ComponentOneInput for WinForms概述

ComponentOneInputforWinForms控件集是一套完整的输入控件，它具有内置的掩码，以及格式化、解析及验证功能，包含9个独立控件。这套控件允许您以可视化格式来显示动态的数据，对数据集导航进行管理，并可以创建专门的下拉列表等。	**Getting Started** • **C1Input控件** • **Input for WinForms 基于任务的帮助**

关键特性

支持数据绑定，显示动态数据，进行数据导航，显示和编辑日期数据，ComponentOneInputforWinForms 包含了众多的特性：

- 支持Office 2007和2010样式
  所有的C1Input 控件都支持多种视觉样式，包括 Office 2007，Office 2010等。使用样式化的C1Input控件以及其他的ComponentOne WinForms控件，可以为你的应用程序提供一个现代化的且一致的外观
- 支持数据绑定，显示动态数据
  控件有数据绑定和非数据绑定两种模式，在绑定模式下，控件的值可以绑定到数据源的某一个字段。
- 众多的数据源支持
  C1Input控件可以绑定到所有的 .NET 数据源，包括 ADO.NET 数据源对象，以及ComponentOne 数据对象组件。
- 强大的可定制的编辑掩码能力
  C1TextBox以及其派生类都具有强大的编辑掩码功能，包括日期和时间格式，数字以及自定义格式。
- 强大的数据格式化能力以及控件样式定制能力
  开发人员可以根据丰富的格式化数据定制控件的文本，边框，颜色样式等等。
- 支持数据验证
  使用Input for WinForms控件的前后验证属性设置，您就可以指定验证的规则，而无需对事件进行编码处理。前验证可以允许您检查原始输入的文本，而后验证可以对该值是否符合一定的标准进行验证。
- 众多的文化（区域）支持
  所有的C1Input控件都包含了文化（区域）设置，这些设置被用来进行文本比较，数字和时间格式化以及其他的特殊字符。
- 支持下拉编辑器
  C1DropDownControl允许开发人员添加自己的逻辑到增加减少按钮，下拉按钮以及下拉编辑框上面。
- 输入错误提示
  完整的数据错误处理行为，在解析和验证数据的过程中如果检测到错误，显示一条错误信息。
- 下拉按钮以及增加减少按钮
  用于编辑日期，时间的控件：C1DateEdit,C1NumericEdit,以及C1ComboBox控件都包含了下拉和增加 / 减少按钮。
- 内置了多种编辑模式
  C1Input控件支持显示模式和编辑模式，前者的时候用户不能对控件进行编辑，支持为不同的编辑模式设定显示格式。
- 能够正确区分并且处理NULL 值和空值
  C1Input控件提供了灵活的机制来处理NULL 值和空值，程序员可以很好的解决这个问题。

设计时支持

C1Input控件提供了可视化的编辑器来帮助开发人员配置控件，下面的单元描述了如何利用C1Input控件的设计时特性，来设置控件。

上下文菜单

在设计时开发人员可以使用C1Input 控件的上下文菜单来配置控件。

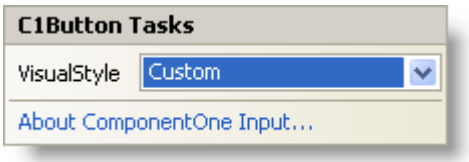
任务菜单

在 VisualStudio 2005，2008以及2010中C1Input控件都提供了智能标签，通过智能标签每一个 C1Input 控件都提供了最常用的属性设置。开发人员可以点击控件右上方的智能标签图标（▶）来打开任务菜单。更多关于每一个C1Input控件的任务菜单的操作指南，参见下面的主题：

C1Button任务菜单

在C1Button控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1Button控件的右上角点击智能标记(▶)图标，会打开控件的任务菜单。



C1Button任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。Thedefault valueisCustom.

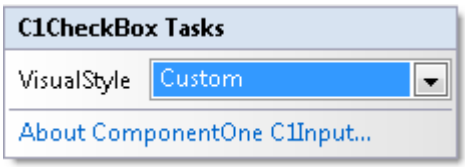
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1CheckBox任务菜单

在C1CheckBox控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1CheckBox控件的右上角点击智能标记)图标，会打开控件的任务菜单。



C1CheckBox 任务菜单提供了如下操作：

- 视觉样式

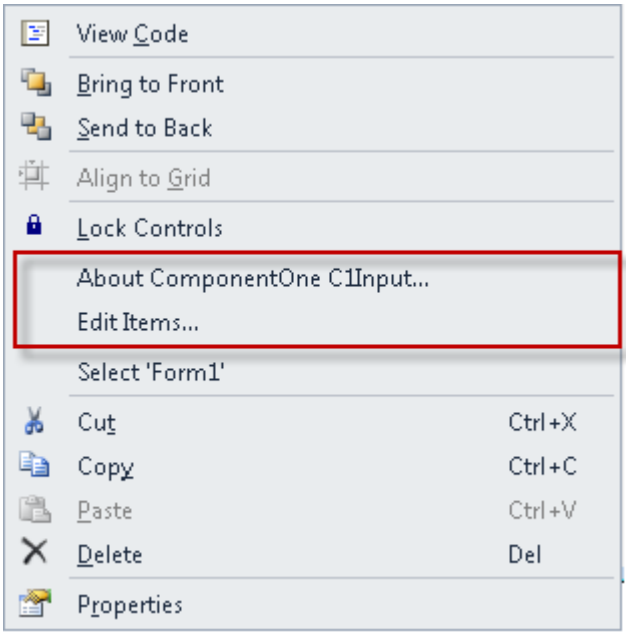
  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1ComboBox上下文菜单

C1ComboBox控件在设计时为开发人员提供了上下文菜单，可以便捷的来设置控件属性，右键点击控件，将会打开C1ComboBox的上下文菜单。



C1Input上下文菜单提供了如下操作：

关于ComponentOne Input
点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

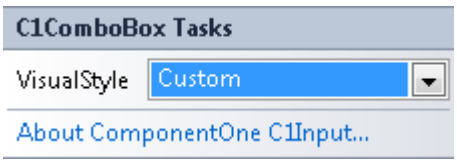
编辑子条目...

点击"编辑子条目..."菜单条目，将会打开字符串集合编辑框，每一行都是一个子条目，编辑完成后，点击保存。

C1ComboBox任务菜单

在C1ComboBox控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1ComboBox控件的右上角点击智能标记（）图标，会打开控件的任务菜单。



C1ComboBox任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

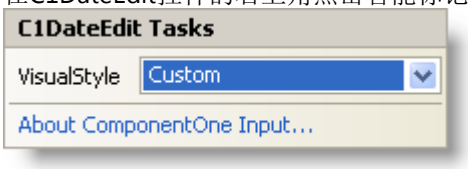
- 关于ComponentOne Input

  点击“关于ComponentOne Input”项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1DateEdit任务菜单

在C1DateEdit控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1DateEdit控件的右上角点击智能标记()图标，会打开控件的任务菜单。



C1DataEdit任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

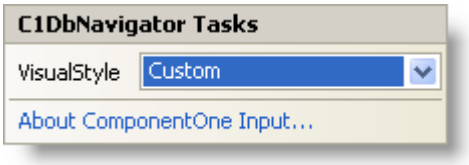
- 关于ComponentOne Input

  点击“关于ComponentOne Input”项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1DbNavigator任务菜单

在C1DbNavigator控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1DbNavigator控件的右上角点击智能标记()图标，会打开控件的任务菜单。



C1DbNavigator任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

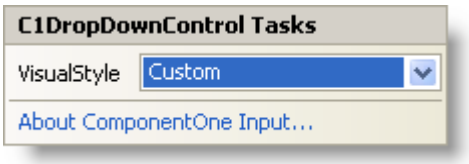
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1DropDownControl任务菜单

在C1DropDownControl控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1DropDownControl控件的右上角点击智能标记 (▶)图标，会打开控件的任务菜单。



C1DropDownControl任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

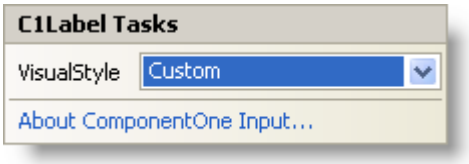
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1Label任务菜单

在C1Label控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1Label控件的右上角点击智能标记)图标，会打开控件的任务菜单。



C1Label任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

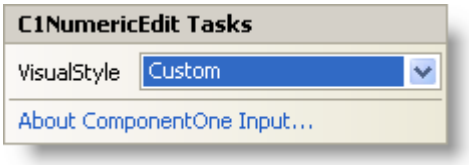
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1NumericEdit任务菜单

在C1NumericEdit控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1NumericEdit控件的右上角点击智能标记)图标，会打开控件的任务菜单。



C1NumericEdit任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

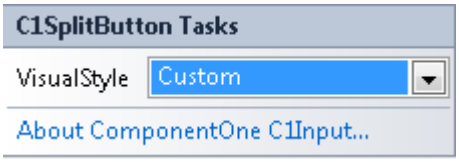
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

C1SplitButton任务菜单

在C1SplitButton控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1SplitButton控件的右上角点击智能标记)图标，会打开控件的任务菜单。



C1SplitButton任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007 银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

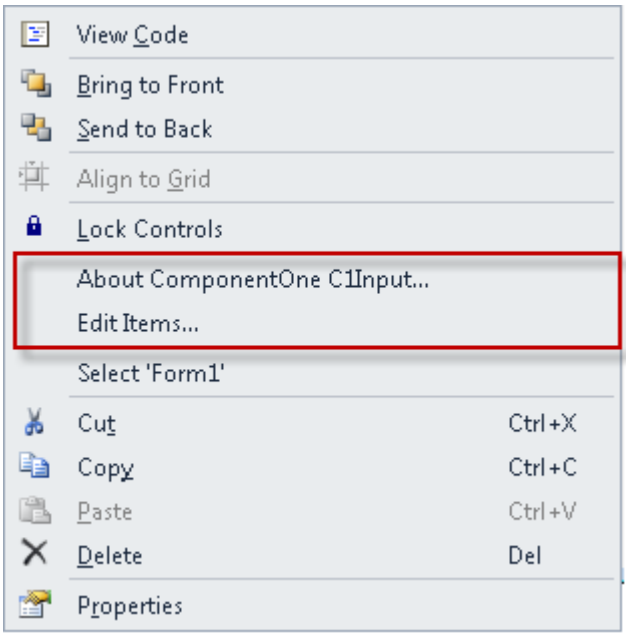
- 关于ComponentOne Input

  点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前

版本，以及一些在线资源链接。

C1SplitButton上下文菜单

C1SplitButton控件在设计时为开发人员提供了上下文菜单，可以便捷的来设置控件属性，右键点击控件，将会打开C1SplitButton的上下文菜单。



C1Input上下文菜单提供了如下操作：

关于ComponentOne Input
点击"关于ComponentOne Input"项目将会显示关于ComponentOne Input 的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

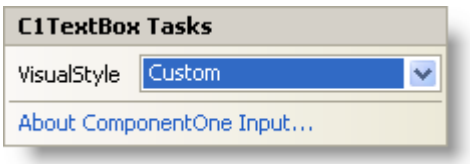
编辑子条目...

