法计算中心。

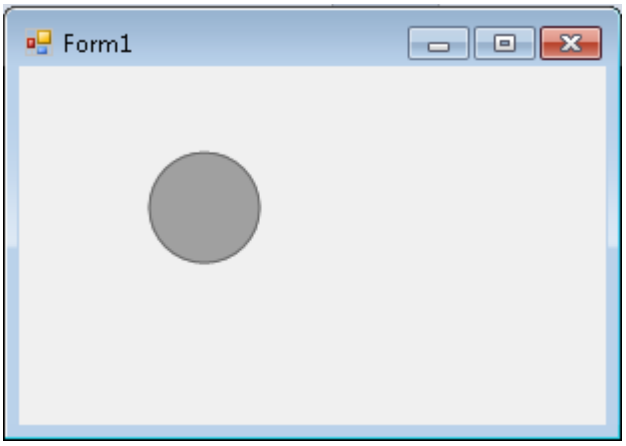
C#

```
c1RadialMenu1.ShowMenu(this, new Point(350, 350));
```

Visual Basic

```
c1RadialMenu1.ShowMenu(Me, New Point(350, 350))
```

当运行应用程序时，C1RadialMenu以一个中心按钮显示，因为还没有添加C1RadialMenuItems 或C1RadialCommandMenuItems:



下面的图片展示了带有一些RadialMenuItems和sub RadialMenuItems的C1RadialMenu。每个添加到C1RadialMenu 的RadialMenuItem显示为一个饼状切片，当点击或者轻敲饼状切片时会打开一个新的带有subRadialMenuItems的环形C1RadialMenu。

RadialMenu外观和行为

下面章节介绍了C1RadialMenu控件的一些常用外观和行为属性。

RadialMenu 动画

当打开或者关闭径向菜单时有动画效果。请使用**UseAnimation**属性在主题页上添加动画效果。

RadialMenu工具提示

径向菜单项目和命令项目具有RadialMenuItem.ToolTip和 RadialMenuCommandItem.ToolTip属性用来创建用户友好的应用程序。例如，能够为每个菜单项目页添加工具提示以便为用户提供更多关于项目和命令的信息。当鼠标滑过项目时，工具提示可以显示文本。

C1RadialMenu.TooltipPostion属性用来指定关联到径向菜单的工具提示文本的位置。你可以选择Left, Top, Right, Bottom, 或者 None（如果你不想要指定位置）。

半径和内半径属性

ComponentOne径向菜单像有内半径的圆环图。内半径指定了半径分割区域外面填充为项目背景颜色，而内部区域填充为径向菜单自身背景。内半径的大小能够使用C1RadialMenu.InnerRadius属性调整。径向菜单中间的圆圈时中心按钮。中心按钮的默认大小是28，可以使用C1RadialMenu.ButtonRadius属性调整。

隐藏C1Radial菜单

C1RadialMenu当失去焦点时会自动隐藏。可以把C1RadialMenu.AutoHide属性设为False实现C1RadialMenu一直显示
�?/P>

减小中心按钮尺寸

中心按钮的默认大小是28，如下所示：

把中心按钮的默认大小从28减为10，可以使用如下代码：

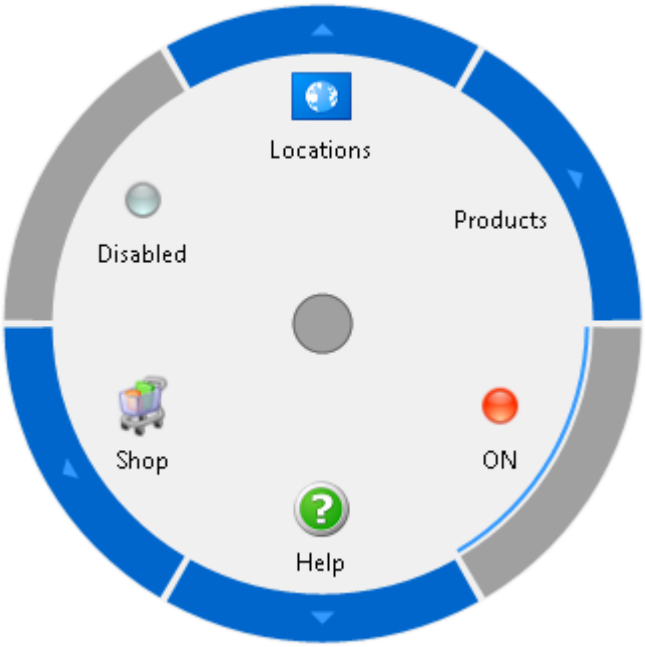
C#

```
c1RadialMenu1.ButtonRadius = 15;
```

Visual Basic

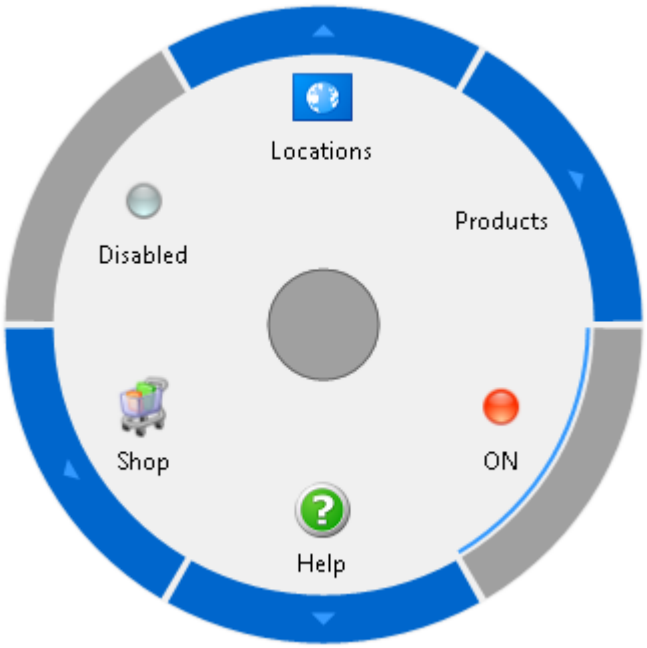
```
c1RadialMenu1.ButtonRadius = 15
```

当ButtonRadius大小从那个28减为15后，外观如下所示：



C1RadialMenu指南

此章节提供创建基本径向图标的步骤。通过此指南你将会学到怎么使用Show方法在表单上显示径向菜单和怎么添加RadialMenuItems 和子RadialMenuItems。一旦完成下面的步骤，径向菜单会如下所示：



C1RadialMenu指南步骤1：添加第一个菜单和子菜单项目

通过使用表哦但设计器或者编程，C1RadialMenu组件能够被添加到表单。无论使用设计器或编程，都需要调用Show方法在运行应用程序时显示C1RadialMenu。

完成下面步骤向C1Radial添加RadialMenuItems：

1. 从工具箱添加C1RadialMenu组件到表单上
2. 添加如下代码调用C1Radial.ShowMenu方法在运行应用程序时制作C1RadialMenu

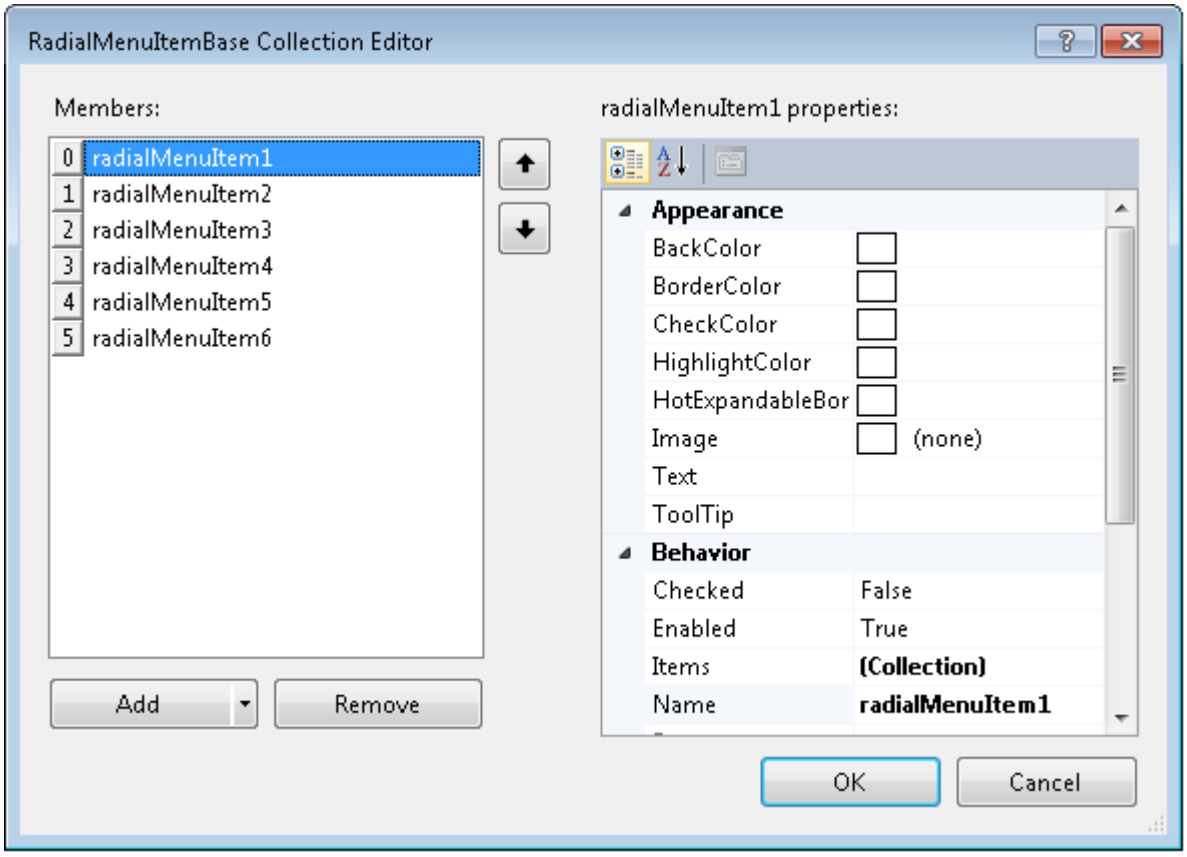
C#

```
c1RadialMenu1.ShowMenu(this, new Point(350, 350));
```

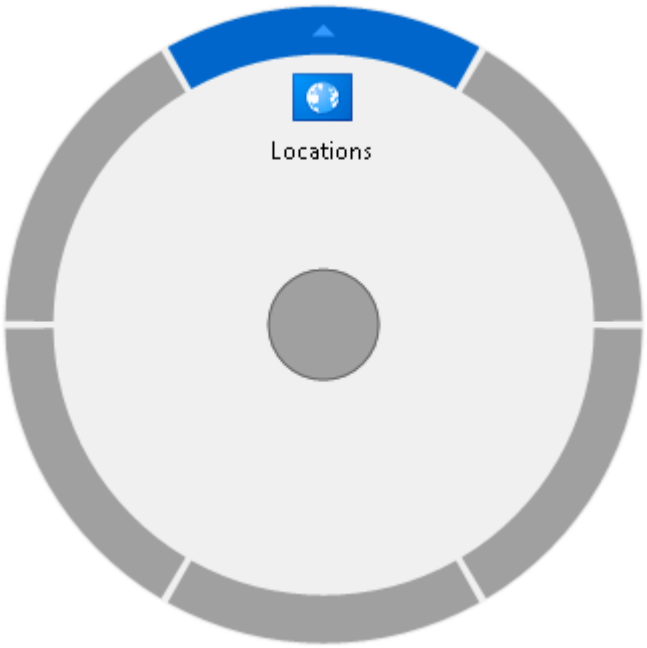
Visual Basic

```
c1RadialMenu1.ShowMenu(Me, New Point(350, 350))
```

4. 右击C1RadialMenu组件选择Properties。
5. 点击Items属性旁边的省略号按钮。RadialMenuItemBase集合编辑器打开。
6. 点击Add下拉框选择RadialMenuItem。重复此步骤5次这样将会由6个RadialMenuItem。



7. 选择**radialMenuItem1** 设置**Text**属性为**Locations, ToolTip** 为**International and local C1 sites**, **Name**为**rmiLocale**。
8. 选择 **Image** 属性旁边的省略号按钮，**Select Resource** 对话框将会出现。
9. 点击 **Import**按钮定位想要添加到工程资源文件里的图片文件。
10. **在工程资源文件中：** 下拉框选中图片比如 **flag_generic**, 然后点击**OK**.
11. 选择 **rmiLocale** 并点击**Items** 属性旁边的省略号按钮。**RadialMenuItemBase Collection Editor**出现。
12. 点击**Add** 下拉框按钮和RadialMenuItem。重复6次。这些菜单项是radialMenuItem1的子菜单。
13. 选择radialMenuItem7 并设置其ToolTip为USA and International，Name为rmiUS.
14. 选择**Image** 属性旁边的省略号按钮。**Select Resource** 对话框出现。
15. **在工程资源文件中：** 下拉框选择图片比如, **flag_usa**, 然后点击**OK**.
16. 选中 **RadialMenuItem1** 设置它的 **ToolTip**属性为**Disabled item** ， **Enabled** 属性为 False.
17. 选择第三个菜单项**radialMenuItem8**, 设置它的**ToolTip**属性为**China**, **Name**为 **rmiChina**, **UserData**为**locale**.
18. 选择**Image** 属性旁边的省略号按钮 。**Select Resource** 对话框出现。
19. 在工程资源文件中：下拉框选择图片，比如 **flag_china**,然后点击OK。
20. 选中 **radialMenuItem9** 设置其**ToolTip** 属性为**Japan**, **Name** 属性为 **rmiJapan**, **UserData** 属性为**locale**.
21. 选择Image 属性旁边的省略号按钮 。**Select Resource** 对话框出现。
22. 在工程资源文件中：下拉框选择图片，比如 **flag_japan**, 然后点击OK。
23. 选中**radialMenuItem10** 设置其**ToolTip** 属性为**India**, **Name** 属性为**rmiIndia**, **UserData** 属性为**locale**.
24. 选择Image 属性旁边的省略号按钮 。Select Resource 对话框出现。
25. 在工程资源文件中：下拉框选择图片，比如 **flag_india**, 然后点击OK。
26. 选中 **radialMenuItem11** 设置其ToolTip 属性为 **South Korea**, **Name 属性为 rmiSouthKorea**, **UserData**属性为**locale**.
27. 选择Image 属性旁边的省略号按钮 。Select Resource 对话框出现。
28. 在工程资源文件中：下拉框选择图片，比如 **flag_south_korea**, 然后点击OK。
29. 选中**radialMenuItem12**设置其ToolTip 属性为**Disabled**，**Enabled** 属性为**False**.
30. 点击**OK**保存并关闭**RadialMenuItemBase Collection Editor**.
31. 运行工程观察：



点击或者轻敲箭头和别的径向菜单将会有子RadialMenuItems 出现。

你可以点击中心返回按钮箭头导航回RadialMenuItem 之前的**Locations**。

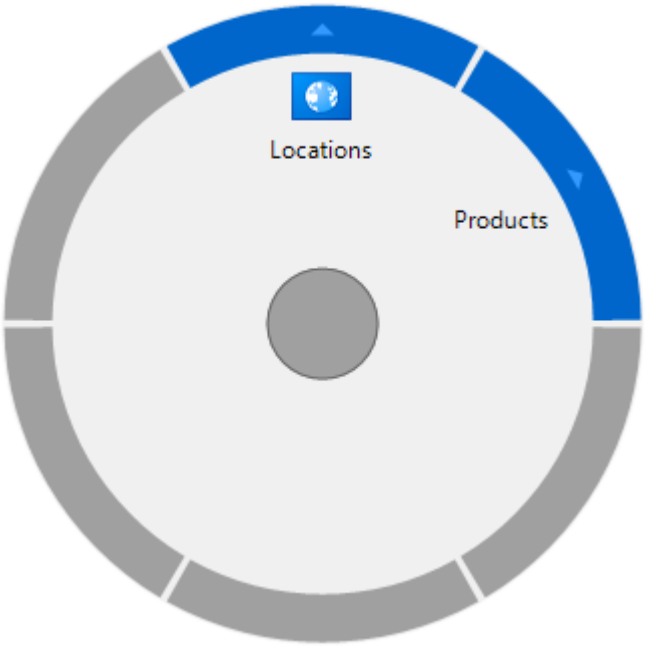
C1RadialMenu指南步骤2：添加第二个菜单和子菜单项目

这一步，你将会向已存在的Products菜单添加子菜单项。

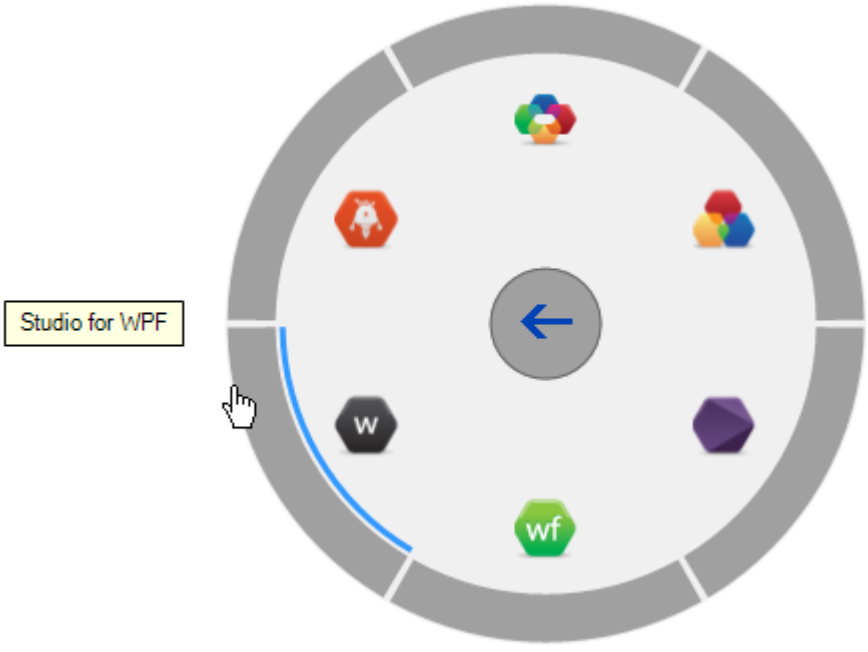
1. 在**RadialMenu**属性窗口点击**Items**属性旁边的省略号按钮。**RadialMenuBaseItem**集合编辑器出现。
2. 从**Members**列表中选中**radialMenuItem2**，设置其**Name**属性为 **rmiProducts**，Text属性为**Products**。
3. 点击**Items**属性旁边的省略号按钮。**RadialMenuBaseItem** 集合编辑器打开。
4. 点击**Add**下拉框按钮选中**RadialMenuItem**。重复此步骤直到有6个RadialMenuItem。
5. 选中第一个成员**radialMenuItem2**，设置其**Image**属性为ultLogo_32，**ToolTip** 属性为Studio Ultimate, **Name** 属性为rmiStuUlt。
6. 选中第二个成员**radialMenuItem7**，设置其**Image**属性为seLogo_32, **ToolTip** 属性为Studio

Enterprise，**Name**属性为rmiStuEnt.

7. 选中第三个成员**radialMenuItem8**，设置其**Image**属性为c1powersuite_logo_32, **ToolTip** 属性为Enterprise-ready reporting and spreadsheets controls for .NET applications, **Name** 属性为rmiStuPower.
8. 选中第四个成员**radialMenuItem9**，设置其**Image**属性为winformsLogo_321, **Tooltip** 属性为Studio for WinForms, **Name** 属性为rmiStuWinForms.
9. 选中第五个成员**radialMenuItem10**，设置其**Image**属性为wpfLogo_32, **ToolTip** 属性为Studio for WPF, **Name** 属性为rmiStuWPF.
10. 选中第六个成员**radialMenuItem11**，设置其**Image**属性为aspLogo_32, **ToolTip** 属性为Studio for ASP.NET Wijmo，**Name**属性为rmiStuWijmo.
11. 点击**OK**保存并关闭**RadialMenuItemBase** 集合编辑器。
12. 点击**OK**保存并关闭**RadialMenuItemBase** 集合编辑器。
13. 运行工程如下所示：



注意第二个RadialMenuItem, Products已经被添加。点击 **Products** 菜单，其子菜单项目将会如下所示：

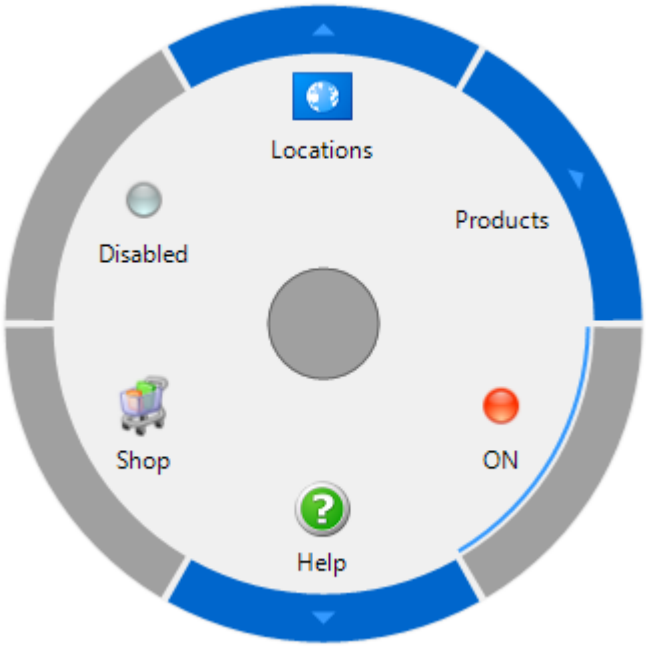


注意当鼠标悬停在菜单项时，工具提示将会显示。点击返回按钮可以回到主径向菜单。

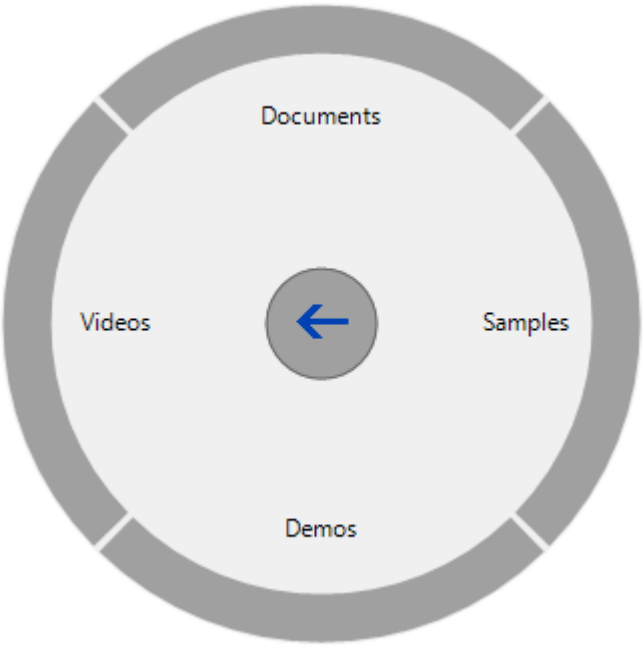
C1RadialMenu指南步骤3：添加剩下的RadialMenuItems

这一步，向已存在的Products菜单添加子菜单项。

1. 在**C1RadialMenu**属性窗口点击Items属性旁边的省略号按钮。**RadialMenuBaseItem**集合编辑器打开。
2. 从**Members list**中选择radialMenuItem3，设置其**Image**属性为bullet_ball_glass_red，**Text**属性为ON，**Checked**属性为True，**Name**属性为rmiCheck，**UserData**属性为Check。
3. 从**Members list**中选择radialMenuItem4，设置其**Image**属性为Help，**Text** 属性为Help, **ToolTip** 属性为Get help, **Name** 属性为rmiHelp.
4. 点击**Items**属性旁边的省略号按钮。**RadialMenuBaseItem**集合编辑器打开。
5. 点击**Add**下拉框按钮选中**RadialMenuItem**。重复此步骤直到有了4个RadialMenuItem。
6. 从**Members list**中选择**radialMenuItem2**，设置其**Text** 属性为Documents **Name** 属性为rmiDocs.
7. 从**Members list**中选择**radialMenuItem3**，设置其**Text** 属性为Samples，**Name**属性为rmiSamples。
8. 从**Members list**中选择**radialMenuItem4**，设置其**Text** 属性为Demo，**Name** 属性为rmiDemos.
9. 从**Members list**中选择**radialMenuItem5**，设置其**Text** 属性为Demos，**Name** 属性为rmiDemos.
10. 从**Members list**中选择**radialMenuItem6**，设置其**Text** 属性为Videos **Name** 属性为rmiVideos.
11. 点击**OK**保存并关闭**RadialMenuItemBase**集合编辑器。
12. 在**RadialMenuItemBase**集合编辑器中从**Members list**中选择radialMenuItem5，设置其**Image** 属性为shoppingcart_full1, **Text** 属性为Shop, **ToolTip**属性为Shop C1, **Enabled** 属性为False, **Name** 属性为rmiStore.
13. 从**Members list**中选择radialMenuItem6，**Image** 属性为bullet_ball_glass_grey, **Text** 属性为Disabled, **ToolTip** 属性为Disabled Item, **Enabled** 属性为False.
14. 点击**OK**保存并关闭RadialMenuItemBase集合编辑器。
15. 运行工程如下所示：