点击"编辑子条目..."菜单条目，将会打开字符串集合编辑框，每一行都是一个子条目，编辑完成后，点击保存。

C1TextBox任务菜单

在C1TextBox控件的任务菜单上可以快捷、容易的设置控件的VisualStyle属性。

在C1TextBox控件的右上角点击智能标记 (▶)图标，会打开控件的任务菜单。



C1TextBox 任务菜单提供了如下操作：

- 视觉样式

  点击视觉样式下拉框，会有一系列的样式选项，包括：系统，Office2007 蓝色，Office2007 黑色，Office2007

银色，Office2010 蓝色，Office2010 黑色，Office2010 银色以及自定义样式。默认值为自定义样式。

- 关于ComponentOne Input

  点击“关于ComponentOne Input”项目将会显示关于ComponentOne Input的对话框，程序员可以获取控件当前版本，以及一些在线资源链接。

使用C1Input控件

下面的章节描述C1Input控件的各种用法。

C1Input控件

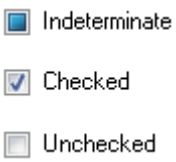
此版本的C1Input包含十个控件：

- **C1Button**
  一个类似标准按钮控件，从System.Windows.Forms.Button类继承。C1Button控件提供了更多的视觉效果。
- **C1CheckBox**
  一个窗口控件，用于设置或改变项目值为真，假，或不确定。C1CheckBox控件默认为允许两中复选状态的复选框，但可以设置ThreeState属性值为有效来切换为允许三中复选状态的复选框。

  下表描述了复选框的三种状态及其影响的视觉效果。

复选框状态	描述
不确定	当只有部分的复选框项目被选中时，框内显示一个带有深色阴影的灰色框。
选中	当项目被选中时，复选框中显示一个勾形符号。
未选中	当项目未被选中时，显示一个空的复选框。

  下图演示了**C1CheckBox**控件的每一种复选框状态：不确定状态，选中状态，和未选中状态。



- **C1ComboBox**
  一个组合框控件，允许用户在下拉列表中浏览选项并且可选择一个或多个选项。
- **C1TextBox**

  主要的数据绑定控件，用于以文本格式输入和编辑任何类型的数据信息。支持格式化数据，编辑掩码，数据验证等其它功能。C1TextBox也支持以格式化和掩码的形式编辑任何类型的数据，包括特殊的日期和时间格式。除了是主要的数据编辑控件外，C1TextBox同时也是C1NumericEdit和C1DateEdit等专门控件的基类。C1TextBox 从标准的System.Windows.Forms.TextBox控件派生。

- **C1DropDownControl**
  一个派生自C1TextBox的控件。C1DropDownControl支持C1TextBox所有的格式化、验证等其他功能。和其他两个从C1TextBox派生的控件一样，它也支持UpDown（微调）和下拉按钮。然而，和那两个专门控件不同，C1DropDown控件允许为UpDown按钮添加自定义逻辑，也能为下拉按钮添加自定义下拉窗体或编辑器。
- **C1DateEdit**
  一个数据绑定控件，派生自C1TextBox，专门用来编辑时间和日期类型的值。除了具有C1TextBox的功能外，C1DateEdit包含一个下拉日历和用于更改数值的向上和向下（微调）按钮。
- **C1DbNavigator**
  一个数据绑定控件，提供一系列方便导航数据源记录行的按钮。它允许移动到开始、最后、前一和后一行，以及更新数据源和刷新数据等常用数据操作行为。
- **C1Label**
  一个只读的数据绑定控件，显示格式化的数据。C1Label派生自标准的System.Windows.Forms.Label控件。

- **C1NumericEdit**
  一个数据绑定控件，派生自C1TextBox，专门用来编辑数字值。除了具有C1Text的功能外，C1NumericEdit包含一个下拉计算器以及用于递增和递减数值的向上和向下（微调）按钮。
- **C1PictureBox**
  一个数据绑定控件，显示存储于数据源字段中的图片，派生自System.Windows.Forms.PictureBox。

C1DbNavigator控件概述

C1DbNavigator类表示C1DbNavigator控件。一个提供一系列用于便捷操作数据源记录行的按钮的数据绑定控件。允许移动到第一、最后、前一和后一行，以及更新数据源与刷新数据源等常用数据操作行为。

NavigatorButtonEnum枚举提供了C1DbNavigator控件可用按钮的列表，包括：Add、Apply、Cancel、Delete、Edit、First、Last、Next、Position、Previous、Refresh和Update。

NavigatorButtonEnum中的Position枚举值可在Position文本框中的文本变化之后触发的BeforeAction事件中引用。请参照 在导航器中更改导航 查看示例。

C1DbNavigator控件包含下列可用于导航和编辑数据集中记录的按钮:

按钮	描述
	第一行按钮。 移动到记录集的第一行。默认可见。
	前一行按钮。移动到记录集的前一行。默认可见。
	下一行按钮。移动到记录集的下一行。默认可见。
	最后一行按钮。移动到记录集的最后一行。默认可见。
	添加按钮。在记录集中添加一行。默认不可见。
	删除按钮。删除记录集中的行。默认不可见。
	编辑按钮。编辑记录集中的行。默认不可见。
	应用按钮。应用在记录中做的更改。默认不可见。
	取消按钮。取消在记录中的更改。默认不可见。
	更新按钮。更新在记录中的修改。默认不可见。
	刷新按钮。刷新记录集。默认不可见。

C1DbNavigator外观

使用C1DbNavigator的各属性，可以方便的自定义C1DbNavigator的按钮，边框和用户界面文本。

C1DbNavigator的按钮属性

下表列举及描述了用于自定义C1DbNavigator 控件中的按钮的属性：

属性	描述

C1DbNavigator.ButtonSize	导航按钮的大小。
C1DbNavigator.ButtonStyle	读取导航按钮的样式。可选值为Flat或Standard。
C1DbNavigator.ButtonTextAlign	控制导航按钮中文本和图片的相对位置。
C1DbNavigator.ButtonTexts	读取或设置按钮上显示的文本。
C1DbNavigator.ButtonToolTips	定义导航按钮上的提示信息的集合。
C1DbNavigator.ColorButtons	指定导航按钮是否有彩色图片。
C1DbNavigator.ColorWhenHover	如果为真，导航按钮在鼠标移动到其上时显示彩色图片。
C1DbNavigator.VisibleButtons	指定哪些按钮可见。

C1DbNavigator的用户界面文本

下表列举及描述了用于自定义C1DbNavigator控件的用户界面文本的属性：

UIString	描述
Row:	表示记录集中当前选中的行号。
of {0}	表示记录集中总共的行数。
(inactive)	表示位置文本框中显示的文本。
(empty)	
Confirmation	点击删除按钮以删除记录中的一行时弹出的对话框的标题。
Do you want to delete the row?	在确认对话框中显示的内容。

C1DbNavigator 位置文本框属性

可以使用Text属性来读取或设置位置文本框中的文本。如果Position文本框不可见，返回值为空字符串。

在Position文本框不可见的情况下设置Text属性将被不起作用。更改Text属性引发数据源位置的更改。

C1DbNavigator行为

C1DbNavigator包含多个事件来控制其行为。例如，当点击一个按钮时，记录集的当前行或者字段发生改变时，点击一个按钮后发生错误时，或者关于视觉样式的属性值改变时，都可以改变C1DbNavigator的行为

下表列举了C1DbNavigator的事件：

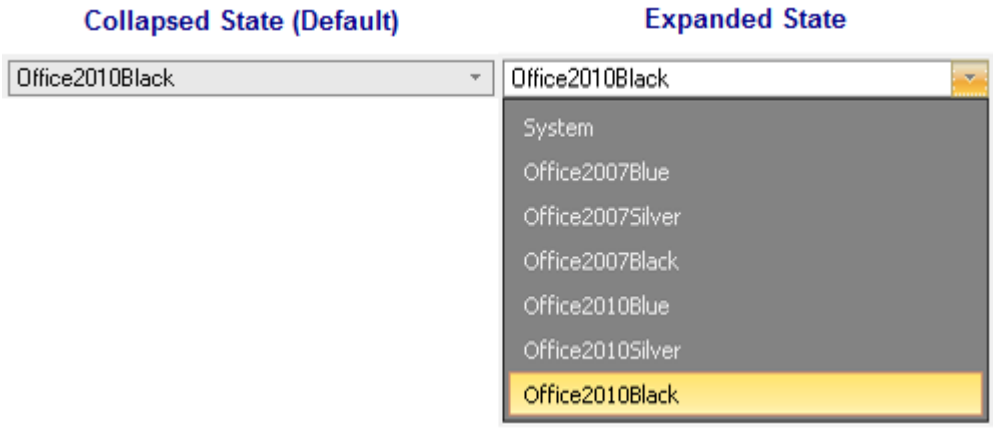
事件	描述
Adding	点击Add按钮时触发。
BeforeAction	点击一个按钮后且将要执行动作前触发。
ButtonClick	点击一个导航按钮且执行动作之后触发。
ButtonCursorChanged	ButtonCursor属性值被改变时触发。
Deleting	点击Delete按钮时触发。

Editing	点击Edit按钮时触发。
Error	点击按钮且执行动作时发生异常时触发。
ItemChanged	当前行发生改变时触发，一些字段被改变了。
PositionChanged	位置发生改变时触发。
RefreshData	点击Refresh按钮时触发。
TextChanged	当C1.Win.C1Input.C1DbNavigator.Text属性值被改变时触发。
UpdateData	点击Update按钮时触发。
VisualStyleChanged	当VisualStyle属性值被改变时触发。

C1ComboBox 控件概述

C1ComboBox是一个组合控件，用于显示可被选择项目的列表。 它的功能类似于ListBox控件，由于项目是可被隐藏的它占用很少的空间。可以通过Items属性添加项目到C1ComboBox控件中，也可以绑定数据到一个字符串数组或者数据源。C1ComboBox包含下列元素： Textbox，Button和 DropDownList。可以在文本框中输入任何数据，或者可以点击下拉按钮后在下拉列表中选择一个项目。参考 C1ComboBox元素 查看更多信息。C1ComboBox控件在默认状态下是折叠的并且只在文本框内显示一个项目。在展开状态下C1ComboBox控件在下拉列表框中显示可选择的项目。

下图演示了C1ComboBox的折叠和展开状态：

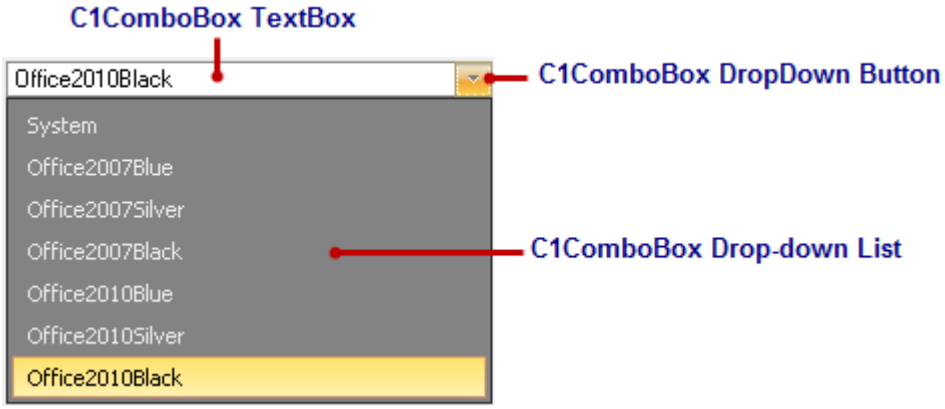


在典型的组合框控件中，显示在右侧的下拉按钮起到显示下拉列表以快速从列表中选择一个选择项的功能。然而，可以在C1ComboBox中添加更多的功能。比如创建一个向上、向下微调按钮来编辑数字值，或者添加一个模态按钮用来在组合框中显示一个模态对话框。

C1ComboBox元素

此章节从视觉和描述性的角度描述组成C1ComboBox控件的元素。

C1ComboBox控件由可编辑的文本框和下拉列表组成。



C1ComboBox文本框

运行时用户可以在C1ComboBox的文本框中输入文本，或者为Text属性指定字符串。如果为Text属性指定了一个值，将会改变C1ComboBox中文本框的值。

在C1ComboBox文本框中输入文本的时候，下拉列表会显示C1ComboBox中相匹配的项目。例如，如果在列举客户名字的C1ComboBox的文本框中输入“Sm”，那么所有以“Sm”开头的客户名字将出现在下拉列表中。

C1ComboBox下拉按钮

C1ComboBox默认显示一个下拉样式的按钮，但可以通过设置VisibleButtons属性来选择是否要显示这个按钮。除了下拉按钮，也可以显示UpDown按钮或者自定义按钮。模态按钮可以显示在下拉按钮、向上向下按钮、或自定义按钮后，或者也可以 单独的显示。参见 C1ComboBox按钮外观 查看更多信息。

可以通过ButtonWidth属性更改按钮的大小。

C1ComboBox项目的下拉列表

下拉列表由C1ComboBox的项目组成，并且只在运行时可见。可以通过C1ComboBox文本框后的触发按钮或下拉箭头来访问下拉列表。

设计时可以在String Collection Editor中添加C1ComboBox的项目。在字符串集合编辑器中一行一行的输入项目。在运行时可以通过Items属性动态的添加项目。

ComboBox 项目模式

可以使用ItemMode属性构建项目的外观。有如下三个选项：

Default

Default模式下每一个C1ComboBox项目显示为一个带图片的字符串。下图演示了C1ComboBox在Default模式下的样式：

具体例子参见ComboBoxImages示例。

HtmlPattern

HtmlPattern模式下每个项目由HTML模板和绑定的数据构成。

下图演示了C1ComboBox在HtmlPattern模式下的样式：

上图使用的HTML模板如下：
<table><tr><td>Country:</td><td>{Text}</td></tr><tr><td align="right">Flag:</td><td><img src='{Text}'></td></tr></table>

具体例子参见ComboBoxItemModes示例。

Html

Html模式下每一个项目都是一个HTML子集的片段。

下图演示了C1ComboBox在Html模式下的样式：

上图使用的Html设置如下：

具体例子参见ComboBoxItemModes 示例。

C1ComboBox 样式

如果不想用预定义的主题，可以设置C1ComboBox的VisualStyle属性值为Custom来为C1ComboBox控件创建自定义的样式。
C1ComboBox包含下列控制外观的属性：

属性	描述
DefaultItemForeColor	读取和设置C1ComboBox中项目的默认前景色
DropDownBackColor	读取和设置C1ComboBox下拉窗体的默认背景色。
DropDownBorderColor	读取和设置C1ComboBox下拉窗体的默认边框颜色。
HotItemBackColor	读取和设置鼠标移动到组合框项目上时的背景色。
HotItemBorderColor	读取和设置鼠标移动到组合框项目上时的背景色。
HotItemForeColor	读取和设置鼠标移动到组合框项目上时的前景色。
Padding	读取和设置下拉窗体内的补白。
TextSpacing	读取和设置组合框项目文本部分之间的空白。

C1ComboBox按钮外观

C1ComboBox默认在文本框右侧显示一个下拉按钮。可以使用C1ComboBox.VisibleButtons属性决定这个按钮是否可见。

C1ComboBox.VisibleButtons 属性有下列可选值：

值	外观或描述
None	C1ComboBox不显示下拉按钮。
UpDown	显示向上、向下微调按钮。
DropDown	显示下拉按钮的默认图片。
Modal	显示模态按钮的默认图片。如果下拉按钮也可见则显示在其右侧。
Custom	读取和设置C1ComboBox控件的自定义按钮。

决定了按钮样式之后就可以为被选择按钮使用默认按钮图片或者创建自定义的图片。下列属性用来为每种按钮样式应用自定义图片（Custom，UpDown，DropDown和Modal）：

- **ButtonImages.CustomImage** - 应用于自定义按钮的图片。
- **ButtonImages.DownImage** - 应用于向下按钮的图片。
- **ButtonImages.DropImage** - 应用于下拉按钮的图片。
- **ButtonImages.ModalImage** - 应用于模态按钮的图片。
- **ButtonImages.Up** - 应用于向上按钮的图片。

ComboBox数据绑定

C1ComboBox可以在运行时绑定到一个枚举或者一个数组，或者在设计时绑定到一个字符串数组或绑定源。绑定C1ComboBox到数据源可以使其浏览数据库中的数据，添加新数据，或者编辑现有数据。

C1ComboBoxFeatures 示例演示了使用如下不同的C1ComboBox数据绑定方法：

- 如何在运行时绑定C1ComboBox到一个枚举
- 如何在运行时绑定C1ComboBox到一个数组
- 如何在设计时绑定C1ComboBox到一个数组
- 如何绑定C1ComboBox 到一个绑定源

为ComboBox项目添加图片

可以方便的将ImageList中的图片添加到C1ComboBox控件的下拉列表中每个项目中。

要在设计时添加图片到C1ComboBox元素中，按如下步骤完成：

1. 在窗体上添加一个C1ComboBox控件。
2. 使用String Collection Editor添加项目到C1ComboBox.Items集合。
3. 在窗体上添加一个ImageList控件。
4. 添加图片到imageList1中。
5. 为C1ComboBox集合中项目设置相应的键（名称）。

要在运行时添加图片到C1ComboBox项目中，加入下列代码：

Visual Basic

```
Visual Basic
c1ComboBox1.ItemsImageList = imageList
imageList.Images.Add("First item", Image.FromFile("First.png"))
c1ComboBox1.Items.Add("First item")
```

C#

```
C#
c1ComboBox1.ItemsImageList = imageList;
imageList.Images.Add("First item", Image.FromFile("First.png"));
c1ComboBox1.Items.Add("First item");
```

添加C1ComboBox项目

可以使用Add方法以编码的方式方便的 添加项目到C1ComboBox中，或者在设计时通过String Collection Editor添加项目。如果已有多个项目，Add方法添加新项目到下一个位置。如果需要将添加项目在列表的特定位置，可以使用Insert方法。可以使用AddRange方法将整组对象或字符串添批量加到C1ComboBox的项目列表中。

以编码的方式添加项目

使用C1ComboBox类的Add方法将项目添加到C1ComboBox中。集合通过Items属性引用。

Visual Basic

```
Visual Basic
c1ComboBox1.Items.Add("Pittsburgh")
```

C#

```
C#
c1ComboBox1.Items.Add("Pittsburgh");
```

通过String Collection Editor添加项目

1. 在窗体上右键点击C1ComboBox控件然后选择Edit Items。出现String Collection Editor编辑器。
2. 在String Colllection Editor编辑器中，输入字符串然后按Enter来添加下一个字符串。

在特定位置插入字符串或对象

如下例子在第五个位置插入字符串Chicago：

Visual Basic

```
Visual Basic
c1ComboBox1.Items.Insert(4, "Chicago")
```

C#

```
C#
c1ComboBox1.Items.Insert(4, "Chicago");
```

传递一个字符串数组

要传递一个字符串数组，完成以下步骤：

1. 在窗体上添加一个C1ComboBox 控件。
2. 在窗体上添加一个Button控件。
3. 创建如下Button_Click事件处理程序并添加如下代码来传递字符串数组到C1ComboBox。

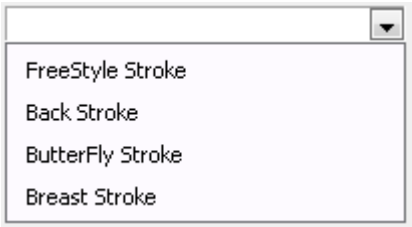
   Visual Basic

   ```
   Visual Basic
   Type your Drop Down Section text here.
   ```

   C#

   ```
   C#
   private void button1_Click(object sender, EventArgs e)
       {
           string[] items = { "FreeStyle Stroke", "Back Stroke", "ButterFly Stroke", "Breast Stroke"};
           c1ComboBox1.Items.AddRange(items);
       }
   ```

4. 运行程序并点击按钮。
5. 点击C1ComboBox控件的下拉按钮，项目中的字符串显示在下拉列表中。



删除C1ComboBox项目

可以以编码的方式或在设计时通过Strings Collection Editor方便的删除C1ComboBox的所有或特定项目。

以编码的方式删除所有项目

要以编码的方式删除 C1ComboBox的所有项目，完成如下步骤：

Visual Basic

```
Visual Basic
c1ComboBox1.Items.Clear()
```

C#

```
C#
c1ComboBox1.Items.Clear();
```

以编码的方式删除一个项目

使用Remove或 RemoveAt方法以编码的方式删除一个项目。Remove方法删除一个指定的或者被选择的项目。RemoveAt方法删除在指定位置的项目。

下列代码演示了如何使用Remove和RemoveAt方法删除指定的或者被选择的项目：

Visual Basic

```
Visual Basic
' To remove item with index 0:
c1ComboBox1.Items.RemoveAt(0)
' To remove currently selected item:
c1ComboBox1.Items.Remove(ComboBox1.SelectedItem)
' To remove "Chicago" item:
c1ComboBox1.Items.Remove("Chicago")
```

C#

```
C#
// To remove item with index 0:
comboBox1.Items.RemoveAt(0);
// To remove currently selected item:
```

```
comboBox1.Items.Remove(comboBox1.SelectedItem);
// To remove "Chicago" item:
comboBox1.Items.Remove("Chicago");
```

通过SelectedItemChanged事件用数据填充C1ComboBox

要在组合框中一个特定项目被选中时填充C1ComboBox，按如下步骤使用SelectedItemChanged事件：

1. 在窗体上添加两个C1ComboBoxe控件。
2. 使用String Collection Editor 在第一个C1ComboBox控件中添加字符串 “Pittsburgh”。
3. 在C1ComboBox的Properties窗口中双击SelectedItemChanged事件来创建SelectedItemChanged事件的事件处理程序。
4. 在SelectedIndexChanged事件处理程序中添加如下代码：

Visual Basic

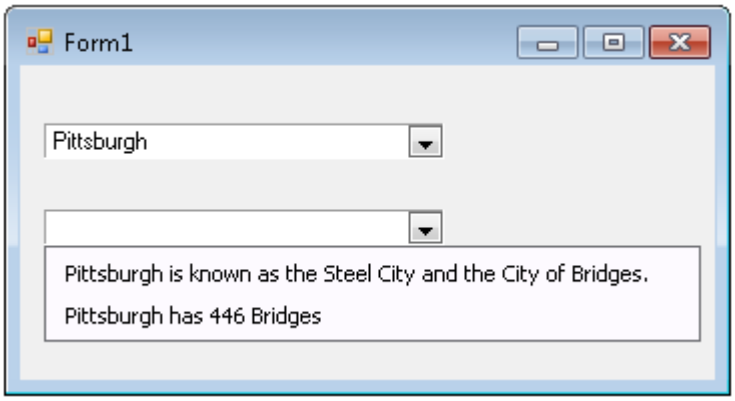
```
Visual Basic
Private Sub comboBox1_SelectedItemChanged(sender As Object, e As EventArgs)
  c1ComboBox2.Items.Clear()
  If c1ComboBox1.SelectedItem.ToString() = "Pittsburgh" Then
    c1ComboBox2.Items.Add("Pittsburgh is known as the Steel City and the City of Bridges.")
    c1ComboBox2.Items.Add("Pittsburgh has 446 Bridges")
  Else
    c1ComboBox2.Items.Add("You did not select Pittsburgh.")
  End If
End Sub
```