禁用的菜单以灰色边框显示。勾选的菜单，ON，灰边框内侧显示为蓝色的细边框。
点击**Help**按钮打开一个新的径向菜单，并带有Help菜单的子菜单项目。


Documents
Videos
Samples
Demos

WinForms 菜单和工具栏示例

请注意，本ComponentOne软件工具带有各种示例工程和/或Demo，这些工程或者Demo可能使用到了包含在ComponentOne Studio套件中的其他开发工具。

您可以通过ComponentOne 示例 Explorer访问各种示例。为查看示例，单击"开始"按钮，然后单击ComponentOne | Studio for WinForms | 示例 | Menus and Toolbars 示例。

点击以下链接中的任意一个查看C1Command示例的列表：

▶ Visual Basic 示例

示例	描述
CreateMenusInCode	演示如何通过代码创建并设置上下文菜单。完全通过代码为一个文本框创建一个C1ContextMenu，窗体上没有任何其他的C1Command元素。为菜单项挂接单击事件的事件处理程序。同时为命令设置了状态查询处理函数，因此他们的状态（启用等等。）将由C1Command自动保持更新。此示例调用C1.Win.C1Command命名空间。
ProgramOutBar	此示例从一个包含两个空白页的C1OurBar的窗体入手（在设计时创建）。该窗体同时也包含一个空白的CommandHolder。在窗体加载的处理程序中，在此CommandHolder上创建了若干命令。两个工具栏也创建并向命令填充了链接。工具栏放置在OutBar页面上。
SimpleMenusInCode	此示例演示如何通过定制其的外观并附加事件处理程序，在代码中创建菜单和工具栏。该示例应用程序是一个具有富文本编辑器控件的窗体。Form Load事件处理程序创建C1Command命令来加载一个文本文件，创建一个新的窗体，或退出应用程序。它还创建了一个主菜单和一个工具栏允许用户调用这些命令。
SimpleTextEditor	显示C1Command菜单和工具栏的基本用法。建立了基于富文本编辑文本框的文本编辑器，并包含了C1Command菜单和浮动工具栏。工具栏可以被最终用户进行定制。它们的布局和自定义的设置将自动由C1Command保存在应用程序的配置文件中。此示例使用C1ToolBar，C1ContextMenu，C1CommandMenu，C1Command，C1MainMenu，C1CommandLink，C1CommandDock，以及C1CommandHolder控件。
SimpleTextEditor2	演示如何添加任意控件至一个工具栏项。该示例和SimpleTextEditor示例类似，但是额外地包含了一个组合框，该组合框嵌入在一个可移动且可停靠的C1ToolBar中，实现了一个简单的文本搜索功能。此示例使用C1ToolBar，C1CommandMenu，C1CommandControl，C1CommandDock，C1CommandHolder，C1CommandLink，C1MainMenu，C1Command，以及C1ContextMenu控件。

▶ C# 示例

示例	描述
CreateMenusInCode	演示如何通过代码创建并设置上下文菜单。完全通过代码为一个文本框创建一个C1ContextMenu，在此窗体上没有任何其它的C1Command元素。为菜单项挂接单击事件的事件处理程序。同时为命令设置了状态查询处理函数，因此他们的状态（启用等等。）将由C1Command自动保持更新。此示例调用C1.Win.C1Command命名空间。
MdiTabs	此示例演示如何使用C1DockingTab创建一个类似于VisualStudio的多窗口环境。
NonMdiMenuMerge	演示如何使用C1Command提供的方法合并菜单。本示例包含两个窗体，每一个窗体都具有一个C1MainMenu。一个窗体是在应用程序启动时创建的，它是应用程序的主

	窗体。它包含一个命令创建其他窗体。当该窗体被命令创建时，其主菜单将通过编程方式合并到第一个窗体的主菜单上，这和MDI字窗体合并到Windows的方式完全一样。此示例使用C1CommandLink，C1MainMenu，C1Command，C1CommandMenu，以及C1CommandHolder控件。
SelectMdiChildForm	该示例展示了如何从选择器窗口提供一个定制的MDI子窗体。该示例工程包含一个继承的窗体，基于C1SelectMdiChildForm类型。自定义选择器形式重写默认的颜色，除了两个默认的按钮（确认和取消）之外，还提供了一个新的关闭按钮，可以选择需要关闭的窗体。此示例还演示如何从代码调用选择器窗体。
SimpleMenusInCode	此示例演示如何通过定制其的外观并附加事件处理程序，在代码中创建菜单和工具栏。该示例应用程序是一个具有富文本编辑器控件的窗体。Form Load事件处理程序创建C1Command命令来加载一个文本文件，创建一个新的窗体，或退出应用程序。它还创建了一个主菜单和一个工具栏允许用户调用这些命令。

WinForms菜单和工具栏基于任务的帮助

基于任务的帮助章节，假定您已经熟悉在Visual Studio等环境中进行编程，
并且知道如何用一般的控件。

每一个主题通过使用WinForms菜单和任务栏产品，为特定的任务提供解决方案。

每个基于任务的帮助主题还假设您已经创建了一个新的.NET工程。

菜单任务

此章节介绍怎么执行特定的菜单任务。

向C1MainMenu添加一个菜单项

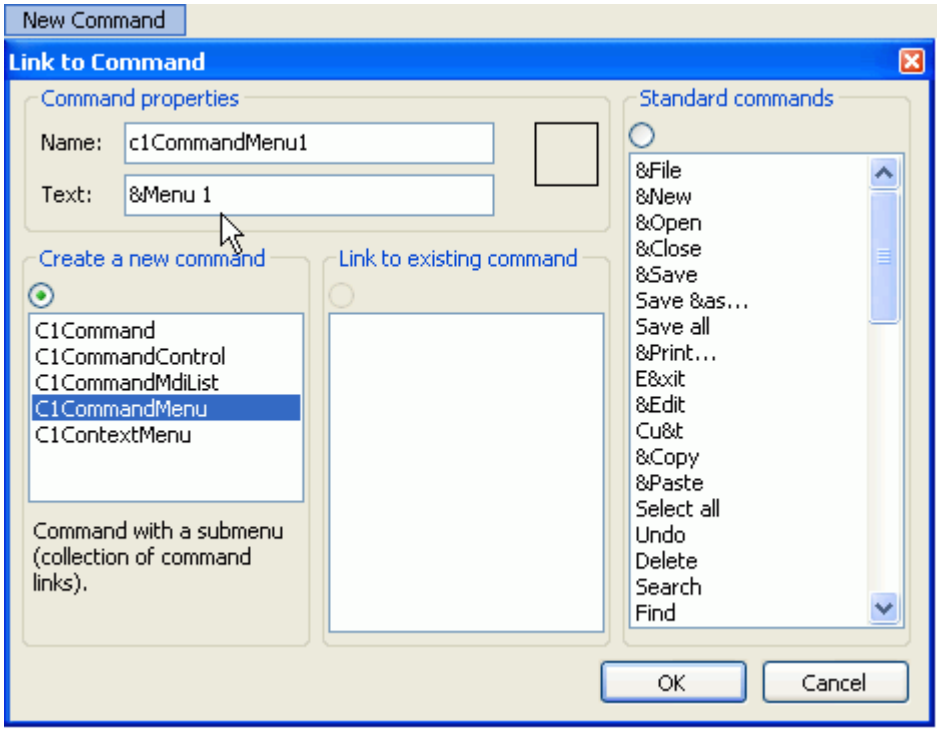
您可以在设计时向C1MainMenu添加一个菜单项，步骤是通过添加一个C1MainMenu组件，然后通过Edit设计器（菜单设计器）向其扩展菜单项。也可以通过编程方式完成，首先向Windows窗体添加一个C1CommandHolder对象，添加C1MainMenu对象，然后为菜单创建命令类型。本主题介绍如何创建一个叫做menu 1的菜单项，该菜单项可以在之后添加子菜单项。

> **注意：**C1MainMenu是一个表示主菜单的控件。它包含一组命令链接的集合，表示菜单项。只有一个主菜单可以添加到窗体。C1CommandMenu是一个命令，一个菜单。

在设计时向C1MainMenu 添加菜单项

为了在设计时向C1MainMenu添加菜单项，请使用Link to Command设计器，并完成以下步骤：

1. 通过执行拖放操作，向窗体上放置一个C1MainMenu。
2. 之后在窗体下方的组件托盘中将自动出现一个C1CommandHolder。
3. 右键单击NewCommand文本，并从其上下文菜单选择编辑。将出现Link to Command设计器。
4. 在Link to Command设计器中，选中Text字段，并键入&Menu1。



5. 选择"确定"。将出现新的菜单（Menu1）。此菜单在设计时的窗体上看起来如下图：

通过编程方式向C1MainMenu添加一个菜单项

为通过编程方式向C1MainMenu添加一个菜单项，完成以下步骤：

1. 在您的工程中添加对C1.Win.C1Command命名空间的引用。
2. 双击此窗体创建一个Form_Load事件处理函数，在您的源文件中声明此命名空间，接下来添加一个C1CommandHolder以容纳此菜单。

Visual Basic

Visual Basic

```
Imports C1.Win.C1Command
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
```

C#

C#

```
using C1.Win.C1Command;
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
```

2. 创建一个新的主菜单，然后添加此主菜单控件到您的窗体上。

Visual Basic

Visual Basic

```
Dim mm As New C1MainMenu
Me.Controls.Add(mm)
```

C#

```
C1MainMenu mm = new C1MainMenu();
this.Controls.Add(mm);
```

3. 创建一个子菜单以保存命令，并为此新菜单设置文本属性。

Visual Basic

```
Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
C1CommandMenu)
mmenu.Text = "&menu1"
```

C#

```
C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
mmenu.Text = "&menu1";
```

4. 创建一个子菜单以保存命令，并为此新菜单设置文本属性。

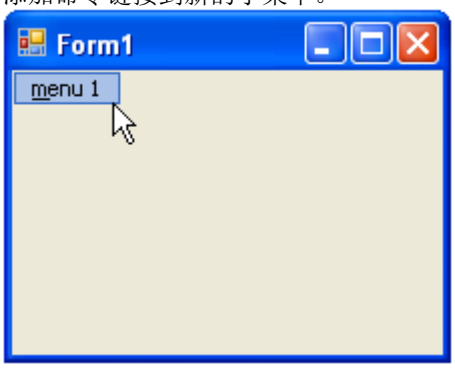
Visual Basic

```
mm.CommandLinks.Add(New C1CommandLink(mmenu))
```

C#

```
mm.CommandLinks.Add(new C1CommandLink(mmenu));
```

5. 添加命令链接到新的子菜单。



向菜单项添加一个图标

您可以在设计时或通过代码向菜单项添加一个图标。

以编程方式添加一个图标至菜单项

使用下面的代码来添加一个图标至菜单：

Visual Basic

```
Visual Basic
创建一个位图的一个新实例，并将它分配给命令对象的Image属性
c1Command1.Image = new System.Drawing.Bitmap(@'D:\componentOne\Images\App.ico')
```

C#

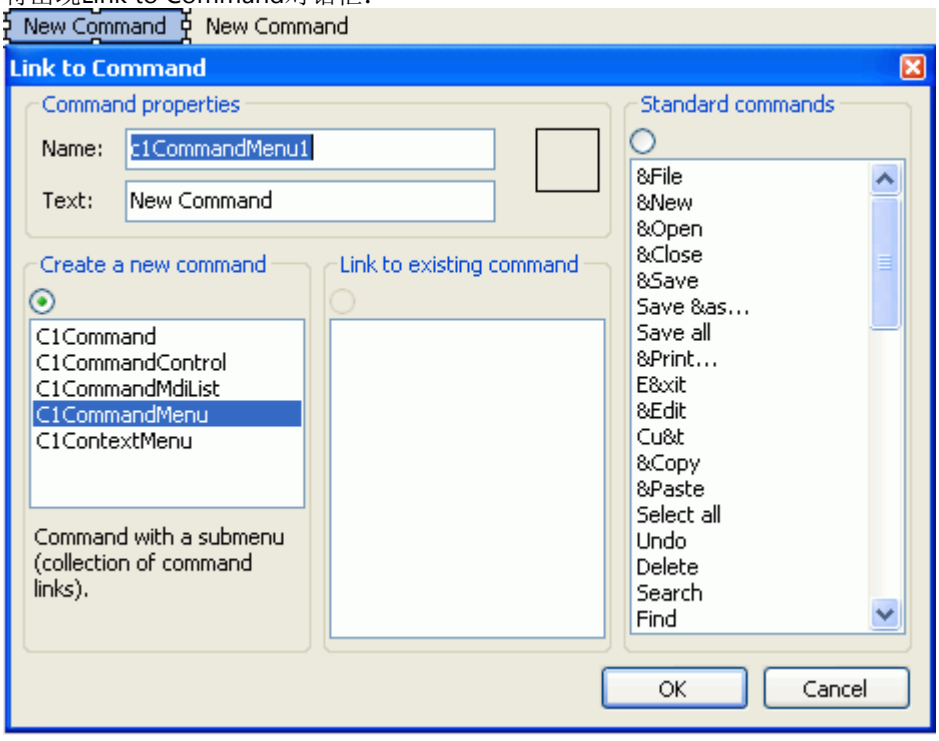
```
C#
//Create a new instance of a Bitmap and assign it
c1Command1.Image = new System.Drawing.Bitmap(@"D:\componentOne\Images\App.ico");
```

在当前菜单项之前添加菜单项

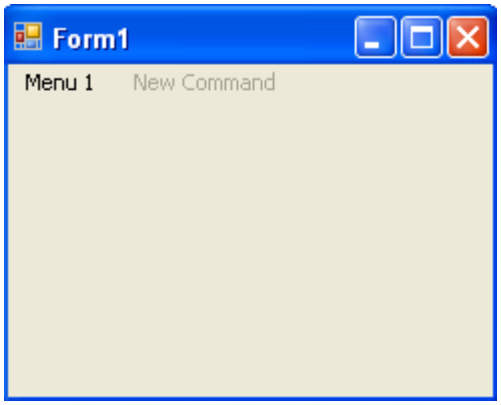
为了在设计时插入一个菜单项，从C1MainMenu的上下文菜单上选择插入项目选项。该插入项目命令建在当前的菜单项之前插入一个新的菜单项。为了在设计时插入一个菜单项，请完成以下步骤：

1. 通过拖放操作将C1MainMenu控件放置在您的窗体上。
2. 右键单击C1MainMenu控件并从其上下文菜单中选择插入项目。

将出现Link to Command对话框：



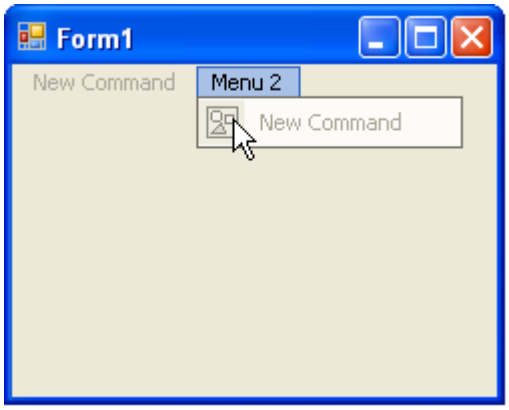
3. 在文本字段中输入Menu 1，并选择确定。下图显示设计时在您窗体上的菜单项：



在当前菜单项之后添加一个菜单项

为了在设计时，在当前菜单最后一个项目之后添加一个新的菜单项，需要完成以下步骤：

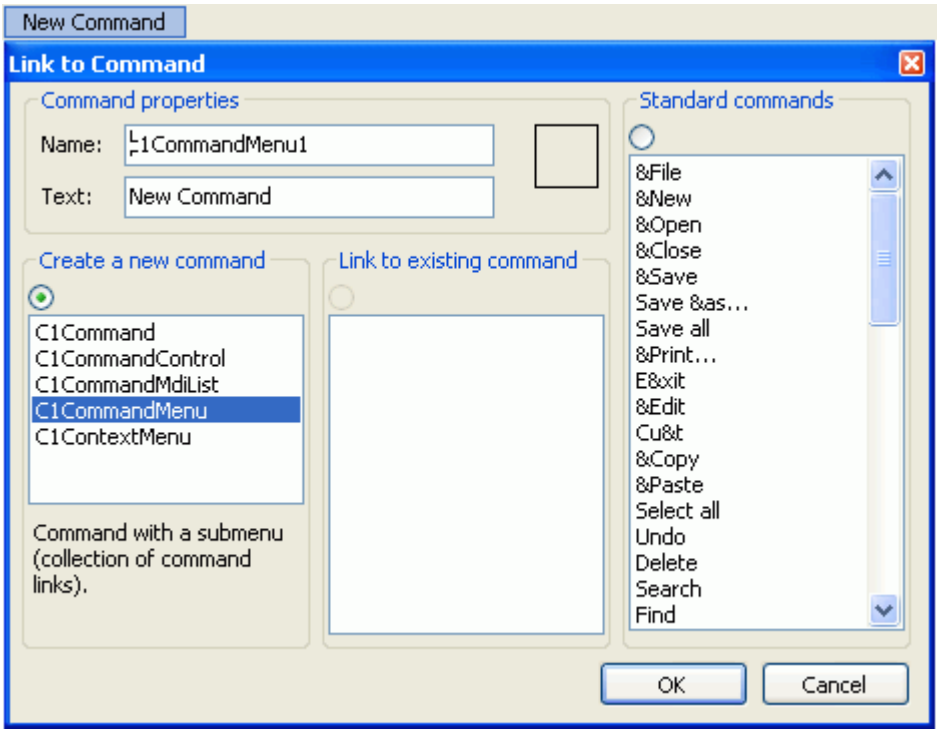
1. 通过拖放操作将C1MainMenu组件放置在您的窗体上。
2. 右键单击C1MainMenu控件并从其上下文菜单中选择追加项目。将出现Link to Command对话框：
3. 在文本字段中输入Menu 2，并选择确定。下图显示设计时在您窗体上的菜单项：



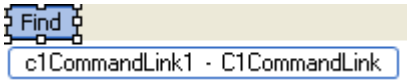
从Link to Command设计器添加一个标准菜单项

为了从Link to Command设计器添加一个具有内置图像的标准命令项，需完成以下步骤：

1. 从C1MainMenu控件上双击空白的命令项。将出现Link to Command对话框：



2. 从列表框选择一个标准的命令，例如，Find。
3. 单击“确定”。
   新的命令项以及其内置的图像将在窗体上的C1MainMenu控件上出现



4. 选择菜单，Find，并从其工具栏上选择C1CommandLink 属性按钮。
5. 将出现C1CommandLink的属性对话框。

> **Note:** The toolbar for the command item will not appear if the Smart Designer is not enabled.