C#

```
C#
private void comboBox1_SelectedItemChanged(object sender, EventArgs e)
    {
      c1ComboBox2.Items.Clear();
      if (c1ComboBox1.SelectedItem.ToString() == "Pittsburgh")
      {
        c1ComboBox2.Items.Add("Pittsburgh is known as the Steel City and the City of Bridges.");
        c1ComboBox2.Items.Add("Pittsburgh has 446 Bridges");
      }
      else
      {
        c1ComboBox2.Items.Add("You did not select Pittsburgh.");

      }
    }
```

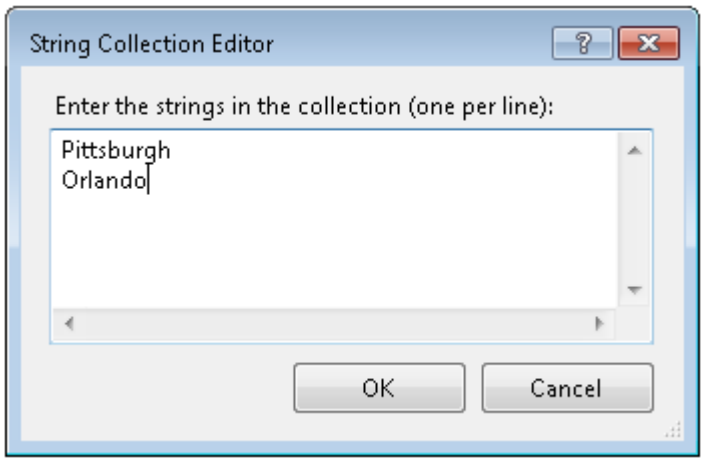
5. 运行程序然后再第一个C1ComboBox中选择Pittsburgh。
6. 点击第二个C1ComboBox的下拉按钮。注意到按照SelectedItemChanged事件处理程序项目已经被添加到下拉列表中。



通过SelectedIndexChanged事件使用数据填充C1ComboBox

要在组合框一个特定位置的项目被选中时填充C1ComboBox，按如下步骤使用SelectedIndexChanged 事件：

1. 在窗体上添加两个C1ComboBox控件。
2. 在第一个C1ComboBox中通过String Collection Editor 一行一行的添加如下图所示的项目：



3. 在C1ComboBox的Properties窗口中双击SelectedIndexChanged事件来创建SelectedIndexChanged事件的事件处理程序。在SelectedIndexChanged事件处理程序中添加如下代码：

Visual Basic

Title Text

```
Private Sub comboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
  c1ComboBox2.Items.Clear()
  If c1ComboBox1.SelectedIndex.ToString() = "1" Then
    c1ComboBox2.Items.Add("You selected the second item.")
  Else
    c1ComboBox2.Items.Add("You did not select the second item.")
  End If
End Sub
```

C#

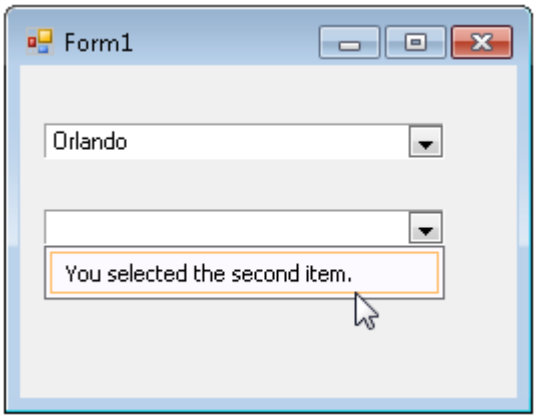
C#

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
        c1ComboBox2.Items.Clear();
        if (c1ComboBox1.SelectedIndex.ToString() == "1")
        {
            c1ComboBox2.Items.Add("You selected the second item.");

        }
        else
        {
            c1ComboBox2.Items.Add("You did not select the second item.");

        }
    }
```

4. 运行程序然后在第一个C1ComboBox中选择第二个项目“Orlando”。
5. 点击第二个C1ComboBox的下拉按钮，注意到按照SelectedIndexChanged事件处理程序项目已经被添加到下拉列表中。



编辑掩码

当您设置EditMask属性为一个掩码字符串时，C1Input控件支持掩码输入。如果你定义了一个编辑掩码，在控件中每一个字符位置将映射到一个特殊的占位符或者文本字符。文本字符，或者说文本，可以对所使用的数据类型起到提示作用。例如，在一个电话号码的格式中，包围电话号码区号的圆括号以及短线就是文本字符：(412)-123-4567.编辑掩码阻止您输入无效字符，并提供了对用户界面的其它增强。

为了启用掩码输入，设置EditMask属性的值为一个由占位符和文本字符组成的掩码字符串，可用的占位符如下表所示：您也可以通过CustomPlaceholders集合定义您自己的占位符。

尽管在简单情况下设置EditMask就已经足够，同时还提供了一个包含一组用来控制掩码输入各种重要方面子属性的MaskInfo属性。其中一个就是上面所提到的CustomPlaceholders集合。其他一些属性为：

属性	描述
AutoTabWhenFilled	如果为True，当掩码填充满时，焦点将自动移动到下一个控件。默认值：False。
PromptChar	在空白输入位置显示的提示字符。默认值：'_'。
SaveBlanks	如果为True，则存储文本包含空白位置，使用StoredEmptyChar定义的值。默认值：False。
SaveLiterals	如果为True（默认值），存储的文本（StoredContent）将包含文本字符。
ShowLiterals	控制当用户输入时，文本字符出现方式的枚举设置。文本字符可以始终出现，从不出现，或者当用户输入位置到达某个文本字符时出现。

SkipOptional	如果为True（默认），可选地掩码位置可以被自动跳过，直至到达第一个允许该字符的输入位置。
StoredEmptyChar	存储在空白掩码位置的字符。默认值：'_'。

关于掩码相关属性的完整列表，请参见MaskInfo类。

如果ShowLiterals属性设置为FreeFormatEntry，则可选掩码位置可以完全被忽略；不需要填充这些空白字符。

C1Input中使用的掩码字符（占位符）和Microsoft Access以及Microsoft ActiveX MaskEdit控件中使用的占位符类似（使用CustomPlaceholders集合可以定义更多的占位符）：

占位符	描述
#	数字占位符表示该位置允许一位数字或加减号（是否输入为可选）。
.	小数点占位符。实际使用的字符是您的国际化设置所指定的小数点占位符。该字符作为掩码处理为文本字符。
,	千位分隔符。实际使用的字符是您的国际化设置所指定的千位分隔符。该字符作为掩码处理为文本字符。
:	时间分隔符。实际使用的字符是您的国际化设置所指定的时间分隔符。该字符作为掩码处理为文本字符。
/	日期分隔符。实际使用的字符是您的国际化设置所指定的日期分隔符。该字符作为掩码处理为文本字符。
\	转义符号，将掩码字符串中的下一个字符作为文本字符。这可以让您在掩码中包含#，&，A，...等字符。该字符作为掩码处理为文本字符。
&	字符占位符（必须输入）。允许输入任何字符。
>	将之后所有的字符转换为大写形式。
<	将之后所有的字符转换为小写形式。
~	关闭之前的<或>设置。
!	使得跟随此编辑掩码的可选字符串从右向左显示，而不是默认的从左到右显示。所以，空格将出现在左侧。
^	关闭之前的!字符的功能。在 ^字符位置之后，空白将出现在右侧。
A	字母数字字符占位符（必须输入）。例如：a – z，A – Z，或者 0 – 9。
a	字母数字字符占位符（可选输入）。
0	数字占位符（必须输入）。例如：0 – 9。
9	数字占位符（可选输入）。
C	字符或空格的占位符（可选输入）。允许任何字符输入。
L	字母占位符（必须输入）。例如：a – z 或者 A – Z。
?	字母占位符（可选输入）。
\n	新行文本字符。当Multiline属性设置为True时适用。
"	双引号括起的包含任意字符的字符串将被视为文本字符串。
Literal	所有的其它符号将作为文本字符显示，即，显示为自身。

示例

上面提到的电话号码，(412)123-4567可以由编辑掩码EditMask设置为(000)000-0000表示。

验证数据

C1Input控件支持两种数据校验方式，对原始输入字符串的校验（PreValidation）以及对用户输入值的校验（PostValidation）。参见Value和Text：显示，验证，和值的更新，以解释校验过程

输入字符串验证（PreValidation）

输入字符串验证由PreValidation属性进行控制。PreValidation类允许您您使用通配符模式字符串或正则表达式字符串指定验证规则。所有的规则（字符串）通过PatternString属性指定。多个规则（子字符串）由ItemSeparator分隔（默认为'|'）。

PreValidation属性定义如何解释PatternString。

值	描述
ExactList	PatternString包含一个可能值的列表，该列表由ItemSeparator分隔。
PreValidatingEvent	PreValidating在验证过程中使用。
Wildcards	PatternString包含一组通配符模式的列表，由ItemSeparator分隔。一下为模式中的保留字符：?（任意单个字符），#（任何一个数字），*（零或多个字符），\（转义符）。您还可以使用PreValidation属性定义你自己的自定义的模式字符。
RegexPattern	PatternString包含一个正则表达式。

使用PreValidatingEvent选项，您可以在PreValidating事件中通过代码执行输入字符串校验。相关的更多信息，请参见事件描述。

如果您使用正则表达式，用到了RegexPattern选项，这里还有一个RegexOptions属性，您可以偶尔用来设置影响正则表达式匹配功能的标志。

> **注意：**输入字符串验证（PreValidation）在DateTimeInput和NumericInput模式下不使用。当启用DateTimeInput或者NumericInput模式时，输入值验证（PostValidation）将被执行。

示例

以下示例描述如何理解Validation和PatternString属性：

- Validation 属性设置为ExactList，PatternString属性设置为red|green|blue：输入的字符串必须是三个允许的字符串值之一，red，green，或者blue，如果CaseSensitive设置为False，则也可以忽略大小写。
- Validation 属性设置为Wildcards，PatternString属性设置为(412)*：输入的字符串必须以(412)开头，如果

CaseSensitive设置为False，则也可以忽略大小写。
- Validation 属性设置为RegexPattern，PatternString设置为 [0-9]*：输入字符串包含一个或多个数字。

输入的值验证（PostValidation）

PostValidation允许您校验由用户输入的Value。PostValidation类允许您：

- 检查该值匹配由Values属性指定的预定义值列表中的某一个值。
- 检测该值，查看是否其小于最小值或者大于最大值，即您可以检测该值属于某一个区间。

你甚至可以指定多个区间的有效值。这些区间由Intervals属性定义，再此您可以指定每一个区间的最大和最小值，是否使用或者忽略最大值或最小值，以及是否不相等是严格匹配的（包括最大值和最小值）。

- 排除一些特殊值可以使用ValuesExcluded属性。
- 以编程方式在PostValidating事件中执行校验。

为区分声明式校验和编程式校验，请使用PostValidation属性，该属性具有两个可选的值：ValuesAndIntervals以及PostValidatingEvent。注意，PostValidatingEvent 将禁用值和值区间的自动校验。如果您希望在事件处理代码中联合使用值和值的区间校验，可以在事件处理代码中调用ValidateValuesAndIntervals 方法。

编辑日期和时间值

C1TextBox支持一种特殊的编辑模式叫做DateTimeInput 模式，使得最终用户可以更加容易地编辑日期和时间类型的值。当DataType属性设置为DateTime并且DateTimeInput属性设置为True（默认值）
时，该模式启用。在DateTimeInput模式下，当前选中的日期或时间字段，比如年，月，日，等等，将高亮显示并单独编辑。格式化的字段以字符串形式表示，比如说LongDate格式的月份或星期几，可以通过键盘以数值形式输入，同时它们的字符串形式将自动更新。向上/向下箭头键或鼠标滚轮可以用来增加/减少当前字段的值。

用来控制输入日期和时间的附加属性：

属性	描述
MinShortYear	可以输入的最小年份，不具有前导零（当DateTimeInput设置为True时）。例如，如果MinShortYear设置为300（默认值），则输入200则是不允许的（将被忽略），而400将被解释为公元0400年。无论这个属性的值为多少，输入0200将被解释为公元0200年。
CurrentTimeZone	默认情况下该属性的值为True，这意味着此日期时间值是不变的，没有按照时区进行调整。如果此属性设置为False，则显示给用户的文本和底层存储的值是不同的。存储的值属于由GMTOffset属性指定相对于格林威治时间时区偏移时间的时区。显示给用户的文本属于由本机设置定义的时区所在的本地时间。显示值并解析用户输入的值，C1Input将依据不同的时区进行调整。

为了使得编辑日期时间值更加方便和容易，您可以使用特定的C1DateEdit控件。除了C1TextBox提供的功能之外，它还支持下拉日历以及上/下按钮（speedbuttons）用来增加/减少当前选中的日期时间字段的值。

编辑数值

C1TextBox支持一种特殊的编辑模式叫做NumericInput模式，这将使得编辑数值类型的值非常容易。当NumericInput设置为True（默认）时，该模式启用，值的类型为某一种数值类型（Byte，UInt16，UInt32，UInt64，SByte，Int16，Int32，Int64，Decimal，Single，Double）。NumericInput 模式下，数值可以通过一个计算器的方式进行编辑。它只接受数字，+ / -符号，以及，如果数据类型和格式允许，小数点和指数。其他字符，如字母，将被忽略。NumericInput模式下，为单精度和双精度值，同时还支持若干特殊的功能键。F9（改变符号），F2（负无穷大），F3（正无穷大），F4（NaN，“不是数值”）。

为了让用户编辑数值时更加方便，您可以使用特定的C1NumericEdit控件。除了C1TextBox提供的功能之外，它还额外支持了下拉计算器以及上/下按钮（speed buttons），用来按照指定的Increment增加/减少值。

下拉以及值增加按钮

为编辑日期和时间以及数值类型的值的专门的C1Input控件，C1DateEdit以及C1NumericEdit控件，支持下拉以及增加/减少（上/下）按钮。

按钮的是否可见由ShowDropDownButton以及ShowUpDownButtons属性控制。

为控制下拉按钮的对齐方式以及距离控件的距离，您可以使用DropDownAlign以及GapHeight属性。为了通过编程方式打开/关闭下拉框，请使用OpenDropDown以及CloseDropDown方法。打开/关闭下拉框同时将触发事件DropDownOpened以及DropDownClosed。你可以在下拉框显示给用户之前，使用DropDownOpened 事件调整下拉属性（主要是，Calendar的属性）。您可以通过DroppedDown属性检查是否下拉框处于打开状态。

C1DateEdit控件

C1DateEdit控制支持上/下按钮和下拉日历。

如果DateTimeInput属性设置为True，则上/下键功能启用。它们的递增/递减日期时间当前选定的字段的值，参见编辑日期和时间值。

下拉日历具有和标准的MonthCalendar（System.Windows.Forms.MonthCalendar）相同的对象模型，同时，额外的按钮，比如说Clear，Today以及两个年份导航按钮则几乎和标准的日历上的外观相同。

按钮的可见性由ShowClearButton以及ShowTodayButton属性控制。这些属性以及其他的全部日历的属性可以在设计时修改，也可以通过Calendar对象以编程方式修改。如果您希望在日历打开之前通过编程方式修改日历的属性，在其显示给最终用户之前，请使用DropDownOpened事件。

C1NumericEdit控件

C1NumericEdit控件支持上/下（spin）按钮以及下拉计算器。

上/下键递增/递减的值由Increment属性进行指定（默认：1）。

下拉计算器和标准的Windows计算器模型一致，允许最终用户无需离开控件进行计算。

自定义下拉功能

C1Input包含一个强大的自定义下拉框功能，除了标准的日历以及计算器下拉控件之外，该功能允许您创建所需要的任何的下拉编辑框。下拉编辑器和您工程中的窗体一样可以进行可视化编辑。

为创建您自己的自定义下拉编辑器，请使用C1DropDownControl。该控件类型从C1TextBox派生，并添加了自定义的下拉功能以及上/下按钮。

为了在您的控件中创建一个下拉编辑框：

1. 添加一个窗体到您的工程，并且使其从C1.Win.C1Input.DropDownForm派生，之后在您的C1DropDownControl上的DropDownFormClassName属性中选择此窗体的类型名称。

2. 在您的派生自DropDownForm类型的窗体中，您可以在需要的时候设置C1DropDownControl的值（通过DropDownForm.OwnerControl属性获取该控件对象），或者您可以在此窗体关闭时，通过PostChanges事件设置控件的值。

   关于自定义下拉窗体可用选项的详细描述，请参见DropDownForm类参考。同时可以参见 C:\Documents and Settings\<username>\My Documents\ComponentOne Samples 目录（默认安装）下的自定义下拉功能的通用用户示例。

如果您需要创建一个带有下拉功能的自定义控件，这可以通过从C1DropDownControl 派生一个自定义的控件类型，并覆盖默认的C1DropDownControl.DefaultDropDownFormClassName属性实现。

可编程的格式化，解析以及验证