6. 单击按钮外观的下拉框的下拉箭头，并选择TextAndImage。
   内置的图像将紧挨着文本“Find”显示。

> **注意：**当添加一个标准命令项至C1ToolBar控件时可以应用同样的过程。

添加子菜单

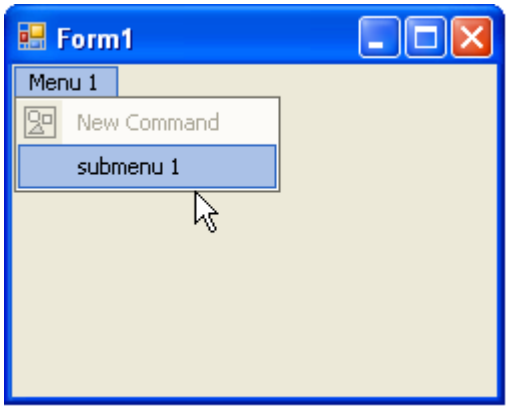
你可以通过设计器或通过代码添加子菜单。单击以下任一链接以展开设计器或者编程方式的具体步骤。

▶ 在设计时向C1MainMenu 添加子菜单项

为了添加一个新的命令链接至菜单（C1CommandMenu或者C1ContextMenu），链接到当前选中的命令链接，需完成以下步骤：

> **注意：**此选项为禁用状态，除非当前的命令链接链接到一个C1CommandMenu或者C1ContextMenu类型的命令。

1. 在C1MainMenu上右键单击一个现有的菜单项，并从其上下文菜单中选择添加子项目。
   将出现LinktoCommand设计器。
2. 在Text文本框字段中输入submenu 1并选择OK。菜单显示的结果如下所示：



通过编程方式向C1MainMenu 添加子菜单项

为了通过编程方式添加一个子菜单项，需完成以下步骤：

1. 在您的工程中添加对C1.Win.C1Command命名空间的引用。
2. 在源文件中声明该命名空间。

   Visual Basic

   ```
   Visual Basic
   Imports C1.Win.C1Command
   ```

   C#

   ```
   C#
   using C1.Win.C1Command;
   ```

3. 双击窗体创建一个Form_Load事件的事件处理程序，然后在之后的步骤中插入以下代码片段至此Form_Load事件处理程序。
4. 添加一个C1CommandHolder 以容纳此菜单，接下来创建一个新的主菜单。

   Visual Basic

   ```
   Visual Basic
   Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
   Dim mm As New C1MainMenu
   ```

   C#

   ```
   C#
   C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
   C1MainMenu mm = new C1MainMenu();
   ```

5. 向您的窗体添加主菜单控件，创建新的主菜单以容纳各种命令，接下来为此新菜单设置文本属性。

   Visual Basic

   ```
   Visual Basic
   Me.Controls.Add(mm)
   Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
   C1CommandMenu)
   ```

```
mmenu.Text = "Menu 1"
```

C#

```
this.Controls.Add(mm);
C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
mmenu.Text = "Menu 1";
```

6. 向新的主菜单添加命令链接，接下来在menu1之下创建并设置一个菜单项。使用命令填充菜单。

Visual Basic

```
mm.CommandLinks.Add(New C1CommandLink(mmenu))
Dim submenu As C1Command = ch.CreateCommand()
```

C#

```
mm.CommandLinks.Add(new C1CommandLink(mmenu));
C1Command submenu = ch.CreateCommand();
```

7. 向新的子菜单项添加文本，并向此子菜单项添加一个新的命令链接。

Visual Basic

```
submenu.Text = "submmenu 1"
'向子菜单项添加一个新的c1commandlink
mmenu.CommandLinks.Add(New C1CommandLink(submenu))
```

C#

```
submenu.Text = "submenu 1";
//向子菜单项添加一个新的c1commandlink
mmenu.CommandLinks.Add(new C1CommandLink(submenu));
```

您的菜单应当如下所示：

添加多个子菜单

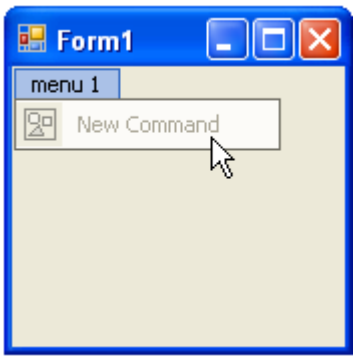
您可以在当前的子菜单之内，添加一个新的子菜单或者另一个菜单，可以通过设计器或者代码完成。单击下面的任意链

接以展开通过设计器或者代码完成这些操作的步骤。

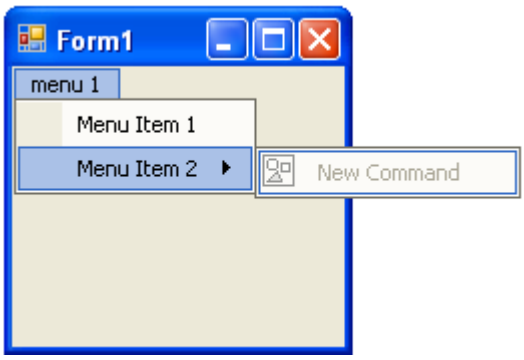
在设计时添加多个子菜单项

为了在设计时添加一个子菜单或者另一个菜单至现有子菜单，需完成以下步骤：

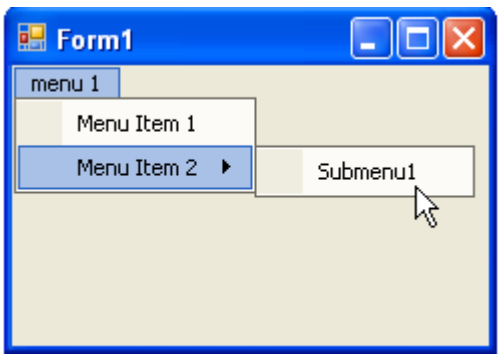
1. 通过拖放操作将C1MainMenu控件放置在您的窗体上。
2. 右键单击C1MainMenu，并从其上下文菜单上选择编辑。将出现Link to Command设计器。
3. 在Text文本框中输入menu 1 并单击OK。菜单将显示如下：



4. 单击新建命令并从其上下文菜单选择编辑。在Text文本框中输入Menu Item 1并选择OK。
5. 在窗体上单击 menu 1。
6. 在您的窗体上单击Menu Item 1并从其上下文菜单中选择追加项目。
   将出现Link to Command设计器。
7. 在命令文本框中输入Menu Item 2，然后选择确定。菜单将显示如下：
8. 在您的窗体上单击Menu Item 1并从其上下文菜单中选择编辑。将出现Link to Command设计器。
9. 在命令类型选择框中选择C1CommandMenu，之后选择确定。Menu Item 2 是一个C1CommandMenu类型，可以容纳多个菜单在其中。



10. 选择新建命令菜单项，并从其上下文菜单选择编辑。
11. 在Text文本框中输入Submenu1并单击OK。菜单外观如下图所示：



编程方式添加多个子菜单

以编程方式为子菜单添加子菜单以及其他菜单，需完成以下步骤：

1. 在您的工程中间添加对C1.Win.C1Command命名空间的引用，并在您的源文件中声明该命名空间。

   Visual Basic

   ```
   Imports C1.Win.C1Command
   ```

   C#

   ```
   using C1.Win.C1Command;
   ```

2. 双击窗体创建一个Form_Load事件的事件处理程序，然后在之后的步骤中插入以下代码片段至此Form_Load事件处理程序。
3. 添加一个C1CommandHolder以容纳该菜单，接下来创建一个新的C1CommandHolder对象。

   Visual Basic

   ```
   Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
   Dim mm As New C1MainMenu
   ```

   C#

   ```
   C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
   C1MainMenu mm = new C1MainMenu();
   ```

4. 向您的窗体添加主菜单控件，接下来创建该主菜单以容纳各种命令。

   Visual Basic

   ```
   Me.Controls.Add(mm)
   Dim mmenu As C1CommandMenu = CType(ch.CreateCommand(GetType(C1CommandMenu)),
   C1CommandMenu)
   ```

   C#

   ```
   this.Controls.Add(mm);
   C1CommandMenu mmenu = ch.CreateCommand(typeof(C1CommandMenu)) as C1CommandMenu;
   ```

5. 设置新菜单的文本属性，然后添加commandLink绑定到新的主菜单。

   Visual Basic

   ```
   ```

```
mmenu.Text = "&menu 1"
mm.CommandLinks.Add(New C1CommandLink(mmenu))
```

C#

```
mmenu.Text = "&menu 1";
mm.CommandLinks.Add(new C1CommandLink(mmenu));
```

6. 在该菜单（menu 1）之后创建并设置一个新的菜单项，接下来向菜单添加命令。

Visual Basic

```
Dim menuitem1 As C1Command = ch.CreateCommand()
```

C#

```
C1Command menuitem1 = ch.CreateCommand();
```

7. 7. 向新的菜单项添加文本，并添加一个新的c1commandlink至menuitem1。

Visual Basic

```
menuitem1.Text = "Menu Item 1"
'add a new c1commandlink to the menuitem1
mmenu.CommandLinks.Add(New C1CommandLink(menuitem1))
```

C#

```
menuitem1.Text = "Menu Item 1";
//add a new c1commandlink to the menuitem1
mmenu.CommandLinks.Add(new C1CommandLink(menuitem1));
```

8. 在菜单（menu1）下面创建一个新的命令菜单表示第二个菜单项，然后为这个新的menuitem2菜单添加一个commandlink。Menuitem2将会是一个包含子菜单的菜单。

Visual Basic

```
Dim menuitem2 As C1CommandMenu = New C1CommandMenu()
menuitem2.Text = "Menu Item 2"
menu.CommandLinks.Add(New C1CommandLink(menuitem2))
```

C#

```
C1CommandMenu menuitem2 = new C1CommandMenu();
menuitem2.Text = "Menu Item 2";
```

```
mmenu.CommandLinks.Add(new C1CommandLink(menuitem2));
```

9. 为这个新建的menuitem2创建一个子菜单，并命名为Submenu1，然后为submenu1添加一个commandlink。

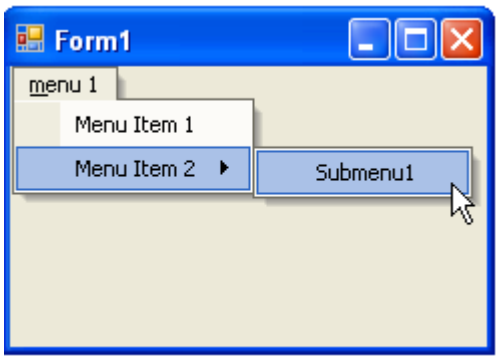
Visual Basic

```
Dim submenu1 As C1Command = ch.CreateCommand()
submenu1.Text = "Submenu1"
menuitem2.CommandLinks.Add(New C1CommandLink(submenu1))
```

C#

```
C1Command submenu1 = ch.CreateCommand();
submenu1.Text = "Submenu1";
menuitem2.CommandLinks.Add(new C1CommandLink(submenu1));
```

10. 保存并运行您的应用程序。菜单在运行时看起来如下所示：



应用快捷键至菜单

本主题演示如何通过键盘使用快捷键以及助记键访问一个菜单，而不是使用鼠标。您可以在设计时或者通过代码应用快捷键以及助记键至一个菜单项。

在设计时添加一个快捷键至菜单项

1. 向您的窗体添加一个C1MainMenu控件。
2. 右键单击该C1MainMenu控件，从上下文菜单中选择编辑，然后点击C1MainMenu。该菜单的默认名称为New Command。
3. 右键单击New Command项，从其上下文菜单中选择属性。在C1CommandLink的属性窗体中，展开Command节点，选择Shortcut属性，并从Shortcut列表框中选择F6。

> **注意：**以下步骤列举如下，让您尝试并查看该快捷键如何工作。

4. 向窗体添加一个PictureBox控件。
5. 双击NewCommand以创建NewCommand的Click事件的事件处理程序。

在事件处理程序中插入代码来改变PictureBox的背景色为紫色：

Visual Basic

Visual Basic

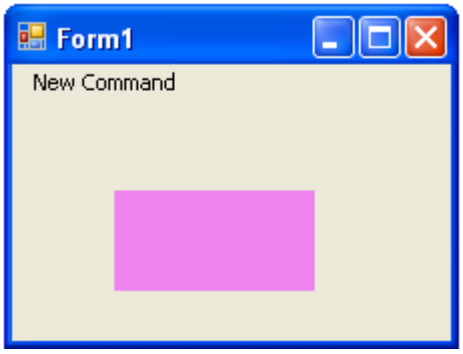
```
If sender Is c1CommandMenu1 Then
        Me.pictureBox1.BackColor = Color.Violet
End If
```

C#

C#

```
if (sender == c1CommandMenu1)
        {
                this.pictureBox1.BackColor = Color.Violet;
        }
```

6. 运行应用程序在窗体出现时，按下F6键。图片框将显示在窗体上。



以编程方式添加一个快捷键至菜单项

1. 在您的源代码文件中，找到菜单中您希望快捷键关联到的菜单项。
2. 通过输入下面的代码为C1CommmandMenu1创建一个快捷键：

Visual Basic

Visual Basic

```
Me.C1CommandMenu1.Shortcut = System.Windows.Forms.Shortcut.F6
```

C#

C#

```
this.C1CommandMenu1.Shortcut = System.Windows.Forms.Shortcut.F6;
```

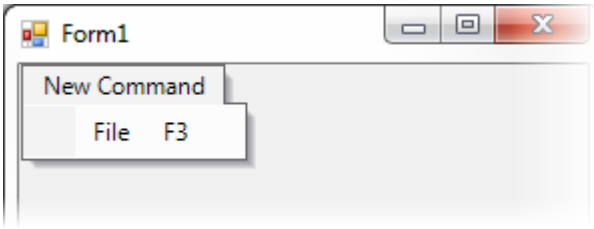
本地化快捷键的文本

为本地化的文本快捷键，需完成以下步骤：

1. 向窗体添加C1MainMenu 控件。
2. 右键点击控件并从上下文菜单中选择编辑。
3. 在Link to Command对话框中单击“确定”。
4. 从属性下拉列表中选择C1CommandLink2。
5. 右键单击New Command并从上下文菜单中选择编辑。在Text文本框中输入File并单击确定。
6. 在属性窗口中，找到并展开C1CommandLink2.Command节点。

7. 选中C1CommandLink2.Shortcut并从下拉列表中选择F3。
8. 对于本话题，选中C1CommandLink2.ShortcutText属性并设置为F3。

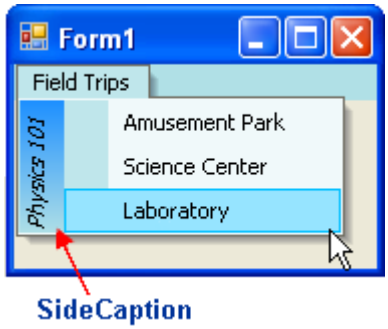
快捷方式的文本将如下面的图片所示：



为CommandMenu创建一个侧标题

为了在设计时使用SideCaption属性以便在命令的旁边显示侧标题，需完成以下步骤：

1. 从属性下拉列表中选择C1CommandMenu。
2. 找到并展开C1CommandMenu.SideCaption节点。
3. 选择BarGradientBegin属性并从Web颜色分类标签中选择DodgerBlue颜色。
4. 选择BarGradientEnd属性并从Web颜色分类标签中选择PowderBlue颜色。
5. 设置Text属性为Physics 101。
6. 侧标题将出现在菜单项的左侧。以下图片演示了侧标题的外观：



注意：显示在在侧标题文本表示参加郊游的班级的名称。

为命令对象创建一个独立的单击事件

为了指定一个和ClickEventHandler委托具有相同签名的方法的地址至C1Command的Click事件，使用AddHandler方法。为此，使用下面的代码：

Visual Basic

```
Visual Basic
Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
'创建文件操作的命令。
Dim cNew as C1Command = ch.CreateCommand()
'使用AddHandler方法指定一个Click事件的委托。
AddHandler cNew.Click, New ClickEventHandler(AddressOf cNew_Click)
```

C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
//创建文件操作的命令。
C1Command cNew = ch.CreateCommand();
//使用AddHandler方法指定一个Click事件的委托。
cNew.Click += new C1.Win.C1Command.ClickEventHandler(cNew Click);
```

创建一个MDI窗体的窗体列表

C1Command具有一个C1CommandMdiList的命令类型，用作为一个MDI（多文档接口）的窗体创建一个窗体列表。一个窗体列表在追踪一个应用程序已经打开的不同MDI子窗体时非常有用。

为创建一个MDI窗体的窗体列表，需完成以下步骤：

1. 在属性窗口中，设置 Form1.IsMDIContainer属性为True。
2. 在解决方案资源管理器中，右击该项目并选择Add|Add New Item。
3. 从对话框中，选择Windows窗体，命名该窗体为MdiChild，然后将其Text属性设置为MdiChild。
4. 添加C1MainMenu组件至您的MDI父窗体，Form1。
5. 在属性窗体中，设置C1MainMenu的Name属性为C1MainMenu1。
6. 创建File菜单。通过Link to Command设计器向C1MainMenu添加一个C1CommandMenu组件以及一个子菜单项。
   设置以下属性：

Command Name	Command Type	Command Text
cmdFile	C1CommandMenu	&File
cmdFileNew	C1Command	&New

7. 创建Window菜单存储的MDI子窗口。在Link to Command设计器中，添加另一个C1CommandMenu至C1MainMenu组件，并向其添加一个C1CommandMdiList类型的子菜单项。

   设置以下属性：

Command Name	Command Type	Command Text
cmdWindow	C1CommandMenu	&Window
c1CommandMdiList1	C1CommandMdiList	<MDI Windows List>

8. 创建一个程序用来将MdiChild窗体的新实例显示为Form1的MDI Children。

   Visual Basic

   Visual Basic

```
Private Sub createNewMdiChild()
        Dim mc As New MdiChild()
        mc.MdiParent = Me
        mc.Text = String.Format("MDI Child Window {0}", Me.MdiChildren.Length)
        mc.Show()
End Sub
```

   C#

C#

```
private void createNewMdiChild()
    {
          MdiChild mc = new MdiChild();
          mc.MdiParent = this;
          mc.Text = string.Format("MDI Child Window {0}",
this.MdiChildren.Length);
          mc.Show();
    }
```

9. 在设计视图中，双击cmdFileNew菜单项以创建一个单击事件处理程序，并调用createNewMdiChild方法。将下面的代码添加到事件处理程序内：

▸ Visual Basic

Visual Basic

```
createNewMdiChild()
```

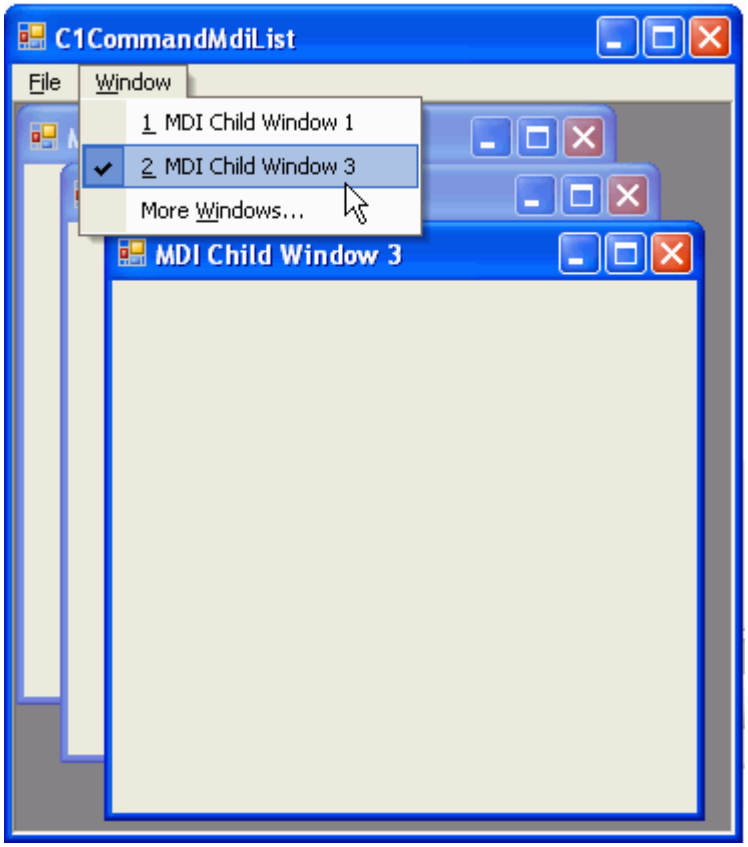
▸ C#

C#

```
CreateNewMdiChild();
```

10. 按F5运行应用程序。
11. 从"文件"菜单上，单击"新建"创建一个新的MDI子窗体。

C1CommandMdiList将扩展可用的MDI子窗体列表。



> **注意：**Window菜单始终显示在应用程序内打开的MDI子窗体列表，同时在具有焦点的MDI子窗体项目的旁边显示一个复选标记。

删除菜单项

当一个菜单项高亮显示时，按下DEL键或者从右键菜单上选择删除，命令链接以及该命令链接指向的命令（简单命令或菜单命令）将被删除，取决于当前哪一个被选中。（为确认当前被选中的项目，请查看Visual Studio 设计器的属性窗口。）

在设计时删除菜单项

为删除命令链接（这将会保留命令本身，之后您可以从另一个命令链接链接到此命令）：

1. 单击要删除的菜单项。确保属性窗体中显示的是命令链接属性而不是命令的属性。
2. 右键单击此项目以打开上下文菜单，并选择删除，或者简单地按下DEL键。
   删除命令本身（这将删除命令，但保留命令链接，您可以再次将其连接到另一个命令或菜单）
3. 单击要删除的菜单项。然后再单击一次。这一动作将选择命令而不是命令链接。确保属性窗口中显示该命令的属性而不是命令链接的属性。
4. 在项目的右键打开上下文菜单并选择删除，或者按DEL键。该命令将被删除，留下空白的命令链接。您可以通过右键单击并从上下文菜单选择编辑，重新将其链接到另一个命令。

编程方式删除一个菜单项

为了删除菜单，可以使用Dispose方法。使用下面的代码来删除菜单或菜单项：

Visual Basic

Visual Basic

```
'该菜单的变量名称叫做menu
'以下代码将删除此菜单
menu.Dispose()

'该菜单项的变量名称叫做 menuitem1
'以下代码将删除此菜单 item
menuitem1.Dispose()
```

C#

C#

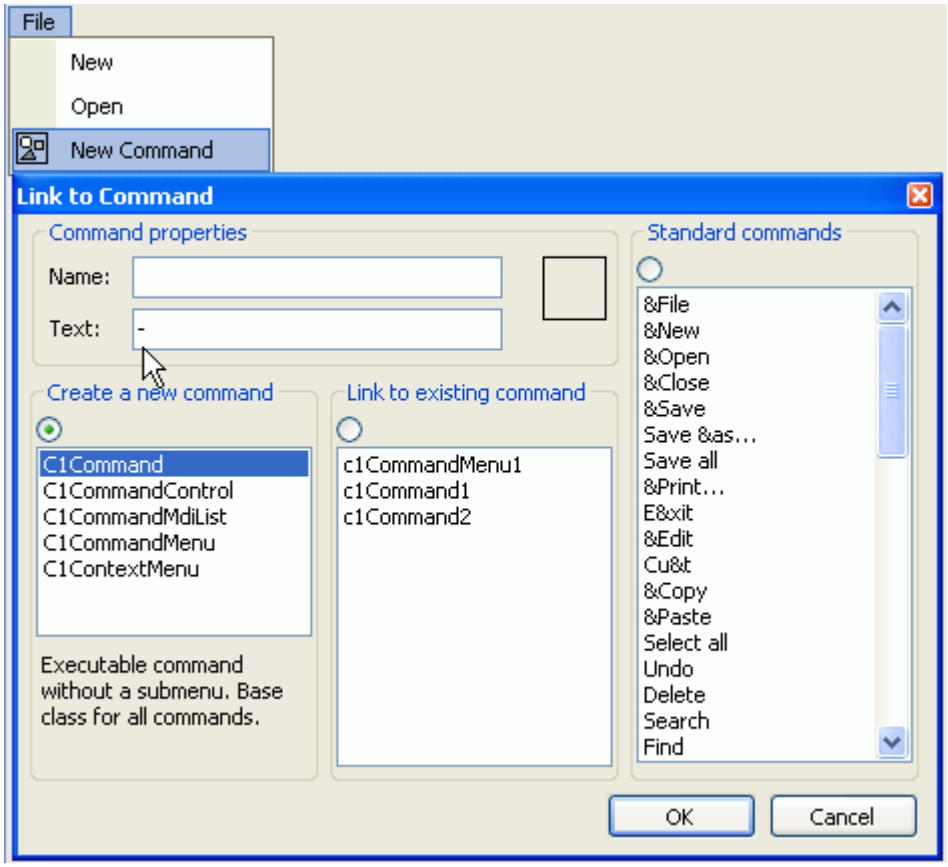
```
//该菜单的变量名称叫做menu
//以下代码将删除此菜单
menu.Dispose();