如果标准的以及自定义的格式不足以满足您的需求，您可以通过设置FormatType属性为UseEvent，然后通过代码在Formatting事件中对值进行格式化，请参见格式化数据。在您的格式化代码中，您可以使用标准的C1Input 格式化逻辑作为一个辅助或者其他任意目的，只需要调用Format方法。

解析同样可以在Parsing事件的事件处理代码中完成，通过设置FormatType属性为UserEvent。如果需要，您可以使用标准的C1Input解析例程，使用以下的ParseInfo方法：Parse，ParseFixed，ParseFloat，ParseInteger，ParseBoolean 以及ParseDateTime。

MaskInfo类也提供了一些有用的编辑掩码管理的方法。

当您需要将Value属性和当前用户输入的文本进行同步，请使用UpdateValueWithCurrentText方法。通常，这一同步过程在空间失去焦点是自动完成，但是在某些特定的情况下，您会需要调用此方法并强制更新Value属性的值。更新Value属性的过程包括解析输入文本，校验，并更新Value属性的值，参见值和文本：显示，验证，更新Value。您也可以仅执行前面两个阶段，解析和校验，而不改变Value的值，通过使用ParseContent以及CheckValidationResult方法。

错误处理

在数据输入表单上，错误处理非常重要。C1Input向开发人员提供了对各种不同的错误条件进行完全控制的能力，比如以下主题所列举的错误。

错误处理

WinForms数据源，比如说ADO.NET以及C1DataObjects，包含一些检测数据逻辑错误的规则，这些规则可以来自于数据源自身或者程序员后期添加，设置RowError属性或者调用SetColumnError方法（C1DataObjects上的SetFieldError ）。您可以使用System.Windows.Form.ErrorProvider组件，显示逻辑的行和列的错误。

为了显示逻辑的列错误，请将ErrorProvider和您希望使用错误提示的C1Input控件放置在一起。

为了在C1DbNavigator控件显示行错误，设置一个ErrorProvider组件的ErrorProvider属性。之后当当前行存在一个错误时，C1DbNavigator将显示一个带有RowError的ToolTip文本的错误图标。

错误的数据显示格式

错误的格式是完全可能发生的，尽管通常在应用程序中已经尽量的避免，但是从数据库或者其他数据源获取的数据仍然可能不匹配C1Input控件定义的编辑掩码。在这种情况下，控件无法显示其格式化的Value属性。尽管C1Input控件具有合理的默认行为对此情况进行处理，您可能也会希望

通知用户有无效数据发生。这可以通过ErrorProvider属性（C1Label控件的(ErrorProvider）完成。如果您设置该属性为一个ErrorProvider组件，C1Input将使用该组件，当其显示无效数据（数据无法被格式化显示在控件上）时，通知错误发生。当检测到错误时，它将调用ErrorProvider.SetControl方法。在此之前，C1Input将触发FormatError事件，在此事

件中，您可以自定义错误消息（ErrorProvider ToolTip文本）的内容，同时可以执行其他动作。

用户输入错误

当C1Input在解析或校验输入值时检测到一个错误，它将触发ValidationError事件。紧接着，默认情况下，它将显示一条错误信息。默认行为是可以改变的，并且以不同的方式进行定制：

C1Input控件具有一个ErrorInfo属性，包含影响错误处理的设置（ErrorInfo类的属性）：

属性	描述
BeepOnError	如果设置为True，则控件会发出一声蜂鸣，以提示有错误发生。默认：False。
CanLoseFocus	如果设置为True，则控件不管是否存在错误，都允许丢失焦点。默认情况下该属性的值为False，表示控件将使中保持有焦点状态，直到错误被纠正。注意如果设置ErrorAction为SetValueOnError或者ResetValue，则会在错误发生之后重置控件的值，之后控件可以正常失去焦点。
ErrorAction	一个枚举值，决定当控件发生错误时，对控件的值进行处理。ErrorAction设置为None（默认值）意味着Value将保持不变，继续保留为失败的值更新之前的状态。如果ErrorAction设置为SetValueOnError，则控件的Value属性将设置为在ErrorInfo类上的ValueOnError属性指定的值。如果ErrorAction设置为ResetValue，则控件的值将设置为控件进入编辑态之前的值。设置ErrorAction属性为ThrowException将中断程序执行，并抛出一个异常，异常的类型为ValidationException。
ErrorMessage	错误信息显示在标准的消息对话框和/或者在异常信息中。
ErrorMessageCaption	显示在错误消息对话框标题栏上的文本。
ErrorProvider	获取或设置一个ErrorProvider对象，用来指示控件的错误状态。
ShowErrorMessage	如果设置为True（默认值），则显示标准的错误消息。
ValueOnError	如果ErrorAction设置为SetValueOnError，用来重置控件的值。
ValueOnErrorIsDbNull	一个布尔类型的属性，用作设置ValueOnError为DBNull（仅在设计时有用）。

除此之外，ErrorInfo.ErrorMessage可以被特定的动作指定：编辑掩码错误（MaskInfo.ErrorMessage），解析（ParseInfo.ErrorMessage），pre-以及post-校验（PreValidation.ErrorMessage，PostValidation.ErrorMessage）。如果一个特定的错误消息没有在这些子对象中指定，则控件上的ErrorInfo.ErrorMessage将生效。注意，您如果您设置ErrorProvider属性为一个ErrorProvider组件，同时设置ShowErrorMessage为False，则可以使用ErrorProvider的图标显示错误，而不再是显示一个消息对话框。

以上所列举的属性，一旦设置在控件的ErrorInfo对象上，将影响该控件全部的错误处理。当错误发生时，这些值可以通过编程方式自定义以处理特定的错误。这通过传递ErrorInfo参数至ValidationError事件实现。传递给ValidationError事件的ErrorInfo参数是控件上ErrorInfo对象的一个副本，包含了其全部的属性设置。它是一个独立的副本，因此您可以修改

位于事件参数中的ErrorInfo对象的属性，仅针对当前的错误，而不会影响控件整体的ErrorInfo设置。通过设置ValidationError事件中的ErrorInfo属性，您可以指定如何处理错误。例如，您可以通过设置ShowErrorMessage属性值为False，以禁止标准的错误消息（作为替代，显示您自己的消息通知），或者您可以修改ValueOnError（设置ErrorAction为SetValueOnError），在或者修改ErrorMessage的内容。请记住，您必须设置传递给ValidationError事件的ErrorInfo参数，而不是控件上的ErrorMessage属性。

在ValidationError事件之后，错误处理将按照在事件参数中指定的ErrorInfo参数的值进行处理。如果ErrorAction设置为ThrowException，则将抛出一个异常（使用ErrorMessage属性的文本）。如果BeepOnError设置为True，则控件发出一声蜂鸣。如果ShowErrorMessage设置为True，则将显示标准的错误消息（使用ErrorMessage的文本）。在此之后，控件的值可能会发生变化，依照ErrorAction指定的行为。最后，如果校验是由于尝试移动焦点至其他控件的行为所触发，移动焦点至其他控件的行为要么会取消，要么允许，按照CanLostFocus的不同设置。如果ErrorProvider属性设置为一个ErrorProvider组件，该组件将用作显示如果errorprovidererrorprovider属性设置为一个组件，该组件将在出错的控件附近显示一个错误图标，同时具有一个ErrorMessage信息的工具提示（调用ErrorProvider.SetError方法）。

如果您通过编程方式在某个事件的处理代码中执行了解析或者校验，并且在离开该事件处理函数时返回了一个错误条件（设置事件的Succeed参数为False），您可以描述该错误，并可以通过设置传递给该事件的ErrorInfo参数的属性指定如何处理该错误。类似的事件参数由以下事件提供：PreValidating，Parsing以及PostValidating。其初始值来自于控件的ErrorInfo属性。您在事件中修改的ErrorInfo参数在之后将传递给ValidationError事件，在此您可以像以上提到的那样对其做进一步的修改。

处理Null值和空值

没有适当的工具，Null值（DBNull）将很难处理。C1Input提供灵活的规则用来处理Null值，允许程序员能够解决遇到的各种实际问题。

显示Null值和空值

当控件处于非编辑状态或者为只读状态时，表示Null值的文本将按照NullText属性指定的内容显示（可以被DisplayFormat属性设置覆盖）。如果EmptyAsNull设置为True（默认值：False），则空字符串也显示和NullText字符串一样的内容。EmptyAsNull属性也可以被派生类重写。在编辑模式下，EditFormat对象的NullText以及EmptyAsNull属性生效，而不是DispalyFormat。

在具有有效的EditMask的编辑模式，Null值和空值字符串显示为一个空白掩码，由文本字符和填充未输入位置的提示字符组成。
当使用DateTimeInput属性设置为True，编辑日期事件类型的值时，Null值由空白控件表示。当用户通过一个键盘敲击或者鼠标单击开始编辑时，控件将立即变为非空值，该值为上一次指定给控件的非空值或者今日的日期。

在编程方式格式化（(FormatType设置为UseEvent）时，Formatting事件仅在非Null值上调用。

输入Null值和空值

除非在编辑掩码模式或者DateTimeInput属性设置为True的日期时间编辑模式，用户可以通过以下方式输入一个Null值：

- 如果控件的文本等于NullText，则结果值为Null。有效的NullText值由NullText决定。和NullText进行比较将是大小写敏感的，不依赖于CaseSensitive属性的设置。
- 如果C1TextBox.EmptyAsNull属性设置为True（默认为False），则清空控件，输入一个空白字符串将导致Null值出现。
  在编程方式解析时，DBNull值可以由程序员在Parsing事件中返回。

如果用户输入一个Null值（可以通过输入一个空白字符串或者和NullText相同的值，参见上文），则输入字符串校验或解析将被跳过，参见输入字符串校验（PreValidation）以及解析（Updating）数据。然而，在这种情况下PostValidation将被执行，参见输入值校验（PostValidation）。

自定义C1Input的外观

C1Input旨在让你很容易自定义它的外观。你有无限的可能性对每个C1input控件改变它的默认外观。C1input为输入框提供了大量的属性样式，以及内置的WindowsXP和Office2007的主题。

视觉样式

自定义C1Input的外观>视觉样式

C1Input有七种内置的主题：System,Office2007Blue,Office2007Black,Office2007Silver,Office2010Blue,Office2010Black,和Office2010Silver

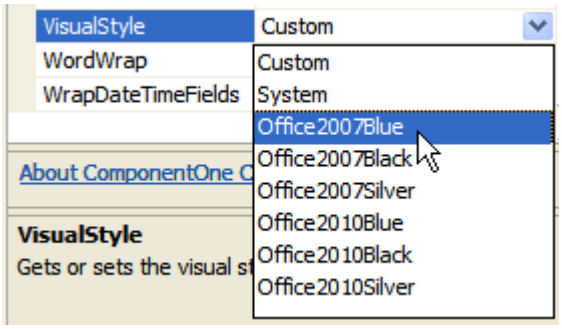
设定C1InputVisualStyle属性会控制绘制下面这些控件时使用的梯度和边框，包括C1TextBox,C1Label,C1DbNavigator,C1DropDownControl,C1DateEdit（包括下拉日历），C1NumericEdit(包括下拉计算器),C1SplitButton,C1ComboBox,和C1Button.

要使用视觉样式自定义一个C1input控件的外观，需要设定VisualStyle为Custom,Office2007Black,Office2007Blue,Office2007Silver,System,Office2010Blue,Office2010Black,或者Office2010Silver.这个属性既可以在设计器中设定也可以在代码中设定。下面的表描述了每一个视觉样式。

视觉样式	描述
Custom	没有视觉样式(像往常一样使用风格和外观属性)。
Office2007Black	样式符合Office2007黑色主题。
Office2007Blue	样式符合Office2007蓝色主题。
Office2007Silver	样式符合Office2007银色主题。
System	样式符合现在的系统设定。
Office2010Blue	样式符合Office2010蓝色主题。
Office2010Black	样式符合Office2010黑色主题。
Office2010Silver	样式符合Office2010银色主题。

使用设计器

在属性窗口中找到VisualStyle属性并将它设定为Custom,Office2007Black,Office2007Blue,Office2007Silver,System,Office2010Blue,Office2010Black,或者Office2010Silver。在下面的例子中，VisualStyle属性设定为了Office2007Blue。



使用代码编辑器

在Form_Load事件中添加代码将VisualStyle属性设置为Custom,Office2007Black,Office2007Blue,Office2007Silver,或者System。下面的代码将一个C1TextBox控件的VisualStyle属性设定为Office2007Blue：

Visual Basic

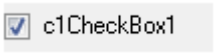
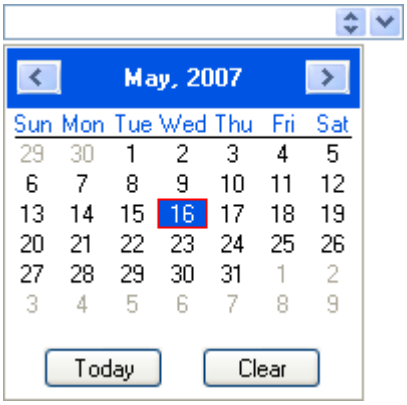
```
Visual Basic
Me.C1TextBox1.VisualStyle = C1.Win.C1Input.VisualStyle.Office2007Blue
```

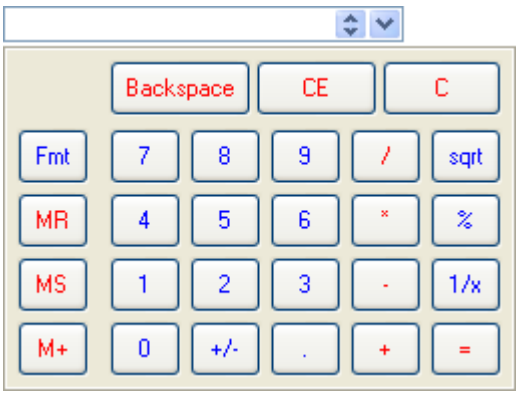
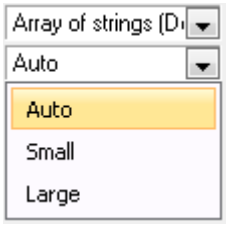
C#

```
C#
this.c1TextBox1.VisualStyle = C1.Win.C1Input.VisualStyle.Office2007Blue;
```

自定义视觉样式

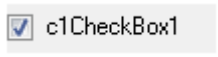
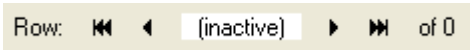
没有使用视觉样式。

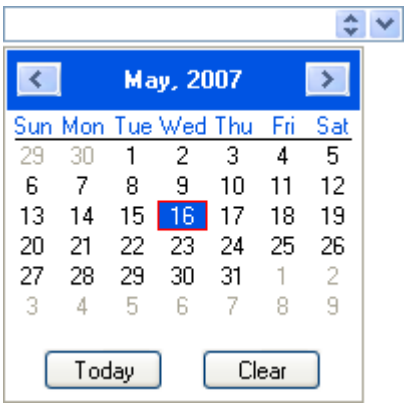
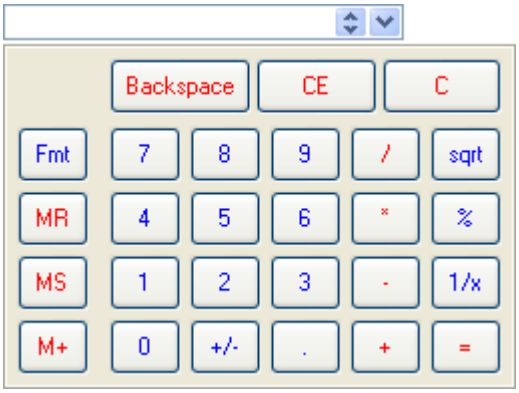
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	
C1DateEdit	
C1NumericEdit	

	
C1Button	
C1ComboBox	

系统的视觉样式

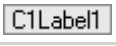
现在系统的设定

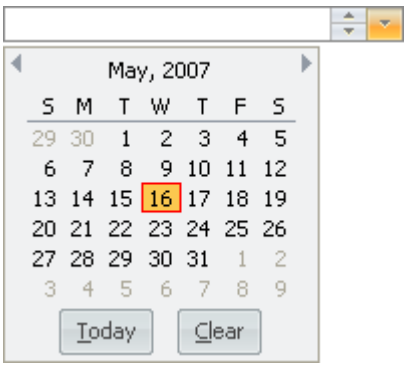
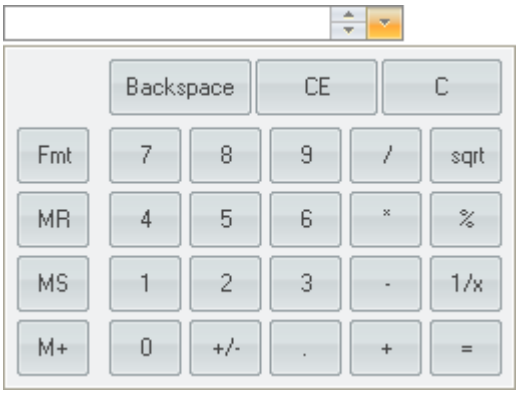
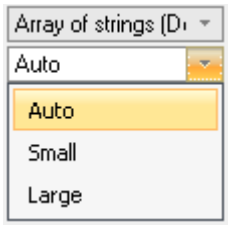
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	
C1DateEdit	

	
C1NumericEdit	
C1Button	
C1ComboBox **注意：为了演示，显示了两个combobox**	

Office2007Black视觉样式

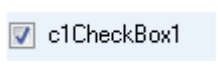
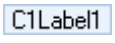
Office2007黑色主题

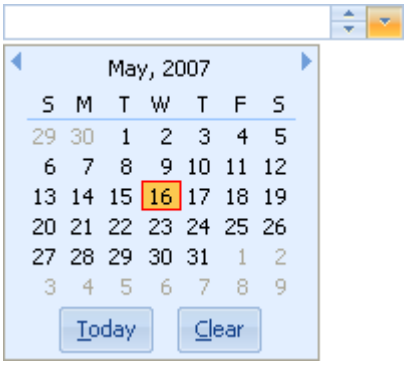
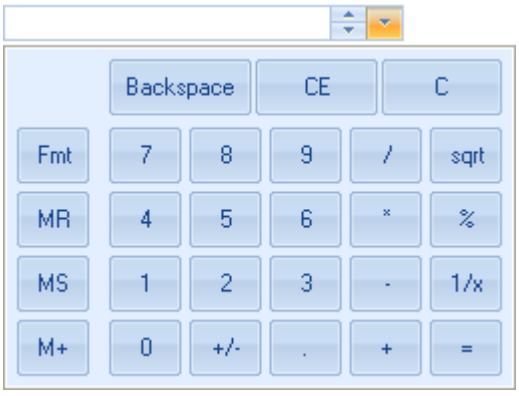
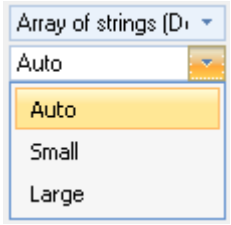
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox **Note:** For illustration purposes two comboboxes are shown.	

Office2007Blue 视觉样式

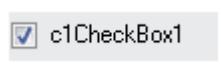
TOffice2007蓝色主题

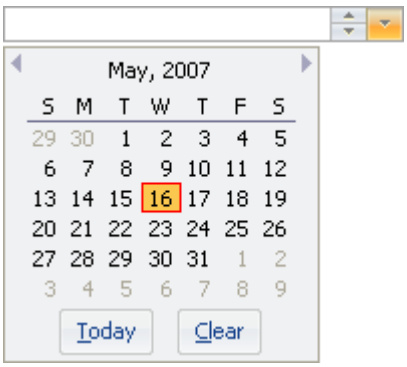
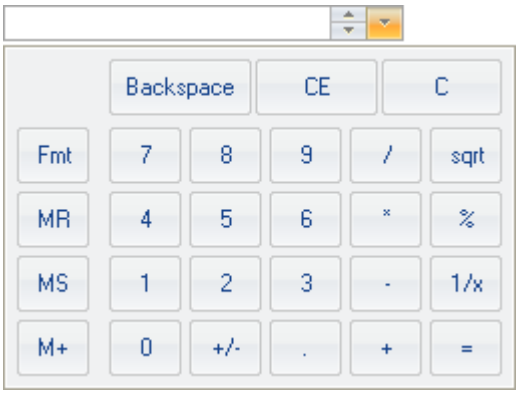
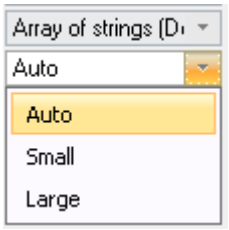
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox **注意：**为了演示，显示了两个combobox	

Office2007Silver 视觉样式

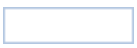
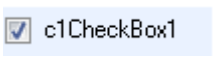
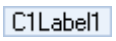
Office2007银色主题

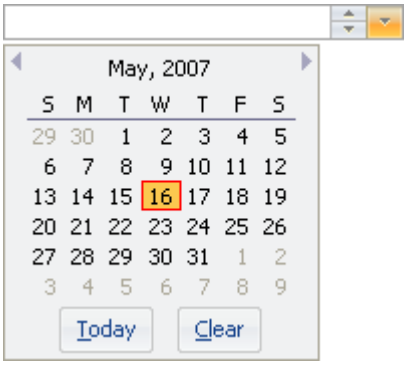
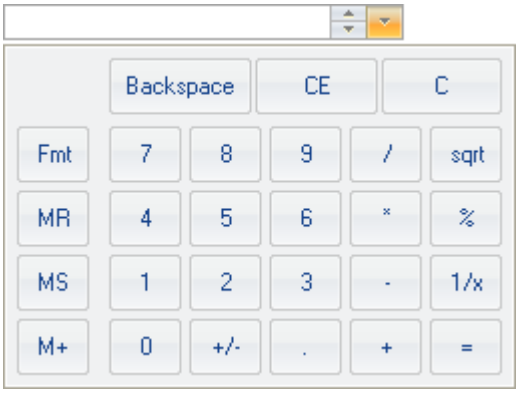
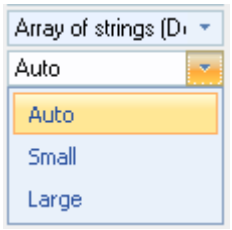
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox **注意：**为了演示，显示了两个combobox	

Office2010Blue 视觉样式

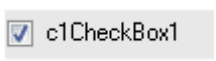
Office2010蓝色主题。

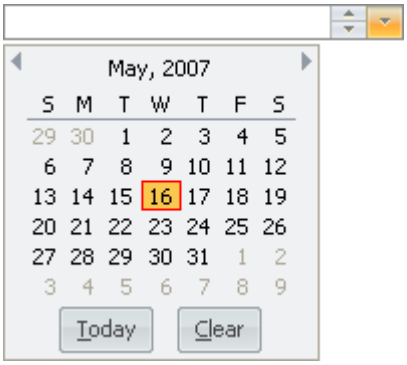
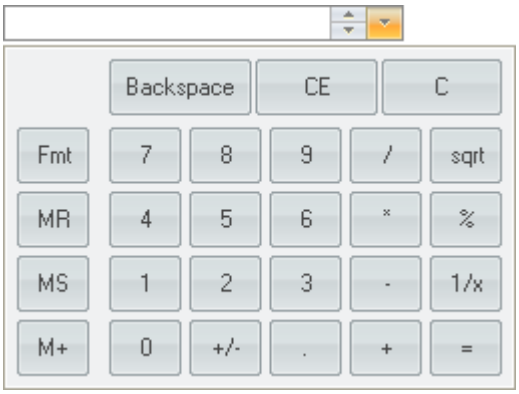
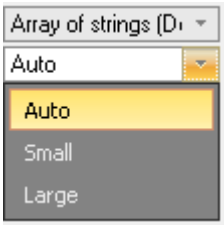
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox 注意：为了演示，显示了两个combobox	

Office2010Black 视觉样式

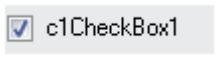
Office2010蓝色主题。

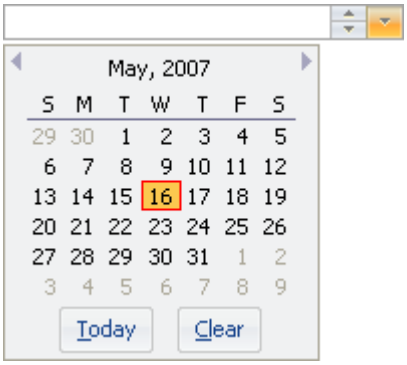
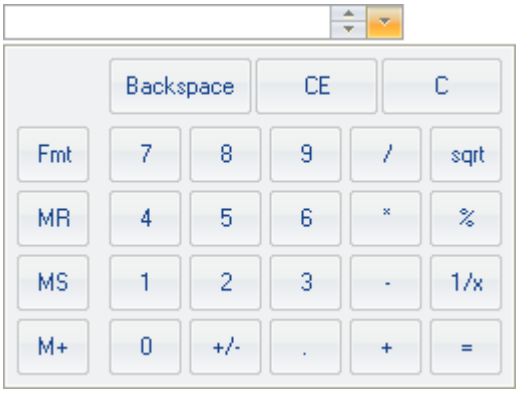
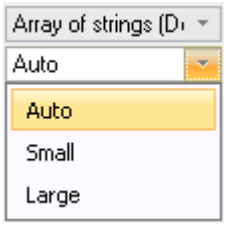
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox 注意：为了演示，显示了两个combobox	

Office2010Silver 视觉样式

Office2010银色主题。

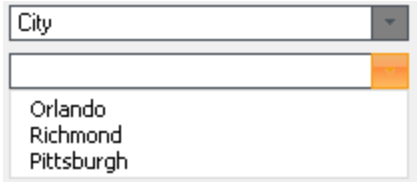
C1TextBox	
C1CheckBox	
C1Label	
C1DbNavigator	
C1DropDownControl	

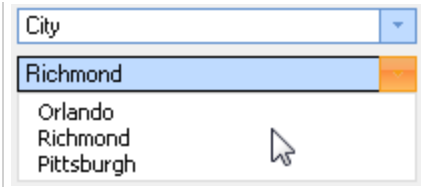
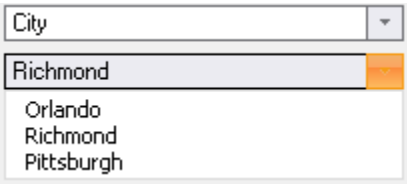
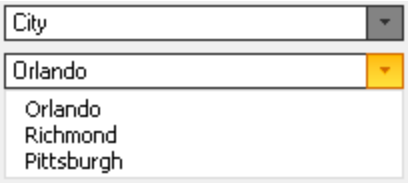
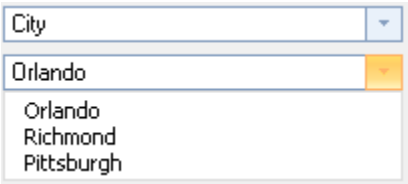
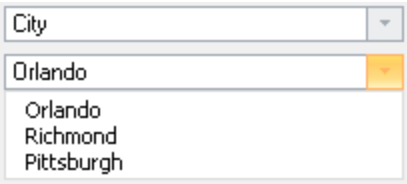
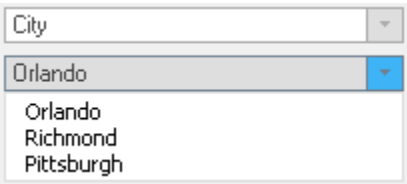
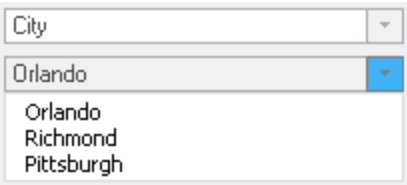
	
C1DateEdit	
C1NumericEdit	