//该菜单项的变量名称叫做 menuitem1
//以下代码将删除此菜单 item
menuitem1.Dispose();
```

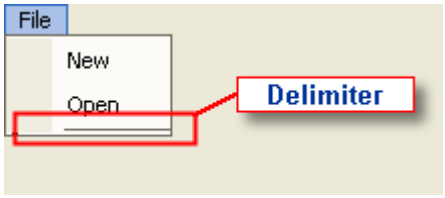
在下拉菜单中显示分隔符

在下拉菜单中显示一个分隔符，请完成以下步骤：

1. 从现有的下拉菜单右键单击一个命令，并从其上下文菜单选择追加项目。
2. 一个新的命令将出现。在新命令的Link to Command设计器中，在Name文本框中删除名称。在Text文本框中将名称改变为一个连字符。Text文本框以及Name文本框字段输入的值应当像下面的Link to Command设计器所示。



3. 选择确定，之后您的命令链接就会显示为一个条分隔符，如下面所示。



在菜单和工具栏显示工具提示

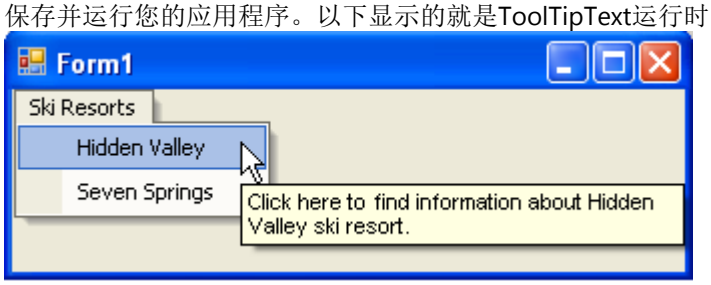
本主题通过创建自己的菜单和工具栏提示演示如何进行自定义工具提示。菜单和工具栏均具有叫做ToolTipText的属性，该属性提供给用户当鼠标悬停在菜单或工具栏上方时，显示工具提示的能力。

为菜单项显示工具提示

1. 右键点击在主菜单中的顶级菜单（C1CommandMenu）从上下文菜单选择属性。
2. 从属性下拉列表选择C1CommandMenu1并展开Command节点。
3. 设置ShowToolTips属性值为True，以启用当鼠标悬停在菜单上方时，出现文本提示的能力。
4. 在窗体上，选中一个您希望显示工具提示的C1Command菜单项。设置
5. ShowTextAsToolTip属性的值为False，因此当鼠标悬停在C1CommandMenu选中的菜单项上方时，命令的文本将不会被用作工具提示。

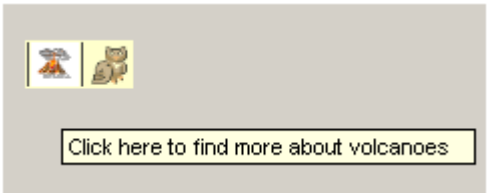
6. 选中ToolTipText属性并在此文本框中输入您希望在鼠标悬停在菜单项上方时显示的文本内容。

   保存并运行您的应用程序。以下显示的就是ToolTipText运行时的外观效果：



为工具栏按钮显示工具提示

1. 从窗体上C1ToolBar控件选中一个工具栏按钮。
2. C1ToolBar命令链接的属性将出现在右侧面板的属性工具栏上。查找C1CommandLinks属性并展开Command属性。
3. 设置ShowTextAsToolTip属性的值为False。
4. 选中ToolTipText属性并在此文本框中输入您希望在鼠标悬停在工具栏按钮上方时出现的工具提示的内容。
5. 保存并运行您的应用程序。以下是工具栏按钮的工具提示文本的显示效果：



注意：如果您将ShowTextAsToolTip保持为其默认值，True，那么当鼠标悬停在工具栏上方时，将显示工具栏按钮的文本内容，而不是工具提示的文本。同时，需要删除您可能输入的工具栏的ToolTipText属性的内容。工具栏看起来像下面的图片：

在Outbar上为按钮显示工具提示

您可以按照上面所列举的相同的方式为显示在Outbar上的按钮显示工具提示。

隐藏较少使用的菜单项

隐藏长期未被使用的菜单项，使用C1CommandHolder对象上的RecentLinksThreshold属性来启用此功能。

为打开此功能，可以将此属性的值设置为一个0到100之间的值。一个好的起始值是50。

合并菜单项

为了将当前的菜单链接和来自于子窗体上菜单的链接合并，请使用**MergeCommandLinks**方法：

▶ Visual Basic

Visual Basic

```
Me.c1CommandHolder1.MergeCommandLinks(Me.c1MainMenu1.CommandLinks,
Me.c1MainMenu1.CommandLinks,
_form2.c1MainMenu1.CommandLinks)
```

C#

C#

```
this.c1CommandHolder1.MergeCommandLinks(this.c1MainMenu1.CommandLinks,
this.c1MainMenu1.CommandLinks,
_form2.c1MainMenu1.CommandLinks);
```

修改菜单的外观

在设计时使用属性编辑器修改C1MainMenus的属性。

通过相关的属性自定义每一个菜单项，如下面的步骤所演示：

1. 右键单击C1MainMenu控件。选择上下文菜单中的菜单属性

   注意：你也可以在属性窗体中查看C1MainMenu的属性。

2. 从属性下拉框选择C1MainMenu1并设置相关的Font属性：

   - Font=Verdana
   - FontSize=11pt

3. 从属性下拉框选择C1MainMenu1并设置相关的属性：

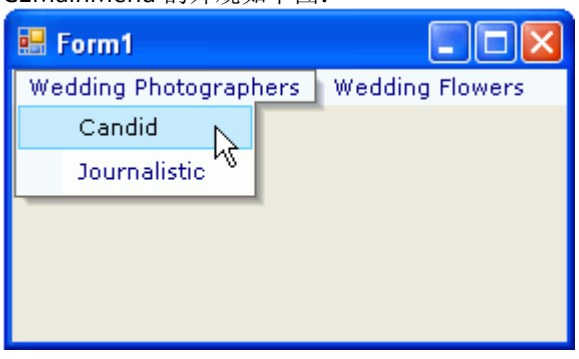
   - BackColor = Select the Web tab and select Alice Blue.
   - BackHiColor= #66CCFF
   - ForeColor = Select the Web tab and select Navy.
   - ForeHiColor = Black

4. 构建并运行此Web应用程序。

该主题演示如下：

- 除非另有修改，菜单项默认从父菜单继承格式。
- 你可以根据需要修改个别项目。

C1MainMenu 的外观如下图：



注意：下图演示本主题中，不带有内容的C1MainMenu的外观。

设置图像/复选标记栏的宽度

为设置C1CommandMenu上的图像/复选标记栏的宽度，请使用ImageBarWidth属性。设置C1CommandMenu的ImageBarWidth属性为一个大于零的值。

当没有安装任何MessageFilter时显示一个对话框窗体

C1Command的菜单和工具栏依靠安装一个消息过滤器（实现IMessageFilter 接口）以实现主菜单以及其他一些功能。在某些情况下（例如，当C1Command用于组件的设计器，并在VisualStudio设计时中运），安装的消息过滤器将不工作。在这种情况下，C1Command仍然可以使用。

下面的代码片段演示如何在当消息过滤器无法安装，显示一个对话框。你可以使用这个方法，比如说，您希望在组件的设计器中显示一个对话框。

Visual Basic

Visual Basic

```
Imports C1.Win.C1Command;
    C1CommandHolder.UninstallMessageFilter()
    '创建 C1CommandMsgHook
    Dim hook As New C1CommandMsgHook()
    hook.Install()
    Try
        result = dialog.ShowDialog()
    Finally
                Hook.Uninstall();
    End Try
```

C#

C#

```
using C1.Win.C1Command;
        ...
        C1CommandHolder.UninstallMessageFilter();
        C1CommandMsgHook hook = new C1CommandMsgHook();
        hook.Install();
        try
        {
          result = dialog.ShowDialog();
        }
        finally
        {
          hook.Uninstall();
        }
```

在菜单末尾对项目换行

为了在菜单末尾对项目进行折行，请使用C1MainMenu的Wrap属性。

如果为True（默认值），当窗体宽度不足时，菜单将折行显示。如果为False，菜单不折行，替代方案是在末尾添加一个扩展菜单。

工具条任务

此章节介绍怎么执行特定的工具条任务。

在工具栏中添加任意控件

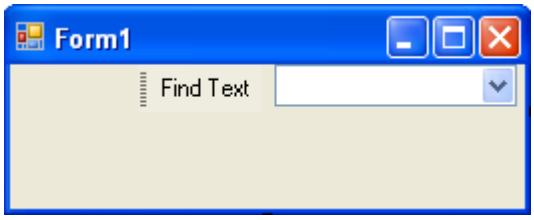
按以下步骤在工具条中添加任意控件：

1. 在窗体上添加C1CommandDock控件。
2. 拖动一个C1ToolBar控件到C1CommandDock控件内。
3. 右键点击窗体上的C1ToolBar控件。在上下文菜单中选择Edit。弹出Link to Command编辑器。在Link to Command编辑器中按如下设置：

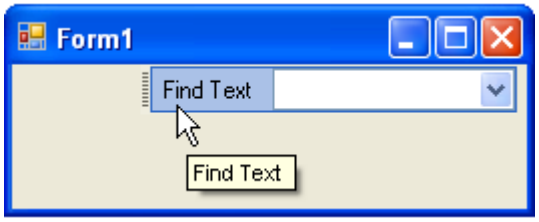
   - Text文本框设置为Find Text:
   - Name设置为c1CommandControl1
   - Command Type设置为C1CommandControl

   点击编辑器对话框中的确认按钮。

4. 在创建新命令列表框中选择C1CommandControl。
5. 在Properties下拉列表中选择c1CommandLink1，然后设置ButtonLook属性值为Text。这个工具条按钮显示为文本样式。
6. 在工具箱中选择Windows Form标签，拖动一个ComboBox控件到C1ToolBar控件的右侧。组合框显示如下图所示：



7. 在Properties下拉列表中选择c1CommandControl1，然后选中Control属性，在属性值的下拉列表中选择comboBox1。
8. 编译运行程序。运行时工具条显示如下图所示：



在工具条中加入图片

此主题假定你已经创建了一个工具条按钮。下列步骤演示了如何添加一个图片到工具条按钮中：

1. 在源文件中定位到需要加入图片的位置。
2. 键入下列代码来添加一个图片到你的工具条按钮中。

Visual Basic

```
Visual Basic
cNew.Image = System.Drawing.Image.FromFile("C:\bitmap\New.bmp")
```

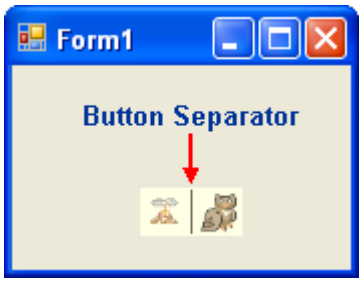
C#

```
C#
cNew.Image = System.Drawing.Image.FromFile("C:\\bitmap\\New.bmp");
```

在按钮中插入分隔符

步骤如下：

1. 右键点击第二个工具条按钮，然后在上下文菜单中选择Properties。
2. 在外观属性类别中设置Delimiter属性值为True。
3. 保存运行程序。按钮分隔符显示如下图所示：



将水平工具条的位置调整为垂直

按以下步骤使用Horizontal属性将工具条位置从水平改变为垂直。

1. 创建一个工具条。
2. 使用如下代码使工具条显示为垂直：

Visual Basic

```
Visual Basic
toolbar.Horizontal = False
```

C#

```
C#
toolbar.Horizontal = False;
```

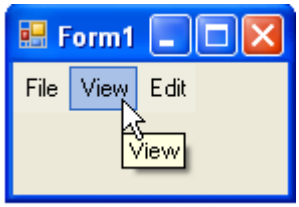
创建工具条

你可以在设计时或者通过代码来创建一个工具条。点击下列任何一个链接以展开通过设计器或者代码方式的步骤。

▸ 在设计时创建一个工具条

按如下步骤使用Link to Command编辑器创建一个C1ToolBar。

1. 在窗体中加入一个C1ToolBar控件，然后右键点击C1ToolBar控件并且从上下文菜单中选择Edit。弹出被选择工具条的Link to Command编辑器。
2. 在Text文本框中输入File，然后点击OK按钮。
3. 在Properties下拉框中选择c1CommandLink1。设置这个File工具条按钮对应的CommandLink的ButtonLook属性值为Text。这个File工具条按钮以文本方式显示。
4. 右键点击C1ToolBar控件并且在上下文菜单中选择Append Item。将在当前的CommandLink后添加一个新的命令连接。弹出Link to Command编辑器。
5. 在Text文本框中输入Edit，然后点击OK按钮。
6. 在Properties下拉列表中选择c1CommandLink2。设置这个Edit工具条按钮对应的CommandLink的ButtonLook属性值为Text。
7. 右键点击Edit工具条按钮然后在上下文菜单中选择Insert Item。将在其后添加一个新的命令连接。弹出Link to Command编辑器。
8. 在Properties下拉列表中选择c1CommandLink3。设置这个View工具条按钮对应的CommandLink的ButtonLook属性值为Text。
9. 编译运行程序。
   在运行时，工具条显示如下图所示：



▸ 以编程方式的创建工具条

按以下步骤以编程方式创建一个包含文本按钮的C1ToolBar：

1. 添加C1.Win.C1Command引用到项目中。
2. 在代码中申明命名空间，然后添加一个用来容纳工具条的C1CommandHolder。

   ▸ Visual Basic

   Visual Basic
   ```
   Imports C1.Win.C1Command
   Dim ch As C1CommandHolder = C1CommandHolder.CreateCommandHolder(Me)
   ```

   ▸ C#

   C#
   ```
   using C1.Win.C1Command;
   C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
   ```
3. 创建一个新的C1ToolBar，然后将它添加到窗体中。

   ▸ Visual Basic

Visual Basic

```
Dim tb As New C1ToolBar()
Me.Controls.Add(tb)
```

C#

```
C1ToolBar tb = new C1ToolBar();
this.Controls.Add(tb);
```

4. 将C1CommandHolder分派给C1ToolBar，然后创建一个新的工具条命令。新添加的命令的名字设置为File。

Visual Basic

```
tb.CommandHolder = ch
Dim cFile As New C1Command()
cFile.Text = "File"
```

C#

```
tb.CommandHolder = ch;
C1Command cFile = new C1Command();
cFile.Text = "File";
```

5. 为新命令创建一个新的CommandLink，然后将新CommandLink加入到工具条中。

Visual Basic

```
Dim cl As C1CommandLink
cl = New C1CommandLink(cFile)
tb.CommandLinks.Add(cl)
```

C#

```
C1CommandLink cl = new C1CommandLink(cFile);
tb.CommandLinks.Add(cl);
```

6. 让这个CommandLink显示为文本样式，然后创建另外一个名字为View的命令。

Visual Basic

```
cl.ButtonLook = ButtonLookFlags.Text
Dim cView As New C1Command()
cView.Text = "View"
```

C#

C#

```
cl.ButtonLook = ButtonLookFlags.Text;
C1Command cView = new C1Command();
cView.Text = "View";
```

7. 为新的View命令创建一个新的CommandLink，然后将它加入到工具条中。

Visual Basic

```
cl = New C1CommandLink(cView)
tb.CommandLinks.Add(cl)
```

C#

```
cl = new C1CommandLink(cView);
tb.CommandLinks.Add(cl);
```

8. 让这个View CommandLink显示为文本样式，然后创建另外一个名字为Edit的命令。

Visual Basic

```
cl.ButtonLook = ButtonLookFlags.Text
Dim mEdit As New C1Command()
mEdit.Text = "Edit"
```

C#

```
cl.ButtonLook = ButtonLookFlags.Text;
C1Command mEdit = new C1Command();
mEdit.Text = "Edit";
```

9. 为新的Edit命令创建一个新的CommandLink，然后将它添加到工具条中。

Visual Basic

```
cl = New C1CommandLink(mEdit)
tb.CommandLinks.Add(cl)
```

C#

```
cl = new C1CommandLink(mEdit);
tb.CommandLinks.Add(cl);
```

10. 让Edit CommandLink显示为文本样式。

Visual Basic

Visual Basic

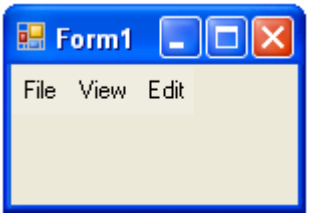
```
cl.ButtonLook = ButtonLookFlags.Text
```

C#

```
cl.ButtonLook = ButtonLookFlags.Text;
```

11. 保存后运行程序。

工具条显示如下图所示：



以编程的方式停靠一个工具条

按以下步骤以使用C1CommandDock让**C1ToolBar**可停靠：

> **注：**此主题假定你已使用代码创建了一个C1ToolBar。

1. 创建一个新的**C1CommandDock**并且赋值给dock变量。

Visual Basic

```
Dim dock As New C1CommandDock()
```

C#

```
C1CommandDock dock = new C1ComandDock();
```

2. 将C1CommandDock控件添加到窗体中。

Visual Basic

```
Me.Controls.Add(dock)
```

C#

```
this.Controls.Add(dock);
```

增加工具条中图片的尺寸

为了显示更大的图标或位图，需要增加工具条按钮的尺寸。这将相应的增加图片的尺寸。可以使用工具条的MinButtonSize属性来实现。MinButtonSize属性的默认尺寸是24 pixels。

按如下步骤在设计时增加工具条按钮的尺寸：

- 打开C1Toolbar任务菜单，在C1ToolBar的MinButtonSize文本框中输入50 pixels。

  或者

- 在属性窗口中设置C1ToolBar.MinButtonSize属性值为40 pixels。

下面图片中工具条的最小按钮尺寸被增加到40像素。

调整工具条的外观

通过以下步骤更改toolbar外观：

1. 在Properties下拉列表中选择C1ToolBar。
2. 展开C1ToolBar对象的Font属性。设置下列字体属性：
   Font.Name = Arial
   Font.Size = 9
3. 设置下列C1ToolBar属性：
   BackColor = #0066CC
   BackHiColor = Pale Turquoise
   ForeColor = White
   ForeHicolor = #003399
4. 展开C1ToolBar的Border属性。设置下列边框属性：
   Border.Light Color = Pale Turquoise
   Border.Style = Outset
   Border.Width = 3

在运行时工具条显示如下图所示：

让工具条按钮中的图片更明亮

SmoothImages属性值默认为True，非选择状态的图片以平滑的方式显示。如果设置这个属性值为False，非选择状态的图片以更明亮的方式显示。

如下表格说明了SmoothImages属性设置为True和False时不同的显示效果：

属性	工具条图片
False	
True	

指定一个停靠/浮动位置

此主题演示了如何使用代码设置C1CommandDock的位置。

使用如下代码将工具条停靠在上方。

Visual Basic

```
Visual Basic
dock.Dock = DockStyle.Top
```

C#

```
C#
dock.Dock = DockStyle.Top;
```

注：如果想让工具条停靠在下方，左方或者右方，改变枚举值Top为相应的位置：Bottom，Left或者Right。

打开自定义功能

设置CustomizeButton属性值为True可以打开自定义功能，允许用户在运行时自定义工具条。参照动态自定义工具条获取这个功能的更多信息。

截断C1ToolBar中的文本

按以下步骤使用WrapText属性来截断C1ToolBar中的文本。
1. 设置ButtonWidth属性为一个大于0的值。
2. 设置WrapText属性值为True。

ContextMenu任务

本章节说明了如何执行特定的上下文菜单任务。

为一个控件添加C1ContextMenu

可以在设计时或者通过代码为一个控件添加C1ContextMenu。点击以下任何一个链接来展开在设计时或者通过代码实现的步骤。

按以下步骤来创建一个上下文菜单并且关联一个菜单项：

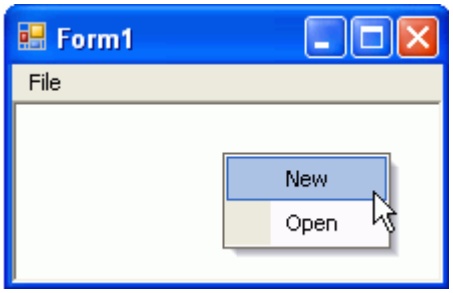
按以下步骤来创建一个上下文菜单并且关联一个菜单项：

注：在本例中，C1ContextMenu是和菜单项关联的。

1. 在窗体中放置一个C1MainMenu，右键点击New Command，然后从上下文菜单中选择Edit。弹出Link to Command编辑器。
2. 在Link to Command编辑器中，设置如下命令属性：
   o Text文本框设置为File
   o Name文本框设置为MenuFile
3. 在Create a new command列表框中选择ContextMenu。这将创建一个带有子菜单并可被用作上下文菜单的命

令。在Link to Command对话框中点击OK按钮。File将显示在新创建的C1MainMenu上。

4. 右键点击File菜单上的New Command项，然后在上下文菜单中选择Edit。在弹出的Link to Command编辑器中设置如下属性：
   o Text设置为New
   o Name设置为cmdFileNew
5. 在Create a new command列表框中选择C1Command。
6. 点击OK按钮。
7. 右键点击New菜单，在上下文菜单中选择Append Item，然后在弹出的Link to Command编辑器中设置如下属性：
   o Text设置为Open
   o Name设置为cmdFileOpen
8. 在Create a new command列表框中选择C1Command。
9. 点击OK按钮。
10. 在工具箱中选择Windows Form标签，然后拖动RichTextBox控件到窗体上。
11. 在Properties下拉列表中选择richTextBox1控件，然后设置其Dock属性值为Fill。
12. 在Properties下拉列表中选择MenuFile，然后选中Category属性，在属性框中输入File。
13. 在Properties下拉列表中选择richTextBox1，然后在C1CommandHolder属性值列表中选择C1ContextMenu，然后选择MenuFile。
14. 运行程序然后右键点击在富文本框中的任何位置。弹出在File菜单中定义的上下文菜单，如下图所示。



以编码的方式为一个控件添加C1ContextMenu

按如下步骤来为一个控件添加一个C1ContextMenu：

1. 在解决方案资源管理器中为项目添加C1.Win.C1Command引用，然后在源文件中添加C1.Win.C1Command命名空间声明。
2. 使用拖拽方式将一个TextBox控件添加到窗体上。
3. 为了创建一个C1CommandHolder来容纳命令，在窗体上双击来创建Form_Load事件处理程序，并且添加以下代码：

> **注：**如果需要在窗体上添加多个命令容器，请使用try和catch语句。这将捕获并且忽略重复添加多个命令容器的错误。

Visual Basic

Visual Basic

```
Dim ch As C1.Win.C1Command.C1CommandHolder
ch = C1CommandHolder.CreateCommandHolder(Me)
```

C#

C#

```
C1CommandHolder ch = C1CommandHolder.CreateCommandHolder(this);
```

4. 创建和设置一个复制命令，然后使用Click事件处理程序处理在点击菜单项时执行复制命令。同时设置命令的查询

事件处理程序，以使命令保持最新的状态。

Visual Basic

```
'Create and set up the Copy command
Dim cmdCopy As C1Command = ch.CreateCommand()
cmdCopy.Text = "&Copy"
AddHandler cmdCopy.Click, AddressOf clickCopy
AddHandler cmdCopy.CommandStateQuery, AddressOf queryCopy
```

C#

```
// Create and set up the Copy command
C1Command cmdCopy = ch.CreateCommand();
cmdCopy.Text = "&Copy";
cmdCopy.Click += new C1.Win.C1Command.ClickEventHandler(clickCopy);
cmdCopy.CommandStateQuery += new
C1.Win.C1Command.CommandStateQueryEventHandler(queryCopy);
```

5. 创建一个上下文菜单来容纳复制命令，然后将这个上下文菜单分派给一个文本框。为了创建一个上下文菜单，必须在C1CommandHolder中创建一个命令并且将它分派给一个上下文菜单。使用C1CommandHolder类的C1CommandHolder.CreateCommand和C1CommandHolder.SetC1ContextMenu方法。

Visual Basic

```
Dim cm As C1ContextMenu = Ctype(ch.CreateCommand(GetType(C1ContextMenu)),
C1ContextMenu)
'Fill it with the links to the commands
cm.CommandLinks.Add(New C1CommandLink (cmdCopy))
ch.SetC1ContextMenu(TextBox1, cm)
```

C#

```
C1ContextMenu cm = ch.CreateCommand(typeof(C1ContextMenu)) as C1ContextMenu;
// Fill it with the links to the commands
cm.CommandLinks.Add(new C1CommandLink(cmdCopy));
ch.SetC1ContextMenu(textBox1, cm);
```

6. 调用clickCopy方法来处理复制命令的行为。使用queryCopy方法提供复制命令的当前状态。当在上下文菜单中点击复制命令，当前文本框中的文本将被复制。可以使用以下代码来实现：

Visual Basic

```
Private Sub clickCopy(ByVal sender As Object, ByVal e As
C1.Win.C1Command.ClickEventArgs)
   Me.textBox1.Copy()
End Sub
```

```
 'provides the current state of the copy command

Private Sub queryCopy(ByVal sender As Object, ByVal e As
C1.Win.C1Command.CommandStateQueryEventArgs)
   e.Enabled = Me.textBox1.SelectionLength > 0
End Sub
```

C#

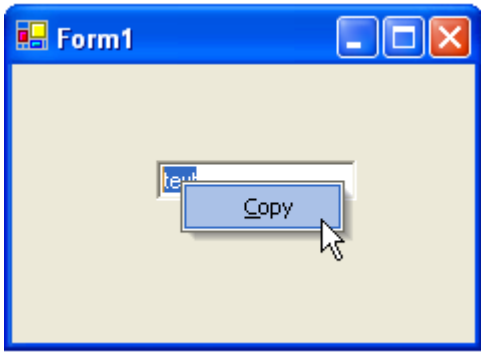
C#

```
private void clickCopy(object sender, C1.Win.C1Command.ClickEventArgs e)
    {
        this.textBox1.Copy();
    }

//provides the current state of the copy command

private void queryCopy(object sender,
C1.Win.C1Command.CommandStateQueryEventArgs e)
    {
        e.Enabled = this.textBox1.SelectionLength > 0;
    }
```

7. 保存后运行程序。在文本框中输入一些文本，然后右键点击在文本上以弹出上下文菜单。如下图片显示了在运行时上下文菜单显示在文本框上的情形。



获取附加在C1TextBox上的C1ContextMenu控件

使用C1CommandHolder类的GetC1ContextMenu方法来确定一个控件被附加了哪个C1ContextMenu。GetC1ContextMenu方法返回附加在特定控件上的上下文菜单。

使用如下代码来获取附加在C1TextBox1上的C1ContextMenu控件的名字：

Visual Basic

Visual Basic

```
'retrieves the contextmenu attached to the C1TextBox control
Dim contextMenu As C1.Win.C1Command.C1ContextMenu
contextMenu = C1CommandHolder1.GetC1ContextMenu(C1TextBox1)
MessageBox.Show(ContextMenu.Name)
```

C#

```
C#
//retrieves the contextmenu attached to the C1TextBox control
C1.Win.C1Command.C1ContextMenu contextMenu;
contextMenu = C1CommandHolder1.GetC1ContextMenu(C1TextBox1);
MessageBox.Show(ContextMenu.Name);
```

关联C1ContextMenu到一个NotifyIcon控件

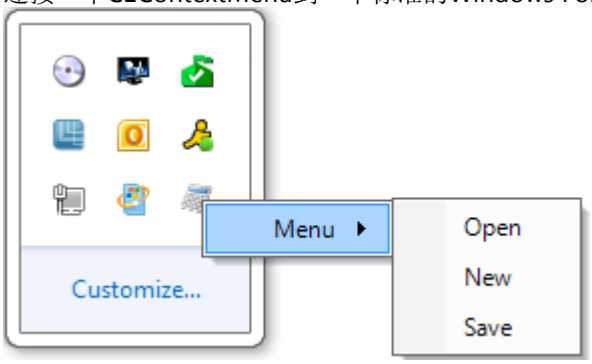
C1ContextMenu支持关联到标准NotifyIcon控件。

按以下步骤在设计时连接一个C1ContextMenu到一个NotifyIcon控件。

1. 在工具箱中双击C1ContextMenu控件，将它加入到组件托盘中。同时也会将一个C1ComponentHolder控件加入组件托盘中。
2. 右键点击C1ContextMenu控件，在上下文菜单中选择Edit。窗体上会显示一个New Command菜单项。
3. 右键点击New Command菜单项，在上下文菜单中选择Edit。弹出Link to Command编辑器。
4. 在Link to Command编辑器中设置如下命令属性：
   o Name文本框设置为MenuFile。
   o Text文本框设置为Menu。
5. 在Create a new command列表框中选择C1CommandMenu。
6. 在Link to Command编辑器中点击OK按钮，新创建的菜单显示在窗体上。
7. 右键点击菜单上的New Command项，在上下文菜单上选择Edit。弹出Link to Command编辑器。
8. 在Link to Command编辑器中设置如下属性：
   o Name文本框设置为OpenFile。
   o Text文本框设置为Open。
9. 在Visual Studio工具条上选择View | Properties。在Properties窗口上方的下拉列表中选择C1ContextMenu1。
10. 设置C1ContextMenu属性值为C1ContextMenu1。
11. 在工具箱的Windows Forms标签内双击NotifyIcon组件，将它添加到组件托盘中。
12. 右键点击在组件上，在上下文菜单中选择Choose an icon。指定一个该组件在运行时显示的图标。
13. 在Visual Studio工具条上选择View | Properties。在Properties窗口上方的下拉列表中选择notifyicon1。
14. 设置C1ContextMenu属性值为C1ContextMenu1以连接这两个组件。
15. 运行程序。指定的用来呈现NotifyIcon组件的图标显示在系统托盘中。注意如果右键点击该图标，将弹出上下文菜单。

此主题阐明

连接一个C1ContextMenu到一个标准的Windows Forms控件。



添加一个C1ContextMenu到C1DockingTab上

按如下步骤来添加一个C1ContextMenu控件到一个C1DockingTab控件上。

1. 在工具箱中找到C1ContextMenu控件，双击此控件将它添加到组件托盘中。一个C1CommandHolder组件将同时被添加到组件托盘中。
2. 通过C1ContextMenu智能标签打开C1ContextMenu的任务菜单。在任务菜单中选择Add Item，添加一个项目到C1ContextMenu中。在菜单中添加另外的两个项目。
3. 在窗体上添加一个C1DockingTab控件然后打开C1DockingTab的上下文菜单。在上下文菜单中选择Add Page。添加若干个页面到C1DockingTab中。在窗体上选择C1DockingTab控件以查看其属性。点击Events按钮并且找到MouseClick事件。输入c1DockingTab1_MouseClick以创建MouseClick事件处理程序。
4. 在窗体上右键点击然后在上下文菜单中选择View Code。在InitializeComponent()方法后插入下列代码：

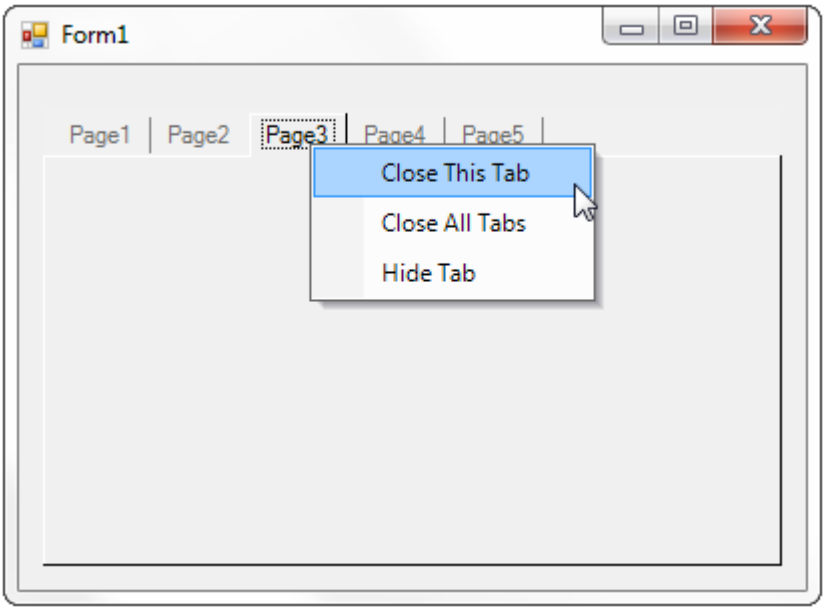
Visual Basic

```
Visual Basic
Private Sub c1DockingTab1_MouseClick(ByVal sender As Object, ByVal e As
MouseEventArgs)
    If (e.Button = System.Windows.Forms.MouseButtons.Left) Then
        If (e.X > Me.c1DockingTabPage1.Location.X) Then
            c1ContextMenu1.ShowContextMenu(Me.c1DockingTab1, New Point(e.X,
e.Y))
        Else
            Me.c1DockingTab1.ContextMenu = Nothing
        End If
    End If
End Sub
```

C#

```
C#
private void c1DockingTab1_MouseClick(object sender, MouseEventArgs e)
    {
        if (e.Button == System.Windows.Forms.MouseButtons.Left)
        {
            if (e.X > this.c1DockingTabPage1.Location.X)
            {
                c1ContextMenu1.ShowContextMenu(this.c1DockingTab1, new
Point(e.X, e.Y));
            }
            else
                this.c1DockingTab1.ContextMenu = null;
        }
    }
```

5. 按F5运行程序。注意当点击一个标签时将显示C1ContextMenu。



DockingTab任务

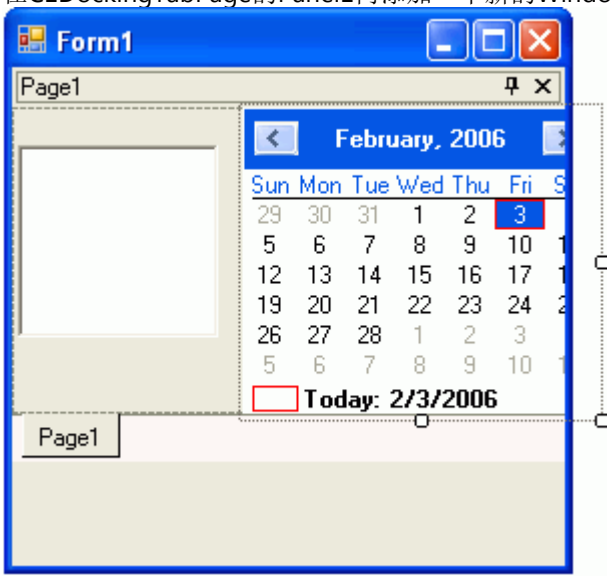
此章节描述如何执行特定的停靠标签任务。

添加滚动条到C1DockingTab中

按以下步骤为C1DockingTab添加一个滚动条：

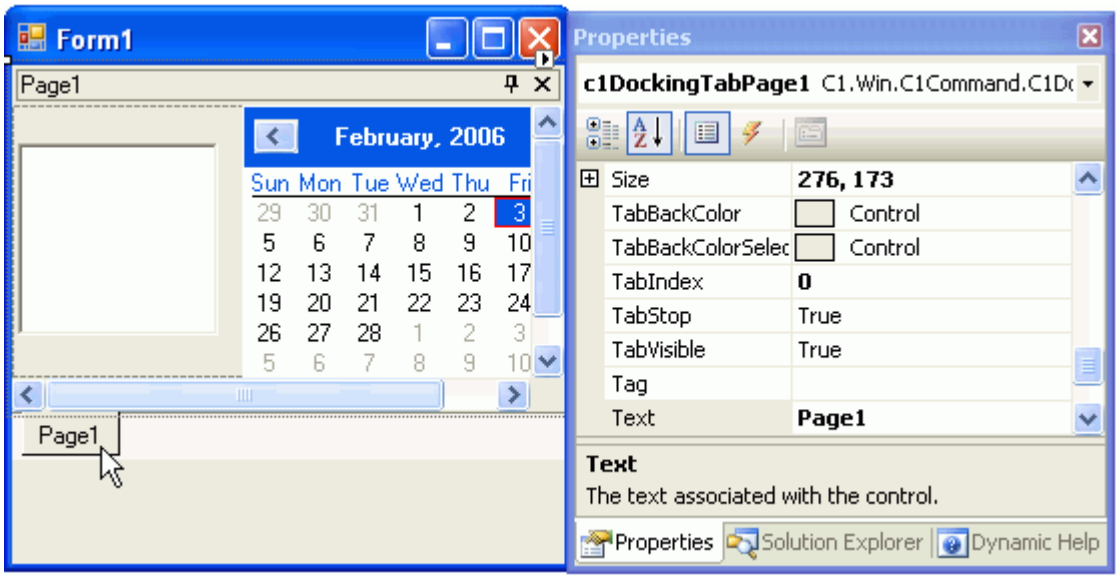
> 注：C1DockingTab没有内置的滚动条。

1. 添加一个C1CommandDock控件到窗体上，然后设置其Dock属性值为Fill。
2. 在C1CommandDock内添加一个C1DockingTab。
3. 在C1DockingTabPage内添加一个Panel控件，然后设置此Panel的Dock属性值为Fill。
4. 在C1DockingTabPage的Panel1内添加一个新的Windows Form控件。



5. 设置Panel1的AutoScroll属性值为True。

滚动条显示在C1DockingTab中。



关闭一个C1DockingTabPage

可以在代码中使用C1DockingTabPage的TabVisible属性来关闭C1DockingTabPage。以下代码用来关闭第一个页面。

Visual Basic

```
Me.C1DockingTab1.TabPages(0).TabVisible = False
```

C#

```
this.C1DockingTab1.TabPages(0).TabVisible = False;
```

确定C1DockingTab是否浮动

使用C1DockingTab的Floating属性（仅运行时有效）来确定停靠的标签当前是否是浮动的。

Visual Basic

```
Public Floating As Boolean
```

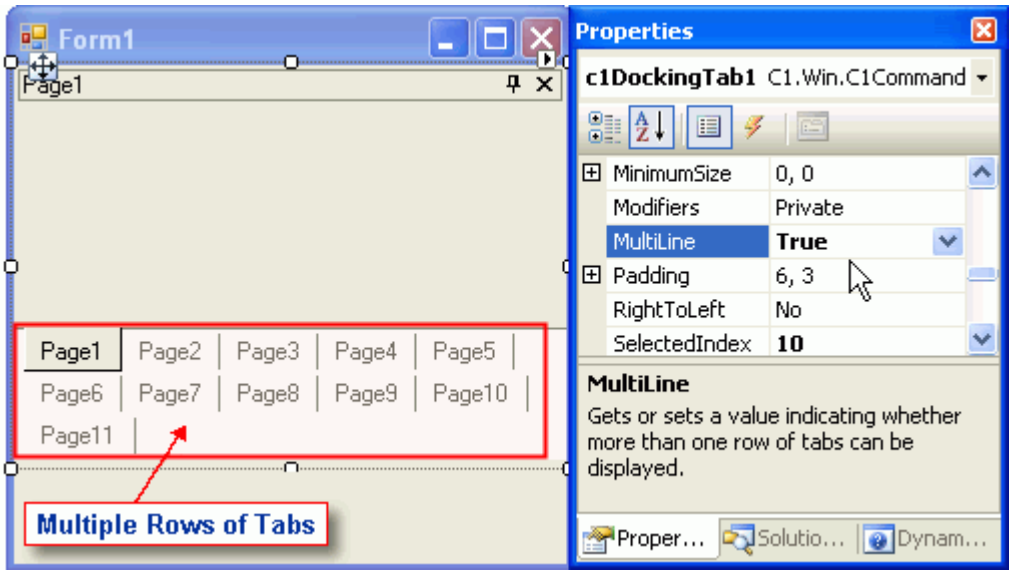
C#

```
Public bool Floating {get;}
```

显示多个标签行

设置C1DockingTab类的MultiLine属性值为True，以在一个C1DockingTab中以多行方式显示标签。

1. 设置C1DockingTab的MultiLine属性值为True。
2. 在C1DockingTabPage的Page1上右键点击，然后在上下文菜单中选择Add Page。在C1DockingTab上添加若干个页面。标签连续的显示在另外的行上。



在每一个C1DockingTabPage中显示相同的控件组

为了避免用复制的方式实现将第一个C1DockingTabPage中的控件组显示在其他的C1DockingTabPage中，请执行如下步骤：

1. 在第一个C1DockingTabPage中添加一个Panel，然后设置其Dock属性值为Fill以使其充满整个C1DockingTabPage。
2. 在面板中添加一些控件。
3. 在SelectedIndexChanging事件处理程序中将前一个停靠标签页面中的控件移动到新的页面中。

Visual Basic

```
Visual Basic
Me.dockingTab.SelectedTab.Controls.Add (myPanel)
```

C#

```
C#
this.dockingTab.SelectedTab.Controls.Add (myPanel);
```

使DockingTab可停靠和浮动

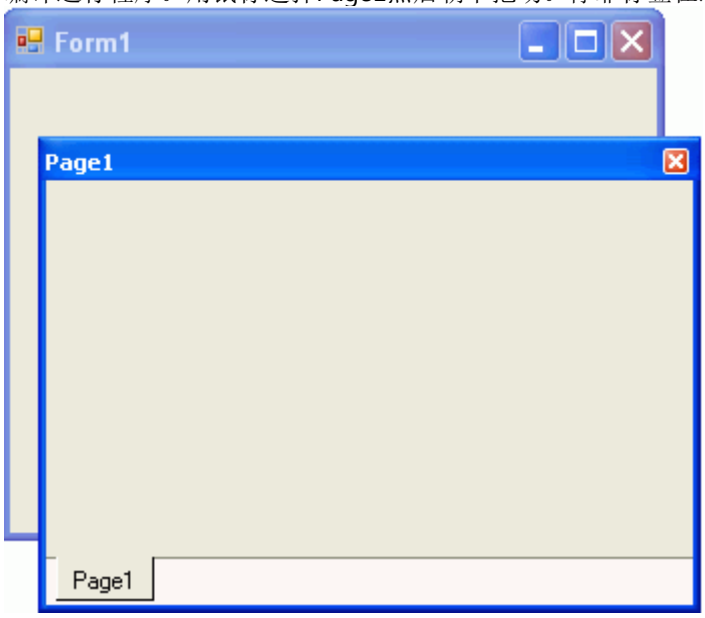
按如下基本操作将一个C1DockingTab添加到窗体中：

1. 使用拖拽方式将C1CommandDock控件添加到窗体上。
2. C1CommandDock默认停靠在窗体的左方。选择C1CommandDoc的Dock属性上的下拉箭头，然后点击上方的矩

形框。这将会让C1CommandDock控件停靠在窗体的上方。

3. 使用拖拽方式将C1DockingTab控件添加到C1CommandDock中。窗体的显示如下图所示：

4. 编译运行程序。用鼠标选择Page1然后朝下拖动。停靠标签在运行时的显示如下图所示：



注意：可以使用C1CommandDock的FloatHide属性控制C1DockingTabPage在运行时的行为。此属性允许选择在程序失去焦点的时候让标签页面保持焦点状态。C1CommandDock的FloatHide属性有三个可选设置：Default, Never和FocusLost。

启用或禁用焦点提示

可以容易的使用C1DockingTab的TabsShowFocusCues属性来启用或禁用C1DockingTab的焦点提示。默认情况下，这个属性值为True。此主题涵盖了在设计时和通过代码设置这个属性值为false。

在设计时

在Properties窗口，设置C1DockingTab的TabsShowFocusCues属性值为false。

通过代码

Visual Basic

```
Visual Basic
c1DockingTab1.TabsShowFocusCues = false
```

C#

```
C#
c1DockingTab1.TabsShowFocusCues = false;
```

加载和保存C1DockingTab的布局

可以用SaveLayout和RestoreLayout方法来加载和恢复C1DockingTab的布局。

Visual Basic

```
Visual Basic
'Saves the current layout of the tabs on the form to the specified file.
Shared Sub SaveLayout(form As Form, filename As String)

'Restores the previously saved layout of the tabs on the form from the specified file
Sub RestoreLayout(form As Form, filename As String)
```

C#

```
C#
//Saves the current layout of the tabs on the form to the specified file.
static void SaveLayout(Form form, string filename);

//Restores the previously saved layout of the tabs on the form from the specified
file
Static void RestoreLayout(Form form, string filename);
```

下列代码演示了如何应用这些方法：

Visual Basic

```
Visual Basic
//Saves the current layout of the tabs on the form to the specified file.
C1DockingTab.SaveLayout(myForm, "myLayoutFile.xml")

'Restores the previously saved layout of the tabs on the form from the specified file
C1DockingTab.RestoreLayout(myForm, "myLayoutFile.xml")
```

C#

```
C#
//Saves the current layout of the tabs on the form to the specified file.
C1DockingTab.SaveLayout (myForm, "myLayoutFile.xml");
```

```
//Restores the previously saved layout of the tabs on the form from the specified
file
C1DockingTab.RestoreLayout(myForm, "myLayoutFile.xml");
```

运行时移动标签页面

使用CanMoveTabs属性来移动标签到不同的位置。在Properties窗口设置CanMoveTabs属性值为True。

固定C1DockingTab

以编程的方式固定一个C1DockingTab，设置C1DockingTab的AutoHiding属性值为False。使用下列代码：

Visual Basic

```
Visual Basic
Me.C1DockingTab1.AutoHiding = False
```

C#

```
C#
this.C1DockingTab1.AutoHiding = False;
```

注意：为了正确的固定和取消固定一个C1DockingTab，必须将它放置在一个C1CommandDock控件中。

阻止标签在鼠标点击时获取焦点

使用TabsCanFocus属性来阻止标签获取焦点。设置这个属性值为False将阻止标签在鼠标点击的时候获取焦点。

限定特定标签的使用

可以添加SelectedIndexChanging事件处理程序来检测是否将要切换的页面是不想让用户操作的。如果答案是肯定的，设置e.Cancel值为True。例如，如下代码演示了如何添加SelectedIndexChanging事件处理程序。

Visual Basic

```
Visual Basic
Private Sub c1DockingTab1_SelectedIndexChanging(sender As Object, e As
C1.Win.C1Command.SelectedIndexChangingEventArgs)
   If e.NewIndex = 1 And e.CanCancel Then
      e.Cancel = True
   End If
End Sub
```

C#

```
C#
private void c1DockingTab1_SelectedIndexChanging(object sender,
C1.Win.C1Command.SelectedIndexChangingEventArgs e)
```

```
{
   if(e.NewIndex == 1 && e.CanCancel)
      e.Cancel = true;
}
```

NavBar任务

本章节主要展示如何实现特殊导航条任务。

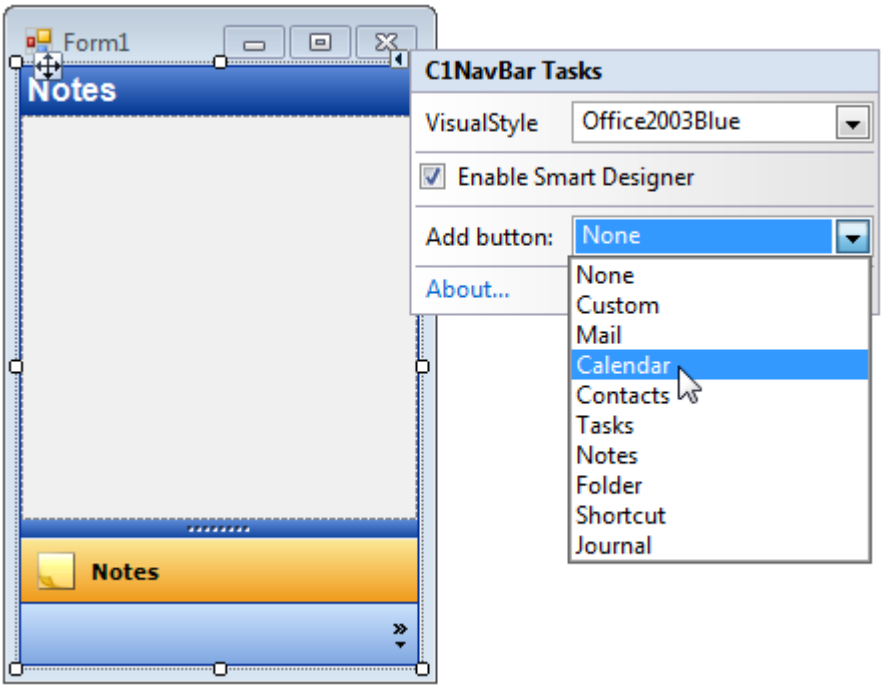
添加一个面板

本节中，你将添加一个面板到C1NavBar控制器，这个过程将会使用到智能标签以及浮动工具条。

使用智能标签

完成以下步骤：

1. 找到工具栏，然后双击C1NavBar按钮。C1NavBar控制器将会添加到表单中。你会注意到一个叫做Notes的面板，缺省情况下将会在控制器中显示。
2. 单击C1NavBar智能标签(▶)，打开C1NavBarTasks菜单。
3. 在C1NavBarTasks菜单中，单击Addbutton下拉按钮，然后选择一种面板类型。例如，选择Calendar类型。

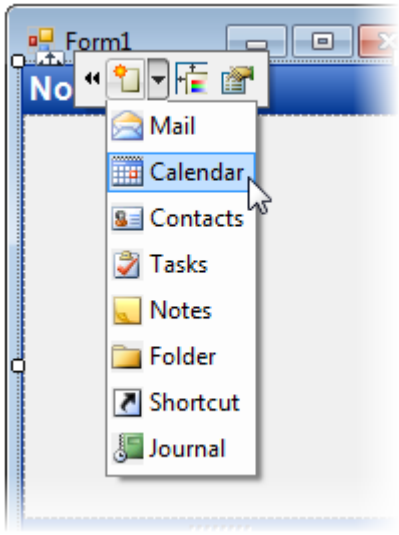


使用浮动工具条

完成以下步骤：

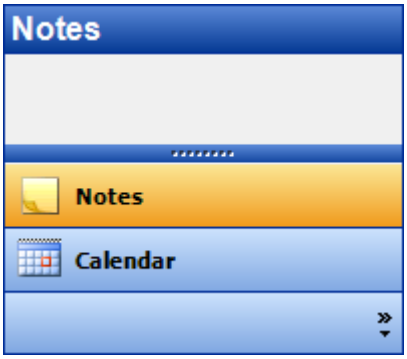
1. 找到工具栏，然后双击C1NavBar按钮。C1NavBar控制器将会添加到表单汇总，你会注意到一个叫做Page1的页面，缺省情况下将会出现在控制器中。
2. 将鼠标的光标在控制器上盘旋，激活浮动工具条。浮动工具条将会出现在页面上，效果如下所示：

3. 在**Adding button and corresponding panel** 下拉列表中，选择一个面板类型。本例中，将选择Calendar类型。



本节效果如下所示：

本节的实现效果如下所示：



创建一个面板页眉

在本节中，你将会学习到如何添加一个页眉到面板中，这个过程将会使用到智能标签以及浮动工具条。

使用智能标签

完成以下步骤：

1. 选择你想要添加页眉的面板，然后单击它的智能标签(▶)。这里将会打开C1NavBarPanelTasks菜单。
2. 从C1NavBarPanelTasks菜单中，选择添加区域页眉。

3. 选择页眉，c1NavBarSectionHeader1，然后在Text属性窗口，使用一个字符串设置它的属性。本例中，将文本属性设置为“HelloWorld“。

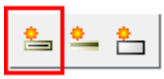
使用浮动工具条

完成以下步骤：

1. 将鼠标的光标在控制器上盘旋，激活浮动工具条。浮动工具条展示效果如下图所示：

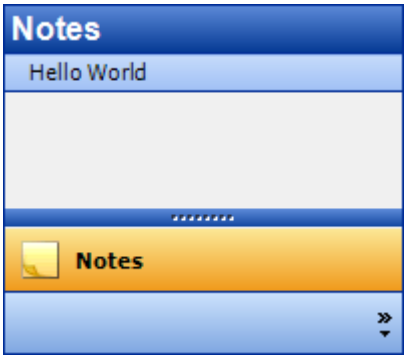
C1NavBar控制器将被添加到表单中。你会观察到一个叫做Page1的页面，在缺省情况下会出现在控制器中。

2. 在浮动工具条中，选择Addsectionheader按钮。



3. 选择页眉，c1NavBarSectionHeader1，然后在属性窗口中，使用字符串设置它的Text属性，本例中，设置它的文本属性为"Hello World"。

本节实现效果如下所示：



在折叠面板中使用垂直文本

当一个面板被缩小时，你可以强制C1NavBar控制器显示垂直文本。是否显示垂直文本，是由NavBarCollapsedBarTextUIString决定的。想要实现这一功能，你将需要设置ShowVerticalTextOnCollapse属性为True。

完成以下步骤：

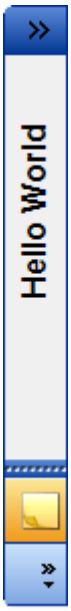
1. 右键单击C1NavBar控制器，打开它的上下文菜单。然后选择属性选项。这将打开属性窗口。
2. 在属性窗口中，完成下面的步骤：

   - 将AllowCollapse属性设置为True。这将允许你在运行时缩小控制器，从而实现在缩小面板中显示垂直文本的功能。
   - 将ShowVerticalTextOnCollapse属性设置为True。
   - 打开UIStrings节点，使用字符串设置UIStrings.NavBarCollapsedBarText明细的内容，然后在NavBarCollapsedBarText文本输入框中输入文本。

3. 按F5，运行项目

本节效果如下所示：

当项目运行时，单击按钮来收缩控制器。收缩面板上的文本显示效果如下所示：



OutBar任务

本节将要展示如何实现特殊的边条任务。

定制C1OutPages标题

在本节中，你将学习到如何定制C1OutBar控制器页面的标题区域。你将创建一个包含三个C1OutPage页面的C1OutBar控制器，设置一部分属性，然后在工程中添加代码，从而为每个标题实现定制颜色。

完成下面的步骤：

1. 将一个C1OutBar控制器添加到你的表单中。
2. 将C1OutPage组件添加到C1OutBar控制器中。
3. 设置以下属性：
   - 将c1OutBar1的VisualStyle属性设置为Classic。如果你需要，你同样可以选择Custom类型。剩余的视觉效果将不显示在自定义标题的绘制过程中。
   - 将c1OutPage1的OwnerDraw属性设置为True。
   - 将c1OutPage2的OwnerDraw属性设置为True。
   - 将c1OutPage3的OwnerDraw属性设置为True。
4. 在属性窗口中，从下拉列表中选择c1OutBar1选项。单击Events按钮，然后双击DrawPage事件来添加DrawPage事件。
5. 将下面的命名空间导入到工程中：

   Visual Basic

   ```
   Visual Basic
   Imports C1.Win.C1Command
   ```

   C#

   ```
   C#
   ```

```
using C1.Win.C1Command;
```

6. 将下面的代码添加到DrawPage事件句柄中。

Visual Basic

Visual Basic

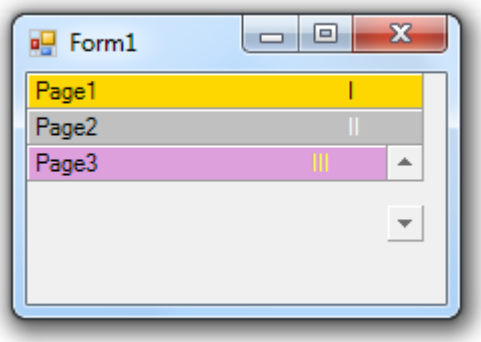
```
'e.page parameter determines the page
If e.Page Is c1OutPage1 Then
   e.Graphics.FillRectangle(Brushes.Gold, e.Bounds)
   e.Graphics.DrawString("I", c1OutBar1.Font, Brushes.Black, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
ElseIf e.Page Is c1OutPage2 Then
   e.Graphics.FillRectangle(Brushes.Silver, e.Bounds)
   e.Graphics.DrawString("II", c1OutBar1.Font, Brushes.White, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
ElseIf e.Page Is c1OutPage3 Then
   e.Graphics.FillRectangle(Brushes.Plum, e.Bounds)
   e.Graphics.DrawString("III", c1OutBar1.Font, Brushes.Yellow, New
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2))
End If
```

C#

C#

```
\\e.page determines the page
If (e.Page == c1OutPage1)
{
    e.Graphics.FillRectangle(Brushes.Gold, e.Bounds);
    e.Graphics.DrawString("I", c1OutBar1.Font, Brushes.Black, new
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2));
}
else if (e.Page == c1OutPage2)
{
    e.Graphics.FillRectangle(Brushes.Silver, e.Bounds);
    e.Graphics.DrawString("II", c1OutBar1.Font, Brushes.White, new
PointF(e.Bounds.Right -40, e.Bounds.Top + 2));
}
else if (e.Page == c1OutPage3)
{
    e.Graphics.FillRectangle(Brushes.Plum, e.Bounds);
    e.Graphics.DrawString("III", c1OutBar1.Font, Brushes.Yellow, new
PointF(e.Bounds.Right - 40, e.Bounds.Top + 2));
}
```

7. 按F5运行工程，你会观察到标题已经实现自定义效果了。最后的产品效果如下所示：

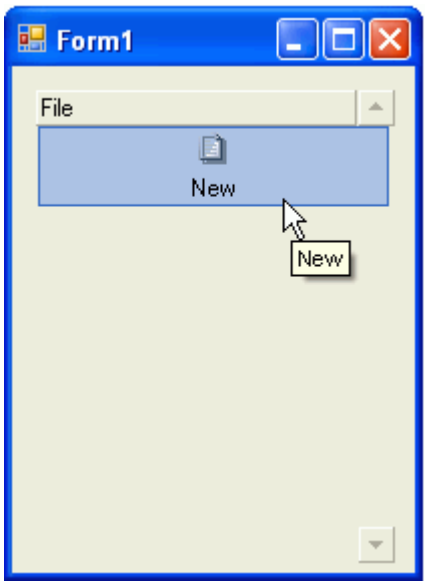


创建并且配置C1OutBar控制器

1. 使用拖放操作将一个C1OutBar控制器放到你的表单中。最新创建的C1OutBar控制器初始化包含一个C1ToolBar控制器的独立页面。（C1OutBar页面能够包含任意的C1ToolBar控制器，在那种情况下，将需要特殊对待。它还能够包含其他任意控制器，例如选项卡控制器的页面）。同样的，一个C1CommandHolder将会自动创建，并且添加到组件托盘中。
2. 从属性窗口的下拉列表中选择C1OutPage1选项。将它的Text属性从Page1改为File。
3. 右键单击C1ToolBar上的（也是唯一的）按钮，然后从上下文菜单中选择AppendItem选项。这里将会打开**Link to Command**对话框。
4. 在**Link to Command**设计器中，设置如下的常用属性。

   - TexttoNew
   - NametocmdFileNew

5. 从创建的新命令列表栏中选择C1Command。
6. 选择OK。
7. 在属性窗口的下拉列表中选择C1CommandLink1选项。然后设置ButtonLook属性为TextAndImage。
8. 选择cmdFileNew选项，然后找到图片属性。单击ellipsis按钮，然后找到需要的图片
9. 选择图片，然后在SelectResource对话框中单击OK。新图片将显示在C1CommandLink1的文本上方。
10. 生成，并且运行应用。运行效果应该近似下图所示：



添加C1OutPage 到C1OutBar中

通过编程添加一个新的C1OutPage到C1OutBar中，完成以下步骤：

1. 添加一个C1OutBar控制器到表单中
2. 导入C1.Win.C1Command命名空间。使用下述代码导入C1.Win.C1Command命名空间。

Visual Basic

```
Imports C1.Win.C1Command
```

C#

```
using C1.Win.C1Command;
```

3. 添加一个新的C1OutPage到C1OutBar中。最后一行代码使用Add方法添加了一个新的outpage1到C1OutBar1。在Form_Load事件中输入下述代码：

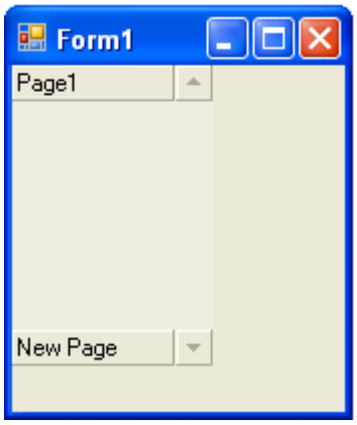
Visual Basic

```
Dim outpage1 As New C1.Win.C1Command.C1OutPage()
Outpage1.Text = "New Page"
Me.c1OutBar1.Pages.Add(outpage1)
```

C#

```
C1.Win.C1Command.C1OutPage outpage1 = new C1.Win.C1Command.C1OutPage();
outpage1.Text = "New Page";
this.c1OutBar1.Pages.Add(outpage1);
```

4. 生成并且运行你创建的C1OutBar新页面。新的页面效果应该如下图所示：

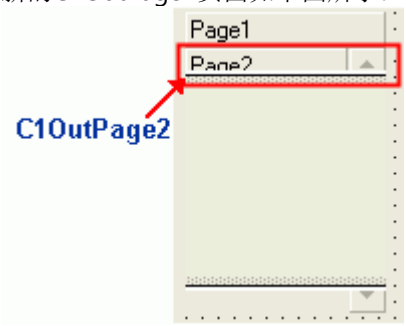


添加多个C1OutPages到C1OutBar中

想要在设计时添加一个新的C1OutPage到新的C1OutBar控制器中，完成以下步骤：

1. 从工具栏中将一个C1OutBar组件通过拖放操作放置到你的表单中。这个C1OutBar将包含一个C1OutPage控制器以及一个C1OutBar控制器。

2. 右键单击C1OutPage1，然后从它的上下文菜单中选择AddPagewithToolBar选项。
3. 新的C1OutPage2页面如下图所示：



需要注意的是，你可以创建若干个C1OutPages，通过在C1OutPages上右键单击的方式，然后在它的上下文菜单中选择AddPagewithToolbar选项。

修改C1OutBar的外观

想要在设计时修改C1OutBar的外观，完成以下步骤：

1. 从工具栏中将一个C1OutBar组件通过拖放操作放置到你的表单中。这个缺省的C1OutBar将包含一个内置C1ToolBar控制器的C1OutPage控制器。
2. 在C1OutBar控制器的属性窗口中，从它的属性窗口下拉菜单中选择C1OutPage1选项。
3. 设置Text属性，将它的属性“Page1”从改为“GoingPlaces”。
4. 右键单击表单中的C1ToolBar控制器，然后从它的上下文菜单中选择AppendItem选项。一个新的命令以及LinktoCommand设计器将会出现。
5. 在LinktoCommand设计器中，设置如下属性：

   - Text："AirTickets"
   - Name："cmdGPAirTicket"

6. 从Createanewcommand列表栏中选择C1Command选项，然后单击OK关闭对话框。

添加文本和图片到命令连接

1. 从属性窗口的下拉列表中选择C1CommandLink1选项，然后将ButtonLook属性设置为TextAndImage。
2. 在C1CommandLink1的属性窗口中，展开命令属性的节点，然后在图片属性字段中单击按钮。Open对话框将会弹出。
3. 浏览找到一个图片，将其添加到机票命令连接中。新的图片将在机票命令连接中显示。

添加一个新的命令连接

1. 右键单击工具条上的机票命令连接，从它的上下文菜单中选择AppendItem选项，一个新的命令连接将显示在工具条上。
2. 在LinktoCommand设计器上，设置如下属性：
   - Text：CityGuides
   - Name：cmdGPCityGuides
   - 将新创建的命令列表框中的命令类型改为C1Command。
3. 选择OK来应用改变，然后关闭LinktoCommand设计器。

添加一个文本和图片到命令连接

1. 在工具条中选择C1CommandLink2,CityGuides选项，然后将它的ButtonLook属性设置为TextAndImage。

2. 展开CommandLink2的属性窗口中的命令属性节点，然后图片属性字段的ellipses按钮。选择需要的图片，然后单击对话框中的Open按钮。新图片将出现在C1CommandLink2文本的上方。

修改C1OutPage1 和C1ToolBar1的外观属性

1. 从属性窗口的下拉列表中选择选项，按照下述内容修改它的属性。
    - 展开Font属性，将Bold属性设置为True。
    - 选择BorderStyle属性，然后从它的下拉列表中选择FixedSingle选项。
2. 在表单中选择选项，然后按照下述内容设置它的属性。
    - BackColor：White
    - BackHiColor：SkyBlue
3. 在属性的下拉列表中选择C1OutBar1选项，然后按照下述内容设置它的属性：

    - BackColor ：#CC99FF
    - Font.Size：9
    - Bold：True

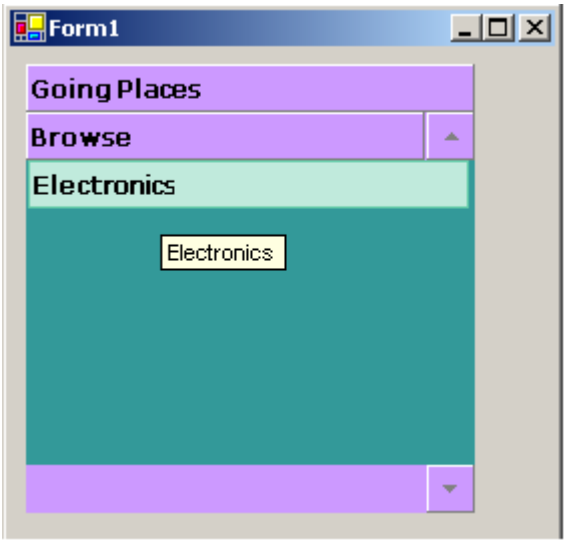
添加一个新的C1OutPage

1. 右键单击GoingPlaces，然后从它的上下文菜单中选择AddPagewithToolbar选项。一个新的C1OutPage2页面将会出现在C1OutPage1,GoingPlaces的下方。
2. 从属性窗口的下拉列表中选择C1OutPage2选项，然后设置它的Text属性为Browse。
3. 在表单上的C1ToolBar2中右键单击，然后从它的上下文菜单中选择AppendItem选项。这里将会打开LinktoCommand设计器。
4. 在设计器中设置如下属性：
    - Text：Electronics
    - Name：cmdBrowseElectronics
5. 从新创建的命令连接列表栏中选择C1Command选项，然后单击OK关闭设计器。
6. 在工具条上选择C1CommandLink3选项，然后将它的ButtonLook属性设置为Text。
7. 在属性窗口的下拉列表中选择C1ToolBar2选项，然后按照下述内容设置它的属性：

    - BackColor：Teal
    - BackHiColor：MediumAquamarine
    - ButtonAlign：Near

8. 生成并运行应用

    C1OutBar的浏览外部页面运行时效果如下图所示：



TopicBar任务

本章节展示怎么执行特定的标题栏任务。

添加和移除TopicBar项目

下列主题说明如何从C1TopicBar控件添加和移除主题页和主题链接。

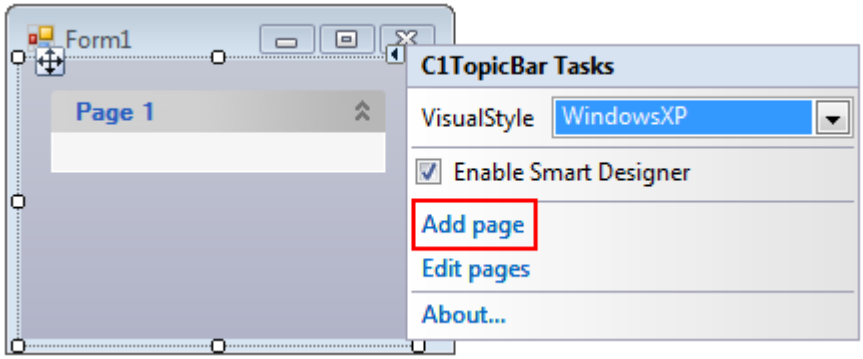
向TopicBar添加主题页

有5种方向 topic bar添加主题页：通过智能标签，浮动工具条，上下文菜单，集合编辑器，或代码。通过本章节能够学习通过以上五种方法向topic bar添加主题页。

使用智能标签

完成以下步骤:

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在控件上。
2. 点击C1TopicBar智能标签(▶) 打开**C1TopicBar Tasks** 菜单.
3. 在**C1TopicBar Tasks** 菜单上, 点击 **Add Page**.

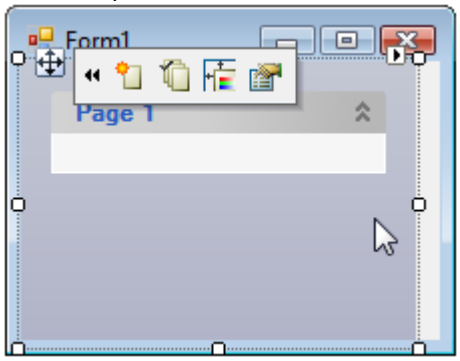


Page 2 被添加到C1TopicBar 控件上。

使用浮动工具条

完成以下步骤:

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。注意默认名称为Page 1的页面出现在控件上。
2. 点击C1TopicBar控件按钮激活浮动工具条。浮动工具条出现在页面上如下所示：



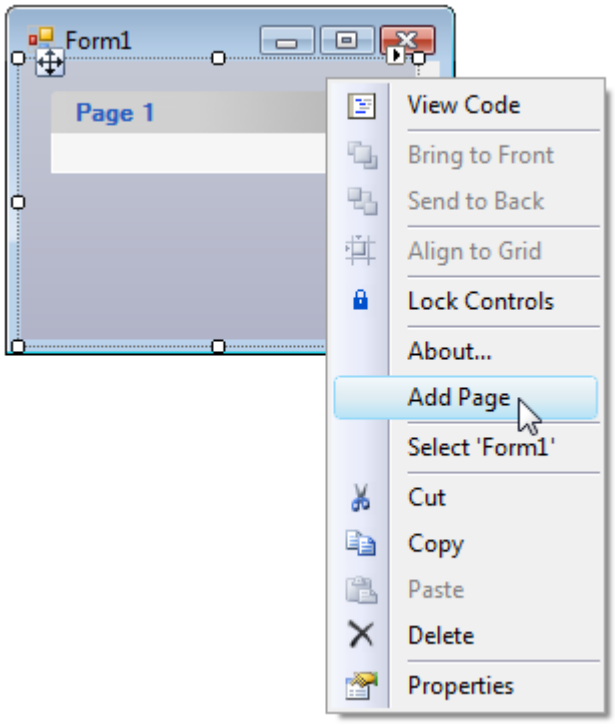
3. 在浮动工具条上，选择**Add topic page** 按钮 .

   Page 2 被添加到C1TopicBar 控件上

使用上下文菜单

完成以下步骤:

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。注意默认名称为Page 1的页面出现在控件上。
2. 右击C1TopicBar控件打开其上下文菜单。
3. 从上下文菜单上选择**Add Page**.

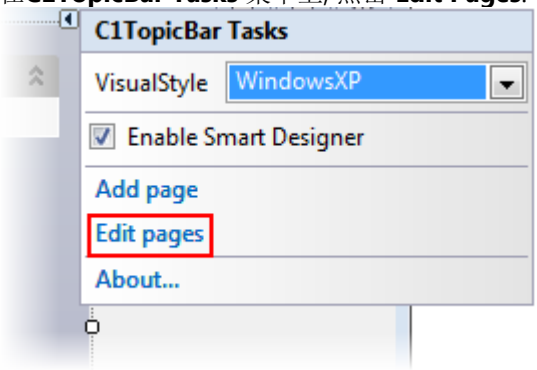


Page 2 被添加到C1TopicBar 控件。

使用集合编辑器

完成以下步骤:

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。注意默认名称为Page 1的页面出现在控件上。
2. 点击C1TopicBar智能标签(▶) 打开**C1TopicBar Tasks** 菜单.
3. 在**C1TopicBar Tasks** 菜单上, 点击 **Edit Pages**.



The **C1TopicPage Collection Editor** opens.

4. Click **Add** to add a page to the collection.
5. Click **OK** to close the **C1TopicPage Collection Editor**.

Page 2 被添加到C1TopicBar 控件。

使用代码

完成以下步骤:

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。注意默认名称为Page 1的页面出现

在控件上。

2. 双击表单空白区域来打开代码视图，注意**Form_Load** 事件已经被添加。
3. 在工程里导入以下命名空间：

Visual Basic

```
Imports C1.Win.C1Command
```

C#

```
using C1.Win.C1Command;
```

4. 在**Form_Load** 事件里添加如下代码，目的是创建新的主题页并添加到topic bar里面，

Visual Basic

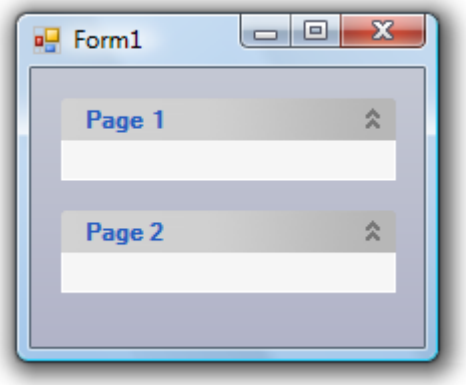
```
'Create new topic page object named "Page 2"
Dim c1TopicPage2 As New C1TopicPage("Page 2")
'Add new topic page to topic
c1TopicBar1.Pages.Add(c1TopicPage2)
```

C#

```
//Create new topic page object named "Page 2"
C1TopicPage c1TopicPage2 = new C1TopicPage("Page 2");
//Add new topic page to topic
c1TopicBar1.Pages.Add(c1TopicPage2);
```

5. 按下F5编译工程，请注意一个新的页面Page 2出现在了控件上。

本章节演示如下：



从TopicBar移除主题页

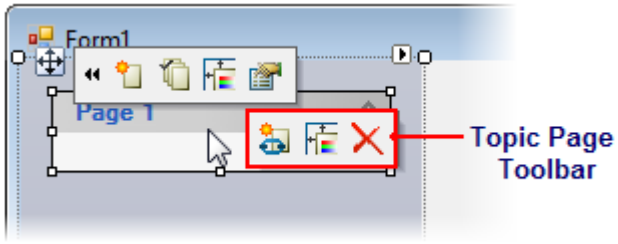
总共有三种方式从topic bar移除页面：可以使用工具条，集合编辑器，代码。从下面的主题中，你可以学到如何使用三

种方法中的任何一种从topic bar移除页面。

使用浮动工具条

完成以下步骤：

1. 导航到Toolbox，双击C1TopicBar图标，C1TopicBar控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在控件上。
2. 在Page 1上滑动鼠标直到浮动工具条出现，主题页的工具条显示如下：



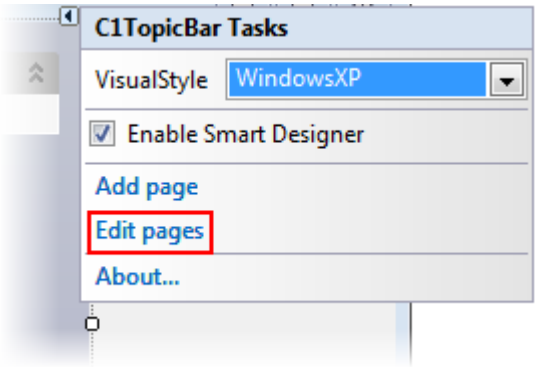
3. 点击Delete topic page按钮  .

主题页从控件上被移除。

使用集合编辑器

完成如下步骤：

1. 导航到Toolbox并双击C1TopicBar图标，C1TopicBar控件被添加到表单上。可以看到命名为Page 1的页面默认出现在控件上。如果你想在控件上添加更多的页面请看为TopicBar添加主题页.
2. 点击C1TopicBar智能标签( ) ，打开C1TopicBar Tasks菜单。
3. 在C1TopicBarTasks菜单上，点击Edit Pages.



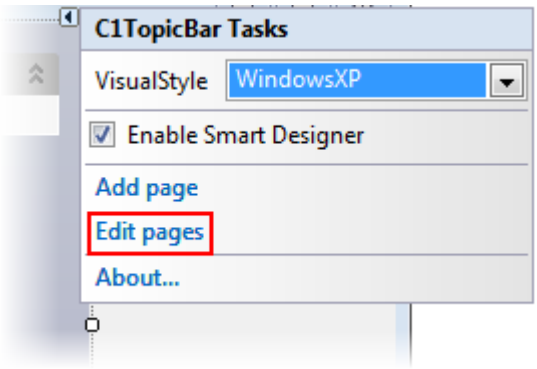
C1TopicPage Collection Editor打开。

4. 在Members 方框里，选择你想移除的页。
5. 点击Remove移除该页。
6. 点击OK 关闭C1TopicPage Collection Editor。
   页面从控件被移除。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar图标，C1TopicBar 控件被添加到了表单上。可以观察到命名为Page 1的页默认出现在控件上。如果你想要在控件上添加更多的页面，请看Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar的智能标签(▶)，打开C1TopicBar Tasks菜单。
3. 在C1TopicBar Tasks菜单上，点击Edit Pages。



C1TopicPage 集合编辑器打开。

4. 选择你想移除的页面，设置该页的Tag 属性为"Tag1"。添加的这个标签给该页面唯一的指示器是为了将来允许你通过FindPageByTag 方法可以找到该页面。
5. 点击OK关闭C1TopicPage 集合编辑器。
6. 双击表单上空白部分并打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图上。
7. 添加下面的代码，可以找到该页面并移除它，参见Form_Load事件：

▸ Visual Basic

```
'Find the page and assign it to a variable
Dim Page1 = c1TopicBar1.FindPageByTag("Tag1")
'Remove the specified page
c1TopicBar1.Pages.Remove(Page1)
```

▸ C#

```
//Find the page and assign it to a variable
var Page1 = c1TopicBar1.FindPageByTag("Tag1");
//Remove the specified page
c1TopicBar1.Pages.Remove(Page1);
```

8. 按下F5新建工程观察已经被移除的页面。

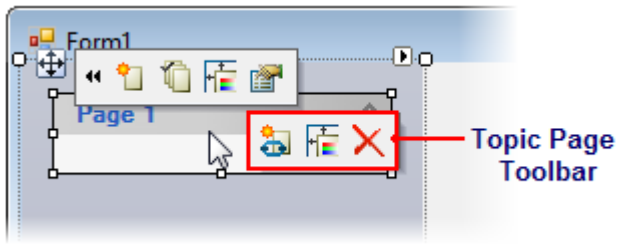
添加主题链接到主题页

有三种方法可以添加主题链接到主题页面上：可以使用浮动工具条，集合编辑器，或者代码，在这个主题中，你将学到使用三种方法中的任何一种添加主题链接到主题页面上。

使用浮动工具条

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。可以观察到命名为Page 1的页面默认出现在控件上。
2. 鼠标在Page 1滑动直到浮动工具条出现。主题页的主题条如下出现：



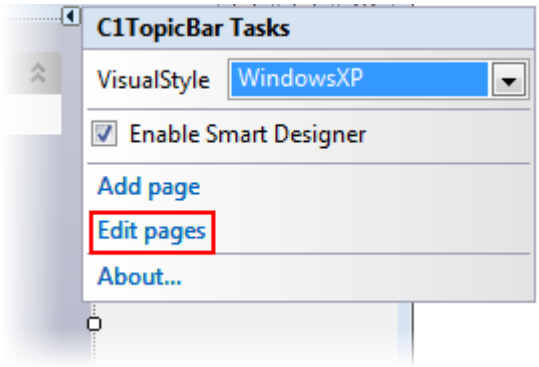
3. 点击Add topic link按钮  .

主题链接Link 就会被添加到主题页上。

使用集合编辑器

完成如下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到表单上。可以观察到一个命名为Page 1的页面默认出现在控件上。
2. 点击C1TopicBar的智能标签( )，打开C1TopicBar Tasks 菜单。
3. 在C1TopicBar Tasks 菜单上，点击Edit Pages。



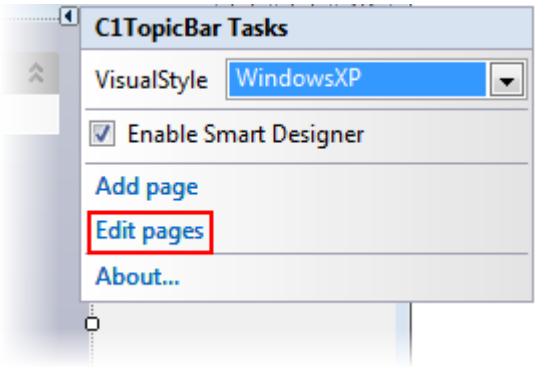
C1TopicPage Collection Editor打开。

4. 在Members方框里选择一个页面。
5. 在Properties网格里，鼠标放置到Links属性，点击省略号按钮 。C1TopicLink Collection Editor打开。
6. 点击Add添加主题链接到主题页。
7. 点击close关闭C1TopicLink Collection Editor。
8. 点击close关闭C1TopicPage Collection Editor。
主题链接Link 就会被添加到主题页上。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单。可以观察 到命名为Page 1的页面默认出现在控件上。如果你想添加更多页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar的智能标签(▶)，打开C1TopicBar Tasks 菜单。
3. 在C1TopicBar Tasks 菜单上，点击 Edit Pages。



C1TopicPage Collection Editor 打开。

4. 选择你想移除的页面，然后设置该页面的Tag属性为"Tag 1"。添加该标签是为了给该页面唯一的指示器以便将来允许你可以通过FindPageByTag 方法找到找到该页。
5. 点击OK关闭C1TopicPage Collection Editor。
6. 双击双击表单上空白部分并打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图上。
7. 引入以下命名空间到工程：

▶ Visual Basic

Visual Basic
```
Imports C1.Win.C1Command
```

▶ C#

C#
```
using C1.Win.C1Command;
```

8. 为Form_Load事件添加以下代码，这个代码可以找到该页，并指定它为一个变量，创建新的主题链接对象，并添加主题链接到主题页。

▶ Visual Basic

Visual Basic
```
'Find the topic page and assign it to a variable
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("Tag1")
'Create a new topic link and assign it a name
Dim c1TopicLink1As New C1TopicLink("Link 1")
'Add the new topic link to the topic page
c1TopicPage1.Links.Add(c1TopicLink1)
```

▶ C#

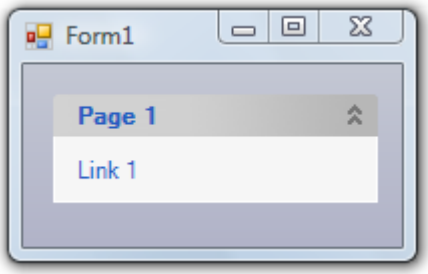
C#
```
//Find the topic page and assign it to a variable
var c1TopicPage1 = c1TopicBar1.FindPageByTag("Tag1");
//Create a new topic link and assign it a name
```

```
C1TopicLink c1TopicLink1 = new C1TopicLink("Link 1");
//Add the new topic link to the topic page
c1TopicPage1.Links.Add(c1TopicLink1);
```

9. 按F5运行工程，就能观察到一个链接已经被添加到了该页面。

主题说明如下：

在此主题里，你学到了如何使用三种方法中的任何一种添加主题链接到主题页面上。此主题的结果展示如下：



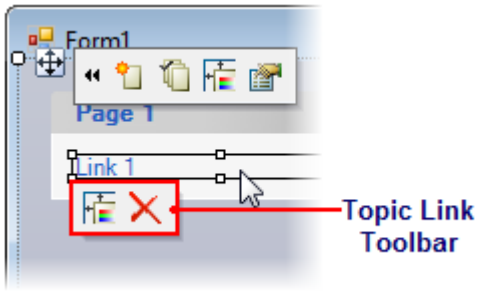
从主题页移除主题链接

你可以使用浮动工具条，集合编辑器，代码，从主题页面移除主题链接，此主题中，你将学习到如何使用上述方法中的任何一种移除主题链接，该主题假定你有一个C1TopicBar 空间包含至少一个主题链接（参见添加主题链接到主题页面）在你的页面上。

使用浮动工具条

完成以下步骤：

1. 使用鼠标，移动到你想移除的主题链接直到它的浮动主题条出现。该主题链接的工具条类似如下：

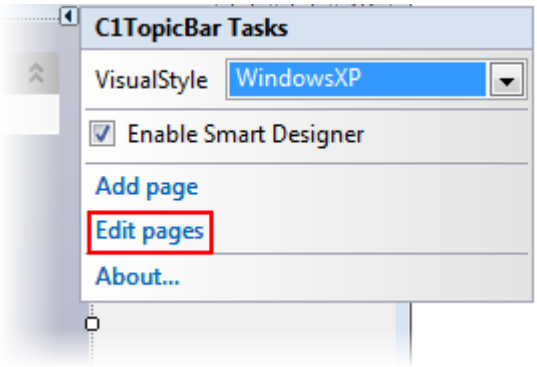


2. 点击Delete topic link按钮  。该主题链接从主题页被移除。

使用集合编辑器

完成以下步骤：

1. 点击C1TopicBar的智能标签()，打开C1TopicBar Tasks 菜单。
2. 在C1TopicBar Tasks菜单上，点击 Edit Pages，



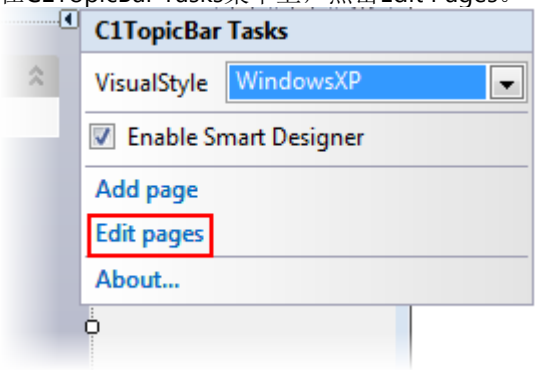
将会打开C1TopicPage Collection Editor。

3. 在Members方框里选择一个页面。
4. 在Properties网格里，鼠标放置在Links属性并点击省略号按钮。
   C1TopicLink Collection Editor打开。
5. 在Members方框里，选择你想移除的链接。
6. 点击Remove移除链接。
7. 点击OK关闭C1TopicPage Collection Editor。
8. 点击OK关闭C1TopicLink Collection Editor。该主题链接从主题页移除。

使用代码

完成以下步骤：

1. 点击C1TopicBar的智能标签()，打开C1TopicBar Tasks菜单。
2. 在C1TopicBar Tasks菜单上，点击Edit Pages。



C1TopicPage Collection Editor被打开。

3. 在Members方框里，选择页面获取你想移除的链接。
4. 在Properties 网格内，将页面Tag 属性设置为“PageWithLink”。你将可以在代码中使用FindPageByTag 方法找到该页。
5. 在Properties 网格，将鼠标放置到Links 属性，点击省略号按钮，C1TopicLink Collection Editor被打开。
6. 在代码中选择你想移除的链接，在Properties 网格，设置链接的标签属性为“LinkToRemove”。你将获得一个唯一游标的链接，你可以在代码中使用FindLinkByTag 方法找到它。
7. 点击OK关闭C1TopicPage Collection Editor。
8. 点击OK关闭C1TopicLink Collection Editor。
9. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器被添加进代码视图。
10. 添加以下代码到Form_Load事件。这些代码可以找到页面和链接，指定它们为变量，然后从页面移除链接。

▸ Visual Basic

Visual Basic

```
'Find page and assign to variable
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithLink")
'Find link and assign to variable
Dim c1TopicLink1 = c1TopicPage1.FindLinkByTag("LinkToRemove")
'Remove link from page
c1TopicPage1.Links.Remove(c1TopicLink1)
```

C#

C#

```
//Find page and assign to variable
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithLink");
//Find link and assign to variable
var c1TopicLink1 = c1TopicPage1.FindLinkByTag("LinkToRemove");
//Remove link from page
c1TopicPage1.Links.Remove(c1TopicLink1);
```

11. 按F5运行工程。

自定义外观

以下主题说明了如何修改C1TopicBar 控件的外观。

添加背景图片

你可以使用BackgroundImage属性为C1TopicBar 控件添加背景图片。在这个主题中，你讲学到使用属性窗口和代码设置属性。

使用属性窗口

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标，添加C1TopicBar 控件到你的表单上。
2. 右键C1TopicBar 控件，打开它的上下文菜单，然后选择Properties。这个Properties 窗口被打开并且光标聚焦在C1TopicBar 控件的属性上。
3. 鼠标放置在BackgroundImage 属性，点击省略号按钮 打开Select Resource 弹框。
4. 选择Local resource 单选按钮并点击Import。Open 对话框打开。
5. 导航到含有你的背景图片的文件夹，选择一个图片，然后点击Open 导入这个图片。
6. Open 对话框关闭，返回给你一个Select Resource对话框。
7. 点击OK关闭Select Resource对话框。
   你的背景图片被添加到C1TopicBar 控件上。

使用代码

完成以下步骤：

1. 导航到Toolbox 并双击C1TopicBar 图标，添加C1TopicBar 控件到你的表单上。
2. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器已经被加载到了代码视图上。

3. 通过添加以下代码设置背景图片（通过你自己的路径和图片名字替换C:\YourImage.jpg）到Form_Load 事件。

Visual Basic

```
Visual Basic
c1TopicBar1.BackgroundImage = System.Drawing.Image.FromFile("C:\YourImage.jpg")
```

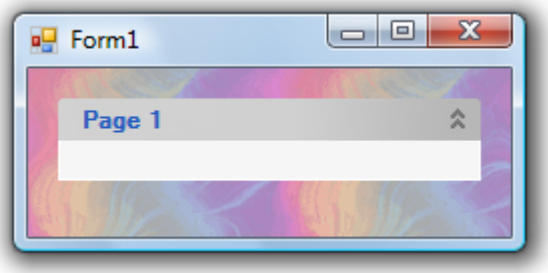
C#

```
C#
c1TopicBar1.BackgroundImage =
System.Drawing.Image.FromFile(@"C:\YourImage.jpg");
```

4. 按F5运行工程。

此主题说明如下：

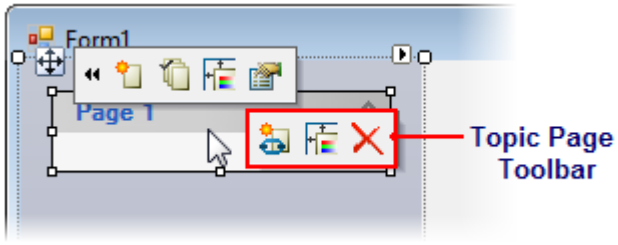
在此主题中，你学到了如何通过Properties 窗口和代码添加背景图片。下面的图片描述了C1TopicBar控件和的自定义背景图片。



添加图标到主题页

在这个主题中，你可以创建一个图片列表，然后列表中的一个图片添加到主题页上，完成以下步骤：

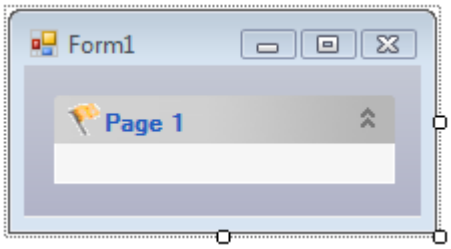
1. 在Toolbox中，双击C1TopicBar 图标，添加C1TopicBar 控件到你的工程。
2. 在Toolbox中，双击Image List图标，添加Image List组件到你的工程。
3. 点击Image List控件的智能标签（），打开Image List Tasks菜单。
4. 选择Choose Images 打开Images Collection Editor
5. 点击Add，打开Open对话框，完成以下步骤：
    1. 导航到你想使用作为图标的图片。
    2. 选择图片。
    3. 按下OK关闭Open对话框并返回Images Collection Editor。可以看到数字为"0"的图片。
6. 按下OK关闭 Images Collection Editor。
7. 右键C1TopicBar 控件打开它的上下文菜单，然后从列表中选择Properties。这个Properties窗口打开并且C1TopicBar 属性获取焦点。
8. 鼠标放置ImageList 属性，点击下拉箭头并从列表中选择imageList1。这样会加载图片列表到控件上，以便控件的主题页可以从列表中拉取图片。
9. 鼠标在Page 1上滑动知道浮动工具条出现。这个主题页的工具条出现如下：



10. 点击Edit topic page appearance按钮打开C1TopicPage Properties编辑器。
11. 点击Image Index下拉箭头并从列表中选择0。
    图标被添加到了Page 1。

主题说明如下：

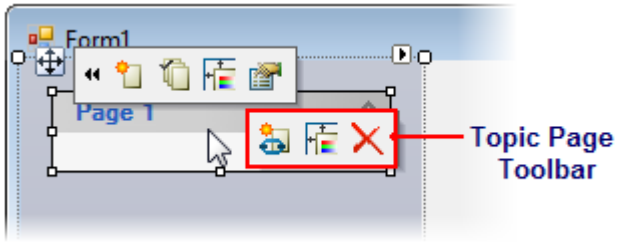
在这个主题中，你添加了图标到主题页面。下面的图片描述了页面和图标。



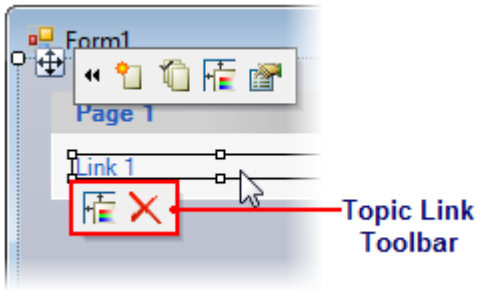
添加图标到主题链接

在这个主题中，你可以创建一个图片列表，然后列表中的一个图片添加到主题页上，完成以下步骤：

1. 在Toolbox中，双击C1TopicBar 图标，添加C1TopicBar 控件到你的工程。
2. 在Toolbox中，双击Image List图标，添加Image List组件到你的工程。
3. 点击Image List控件的智能标签（），打开Image List Tasks菜单。
4. 选择Choose Images 打开Images Collection Editor。
5. 点击Add，打开Open对话框，完成以下步骤：
    1. 导航到你想作为图标的图片。
    2. 选择图片。
    3. 按下OK关闭Open对话框并返回Images Collection Editor。可以看到数字为“0”的图片。
6. 按下OK关闭 Images Collection Editor。
7. 鼠标在Page 1上滑动直到浮动工具条出现。这个主题页的工具条出现如下：



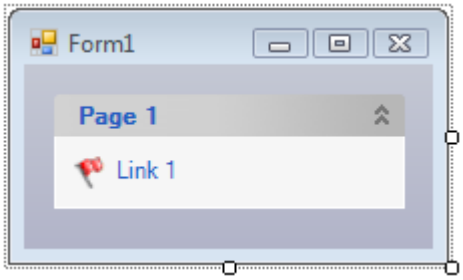
8. 在主题页工具条，点击Edit topic page appearance按钮 ，打开C1TopicPage Properties编辑器。
9. 点击Image List 下拉箭头并从列表中选择imageList1。这样将加载图片列表以便主题页的链接可以从列表中获取图片。
10. 鼠标在Link 1上滑动直到浮动工具条出现。主题链接的工具条出现相似如下：



11. 点击Edit topic link appearance按钮 打开C1TopicLink Properties编辑器。
12. 点击Image Index下拉箭头并从列表中选择0。
    图标被添加到Link 1。

主题说明如下：

在这个主题中，你添加了图标到链接页面。下面的图片通过一个图标描述了一个主题链接。



改变视觉样式

C1TopicBar 控件支持各种视觉样式，可以通过设置VisualStyle 属性添加到控件上。这个主题说明了使用智能标签，属性窗口和代码如何改变视觉样式。

使用智能标签

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标，添加控件到你的页面。
2. 点击C1TopicBar 智能标签( ) 打开C1TopicBar Tasks菜单。
3. 点击VisualStyle下拉箭头并从列表中选择一个样式。例如，选择Office2003Olive。
   选中的视觉样式被应用到控件上。

使用属性窗口

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标添加控件到你的页面。
2. 右键C1TopicBar 控件，打开上下文菜单然后选择Properties。这个属性窗口打开，并且C1TopicBar 控件的属性获得焦点。
3. 鼠标放置VisualStyle 属性，点击它的下拉箭头，并选择一个视觉样式。例如，选择Office2003Olive。
选中的视觉样式被应用到控件上。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击 C1TopicBar 图标。
2. 双击表单中的空白部分打开代码视图。注意Form_Load事件处理器已经被添加到代码视图。
3. 引入以下命名空间到工程：

Visual Basic

```
Imports C1.Win.C1Command
```

C#

```
using C1.Win.C1Command;
```

4. 设置视觉样式，添加以下代码，设置视觉样式为Office2003Olive到Form_Load事件：

Visual Basic

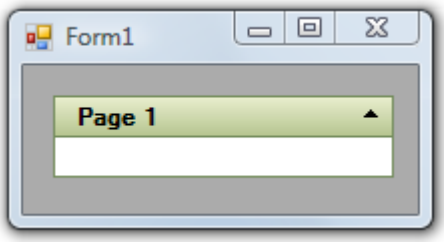
```
c1TopicBar1.VisualStyle = VisualStyle.Office2003Olive
```

C#

```
c1TopicBar1.VisualStyle = VisualStyle.Office2003Olive;
```

此主题说明如下：

在此主题中，你学到了如何使用三种不同的方法改变视觉样式。这个主题的结果如下所示：



自定义展开折叠行为

这部分的主题说明了如何改变C1TopicBar页面的展开折叠行为。

改变展开/折叠动画

C1TopicBar 控件的展开/折叠动画行为被默认设置到System上，意味着这个行为遵循用户的操作系统设置。如果你宁愿有控件在它上面，你可以通过设置Animation属性的动画行为打开动画，或者你可以设置Animation关闭动画。

使用属性窗口

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。可以观察到命名为Page 1的页面默认出现在控件上。如果你想添加更多的页面到表单上，参见Adding Topic Pages to the TopicBar。
2. 右键C1TopicBar 控件，打开上下文菜单然后选择Properties。
3. 属性窗口打开，C1TopicBar 控件的属性获得焦点。
4. 鼠标放置在Animation 属性，点击它的下拉箭头并从列表中选择一个设置。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在了控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图。
3. 引入以下命名空间到工程中：

   Visual Basic

   ```
   Imports C1.Win.C1Command
   ```

   C#

   ```
   using C1.Win.C1Command;
   ```

4. 完成以下其中一项：
   - 打开动画，添加以下代码到Form_Load事件

     Visual Basic

     ```
     c1TopicBar1.Animation = C1AnimationEnum.On
     ```

     C#

```
C#
c1TopicBar1.Animation = C1AnimationEnum.On
```

- 关闭动画，添加以下代码到Form_Load事件：

  Visual Basic

```
Visual Basic
c1TopicBar1.Animation = C1AnimationEnum.Off
```

  C#

```
C#
c1TopicBar1.Animation = C1AnimationEnum.Off;
```

5. 按下F5运行。

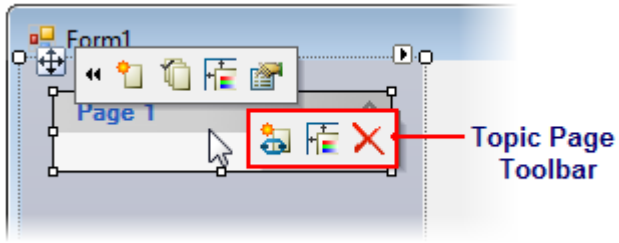
创建折叠主题页

一个被添加到C1TopicBar 控件的页面将被展开。你可以通过设置Collapsed 属性为True来创建折叠页。在此主题中，你将会学到如何使用浮动工具条，集合编辑器和代码设置Collapsed 属性。

使用浮动工具条

完成以下步骤

1. 导航到Toolbox并双击Collapsed 图标。Collapsed 控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 鼠标在Page1（或者你选择的另一个页面）上滑动直到浮动工具条出现。这个主题页的工具条出现如下：

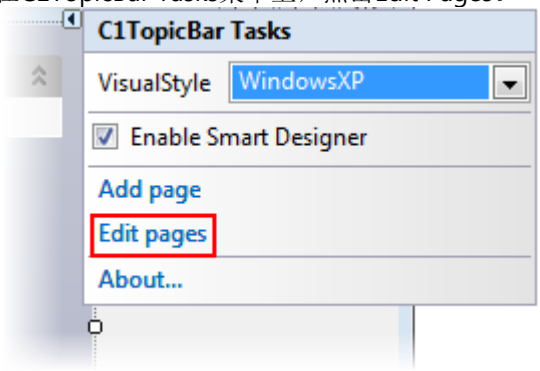


3. 点击Edit topic page appearance按钮 打开C1TopicPage Properties编辑器。
4. 选择Collapsed 选择框并看到这个页面被折叠。

使用集合编辑器

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加进了表单。可以观察到命名为Page 1的页面默认出现在控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar智能标签(▶) 打开C1TopicBar Tasks菜单。
3. 在C1TopicBar Tasks菜单上，点击Edit Pages。



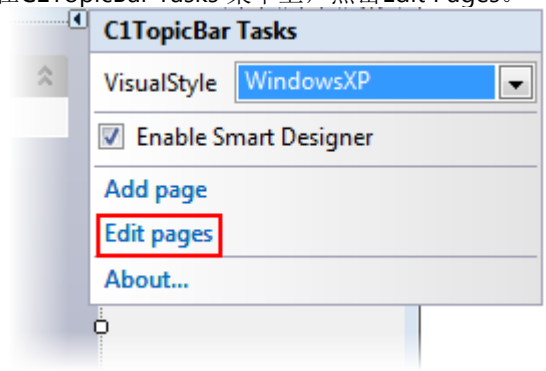
C1TopicPage Collection Editor打开。

4. 在Members方框，选择你想折叠的页面。
5. 在Properties 网格，设置Collapsed 属性为True。
6. 按下OK关闭C1TopicPage Collection Editor并观察到一个主题页被折叠。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。看到命名为Page 1的页面被默认出现在了控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar智能标签打开C1TopicBar Tasks 菜单。
3. 在C1TopicBar Tasks 菜单上，点击Edit Pages。



C1TopicPage Collection Editor打开。

4. 在Members 方框里，选择你想折叠的页面。
5. 在Properties 网格里，设置Tag属性为"PageToCollapse"。此标签是一个唯一的指示器，允许你通过FindPageByTag 方法在代码中找到合适的页面。
6. 点击OK关闭C1TopicPage Collection Editor。
7. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图。
8. 添加以下代码到Form_Load事件。此代码将发现一个页面，指定它为一个变量，然后设置页面的Collapsed 属性为True。

▶ Visual Basic

```
Visual Basic
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageToCollapse")
```

```
c1TopicPage1.Collapsed = True
```

C#

```
C#
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageToCollapse");
c1TopicPage1.Collapsed = true;
```

9. 按F5运行工程。

使用主题条工具提示

这部分主题教你如何使用和操作C1TopicBar 工具提示。你将学到如何为主题页和主题链接创建工具提示，你也可以学到如何禁用控件上所有的工具提示。

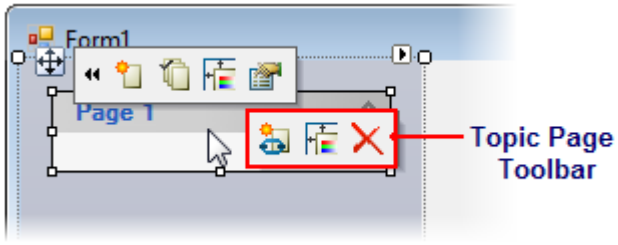
添加工具提示到主题页

你可以使用浮动工具条，集合编辑器和代码添加工具提示到主题页面。

使用浮动工具条

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标，C1TopicBar 控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 鼠标在Page 1（或者你选择的另一个页面）上滑动直到浮动工具条出现。主题页面的工具条出现如下：

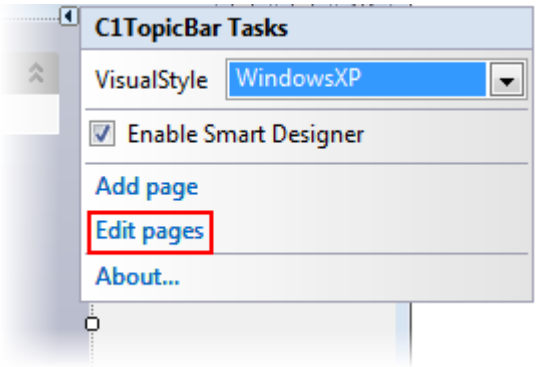


3. 点击编辑主题页外观按钮  ，打开C1TopicPage Properties编辑器。
4. 在Tooltip 文本框内，输入“I am a topic page ToolTip!”。
5. 按F5运行工程。
   鼠标悬浮在你添加的工具提示页面上，可以看到工具提示出现。

使用集合编辑器

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。可以看到命名为Page 1的页面被默认添加到了控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar智能标签(▶) 打开C1TopicBar Tasks菜单。
3. 在C1TopicBar Tasks菜单上，点击Edit Pages。
4. 在Members方框里，选择你想添加工具提示的页面。

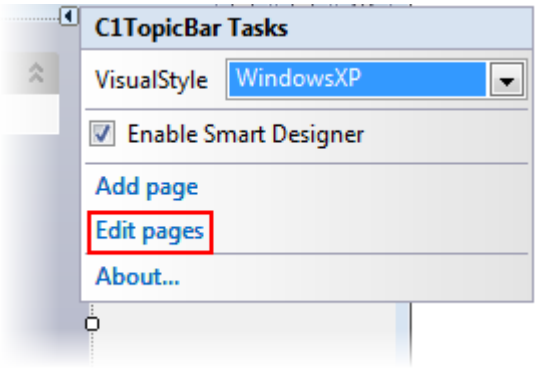


5. 在Properties 网格里，设置ToolTipText 属性为"I am a topic page ToolTip!"。
6. 点击OK关闭C1TopicPage Collection Editor。
7. 按F5运行工程。
8. 鼠标悬浮在你添加了工具提示的页面，看到一个工具提示出现。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。可以看到命名为Page 1的页面默认出现在控件上。如果你想添加更多的页面到控件上，参见Adding Topic Pages to the TopicBar。
2. 点击C1TopicBar 的智能标签(▶)打开C1TopicBar Tasks 菜单。
3. 在C1TopicBar Tasks 菜单上，点击Edit Pages。



C1TopicPage Collection Editor 打开。

4. 在Members方框内，选择你想动态添加工具提示的页面。
5. 在Properties 网格里，设置Tag 属性为"PageWithToolTip"。在之后的步骤中，你将使用这个标签找到主题页面。
6. 点击OK关闭C1TopicPage Collection Editor。
7. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图。
8. 引入下面的命名空间到工程中：

▶ Visual Basic

```
Visual Basic
Imports C1.Win.C1Command
```

▶ C#

C#

```
using C1.Win.C1Command;
```

9. 添加下面的代码到Form_Load事件。此代码将可以找到这个页面，指定它为一个变量，然后设置页面的ToolTipText属性。

Visual Basic

```
Dim c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithToolTip")
c1TopicPage1.ToolTipText = "I am a topic page ToolTip!"
```

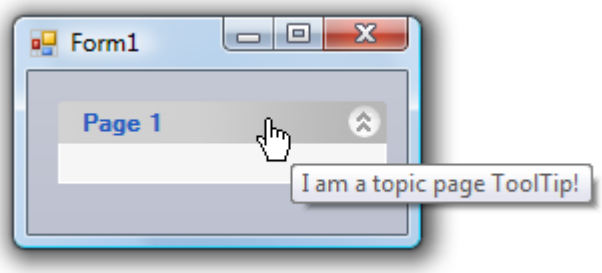
C#

```
var c1TopicPage1 = c1TopicBar1.FindPageByTag("PageWithToolTip");
c1TopicPage1.ToolTipText = "I am a topic page ToolTip!";
```

10. 按F5运行工程。
11. 滑动鼠标在你添加了工具提示的页面上，并看到一个工具提示出现。

主题说明如下：

在此主题中，你学到了如何使用浮动工具条，集合编辑器和代码添加一个工具提示到主题页面上。无论你再此主题中使用哪一种方法，结果将相似如下：



添加ToolTips到链接页面

你可以使用浮动工具条，集合编辑器，代码添加工具提示到主题链接。

使用浮动工具条

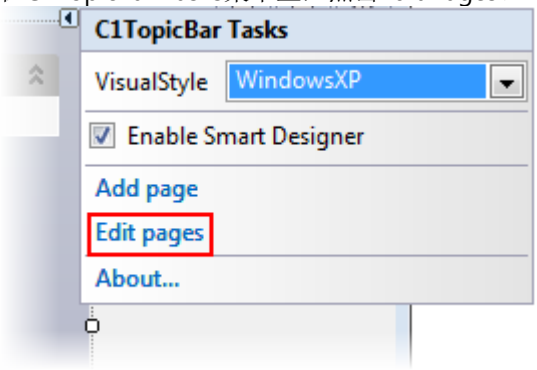
完成以下步骤：

1. 导航到Toolbox双击C1TopicBar 图标。C1TopicBar 图标被添加到了表单上。观察到一个命名为Page 1的页面默认出现在控件上。
2. 添加主题链接到Page 1（参见 Adding Topic Links to Topic Pages）。
3. 鼠标悬浮到新的链接直到它的浮动工具条出现，然后点击Edit topic link appearance按钮  。
4. 在工具提示文本框中，键入“I am a topic link ToolTip!”。
5. 按F5运行工程。
6. 鼠标悬浮到你添加了工具提示的链接上，观察到工具提示出现。

使用集合编辑器

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。观察到命名为Page 1的页面默认出现在控件上。
2. 点击C1TopicBar 的智能标签(▶) 打开 C1TopicBar Tasks菜单。
3. 在C1TopicBar Tasks菜单上，点击Edit Pages。



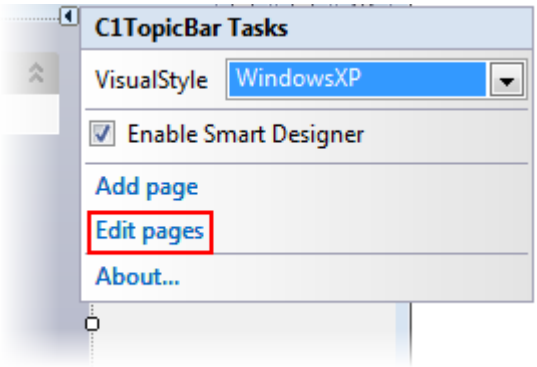
C1TopicPage Collection Editor打开。

4. 在Properties 网格，鼠标放置Links属性并点击省略号按钮[...]。C1TopicLink Collection Editor 打开。
5. 在Members方框，选择你想添加ToolTip的链接。
6. 在Properties 网格，设置ToolTipText 属性为"I am a topic link ToolTip!"；
7. 点击OK关闭C1TopicLink Collection Editor。
8. 点击OK关闭C1TopicPage Collection Editor。
9. 按F5运行工程。
10. 鼠标悬浮在你添加了工具提示的链接上，观察到工具提示出现。

使用代码

完成以下步骤：

1. 导航到Toolbox并双击C1TopicBar 图标。C1TopicBar 控件被添加到了表单上。观察到命名为Page 1的页面默认出现在控件上。
2. 点击C1TopicBar 的智能标签(▶) 打开C1TopicBar Tasks 菜单。
3. 在C1TopicBar Tasks 菜单上，点击Edit Pages。



C1TopicPage Collection Editor打开。

4. 在Members方框里，选择一个页面。

5. 在Properties 网格中，放置鼠标在Links属性并点击省略号按钮 ，C1TopicLink Collection Editor 打开。
6. 在Members方框里，选择你想添加工具提示的链接。
7. 在Properties 网格，设置Tag 属性为"LinkToolTip"。在之后的步骤，你将使用这个标签查找主题链接。
8. 点击OK关闭C1TopicLink Collection Editor。
9. 点击OK关闭C1TopicPage Collection Editor。
10. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器被添加到了代码视图。
11. 添加以下代码到Form_Load 事件。此代码将查找这个页面，指定它为一个变量，然后设置页面的ToolTipText 属性。

Visual Basic

```
Visual Basic
Dim c1TopicLink1 = c1TopicBar1.FindLinkByTag("LinkToolTip")
c1TopicLink1.ToolTipText = "I am a topic link ToolTip!"
```

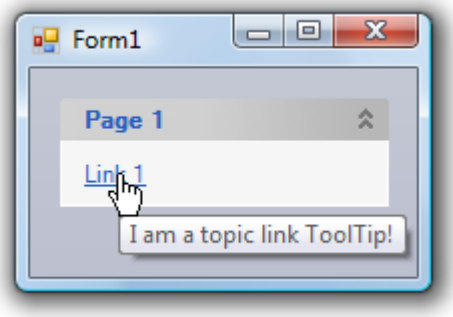
C#

```
C#
var c1TopicLink1 = c1TopicBar1.FindLinkByTag("LinkToolTip");
c1TopicLink1.ToolTipText = "I am a topic link ToolTip!";
```

12. 按F5运行工程。
13. 鼠标悬浮在你添加了工具提示的链接上，观察到工具提示出现。

主题说明如下：

在此主题中，你学到了如何使用浮动工具条，集合编辑器，代码添加工具提示到一个主题页面上。无论你在这个主题中使用哪一种方法，结果相似如下：



禁用ToolTips

你可以通过设置ShowToolTips 属性为False禁用所有Tooltips。

使用属性窗口

完成以下步骤：

1. 右键C1TopicBar 控件打开它的上下文菜单。

2. 选择Properties 打开属性窗口。
3. 鼠标放置在ShowToolTips 属性，点击它的下拉箭头并选择False。

使用代码

完成步骤：

1. 双击表单的空白部分打开代码视图。注意Form_Load事件处理器已经被添加到了代码视图。
2. 设置ShowToolTips 属性为False，添加如下代码到 Form_Load：

   Visual Basic

   ```
   Visual Basic
   C1TopicBar1.ShowToolTips = False
   ```

   C#

   ```
   C#
   c1TopicBar1.ShowToolTips = false;
   ```

3. 按F5运行工程。