C1Button	
C1ComboBox 注意：为了演示，显示了两个combobox	

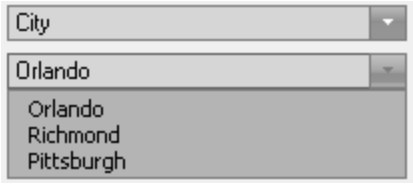
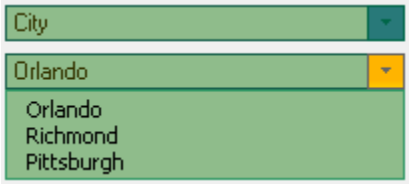
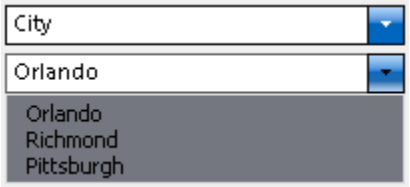
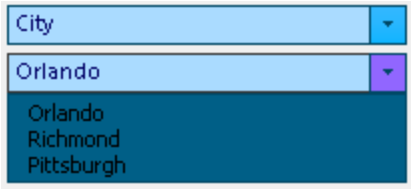
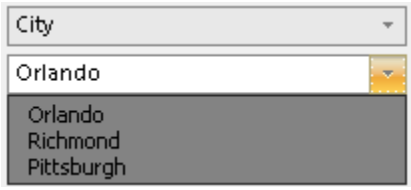
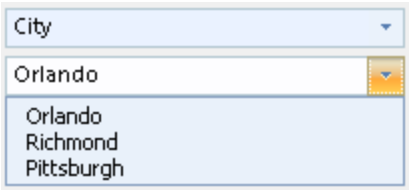
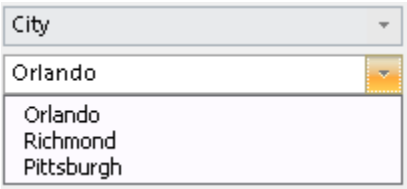
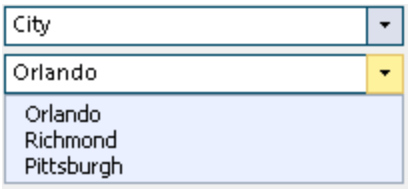
主题

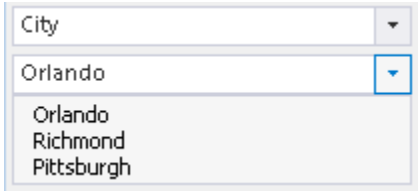
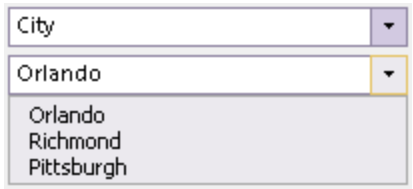
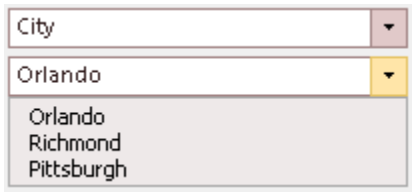
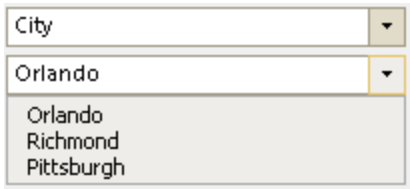
除了视觉样式，你还可以使用C1ThemeController给C1Input控件提供其他的主题。.你也可以用ThemeDesigner创建你自己的主题。

要使用主题自定义C1Input的外观，你需要将C1ThemeController添加到你的控件托盘并且将Themes属性设定为下表列出的预先定义好的样式之一。

主题名称	图片
Office2007Black	

Office2007Blue	
Office2007Silver	
Office2010Black	
Office2010Blue	
Office2010Silver	
Office2013DarkGray	
Office2013LightGray	
Office2013White	
ExpressionDark	

ExpressionLight	
GreenHouse	
RainerOrange	
ShinyBlue	
Violette	
VisualStyleOffice2010Black	
VisualStyleOffice2010Blue	
VisualStyleOffice2010Silver	
VS2013Blue	

VS2013Dark	
VS2013Light	
VS2013DarkSolar	
VS2013Green	
VS2013Purple	
VS2013Red	
VS2013Tan	

设计时如何将样式提供给C1Input控件

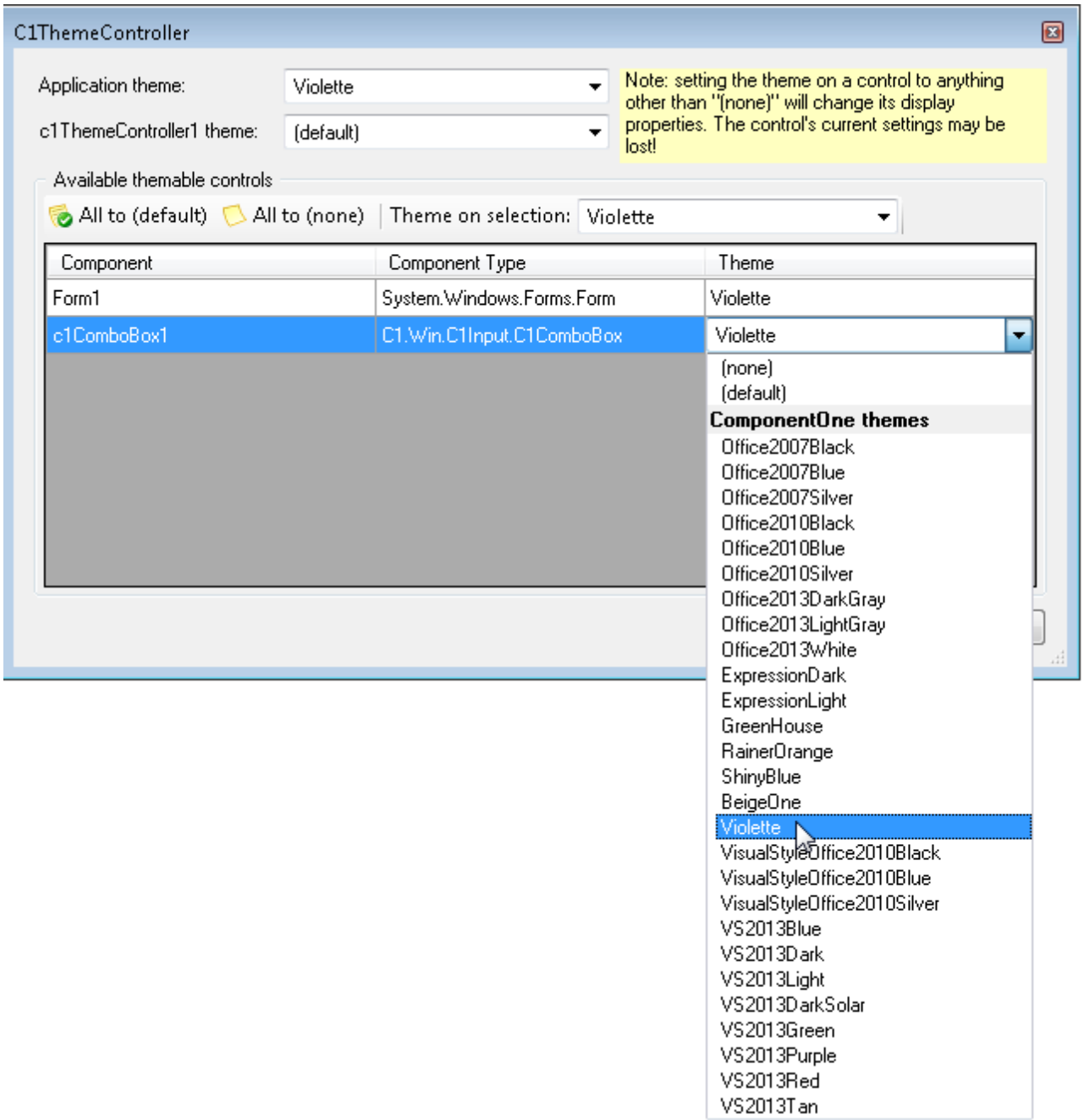
要使用带有任何C1Input控件的C1ThemeController组件，要完成下面的步骤：

1. 在设计时，在窗体上添加任何的C1Input控件，例如C1ComboBox。
2. 将C1ThemeController组件添加到你的控件托盘。C1ThemeController对话框就会出现了。

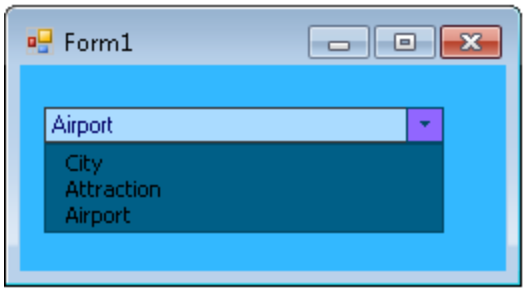
> 如果你使用C1ThemeController2.0组件，一个C1ThemeController对话框就会出现。C1ThemeController对话框用于给应用程序中所有可以设主题的控件快速地设定主题，或者窗体上所有可以设定主题的控件设定主题，或者给不同的控件设定不同的主题。

C1ThemeController对话框列出出现在你窗体上的所有控件。在你添加C1ThemeController对话框之前如果你的窗体上有可以设定主题的控件，该控件就会出现在对话框中。每个控件/组件初始设定为none，这样这些控件的属性就不会设定到不希望的值上了。

3. 3. 在C1ThemeController对话框中点击在Form1和c1ComboBox1旁边的Theme下拉菜单，并选择一个预定义的主题，例如，Violette。



4. 点击OK来保存和关闭C1ThemeController对话框。
5. 运行程序，你就会看到你的Form和C1Combobox按Violette主题显示。



如何用编程给C1Input控件设定主题

下面代码说明了如果编程时使用RegisterTheme和etTheme方法设定内置的主题：

Visual Basic

```
Visual Basic
'Register the theme file with the C1ThemeController
C1.Win.C1Themes.C1ThemeController.RegisterTheme("C:\Users\Documents\Visual Studio 2010\Projects\ThemesProject\ShinyBlue.c1theme")
'Apply it to a control, use the theme name, not the file name
Me.c1ThemeController1.SetTheme(c1ComboBox1, "ShinyBlue")
```

C#

```
C#
//Register the theme file with the C1ThemeController;

C1.Win.C1Themes.C1ThemeController.RegisterTheme(@"C:\Users\Documents\Visual Studio 2010\Projects\ThemesProject\ShinyBlue.c1theme");

//Apply it to a control and use the theme name
this.c1ThemeController1.SetTheme(c1ComboBox1, "ShinyBlue")
```

除了预定义的主题，你还可以使用Themes设计器定制你的主题。更多信息参见Themes forWinForms文件。

C1Themes和VisualStyle属性

许多ComponentOneWinForms控件都有一个叫VisualStyle 的属性，是定义在控件组件中的枚举类型，枚举类型中的类型和可能的值遵循通常的命名模式。典型的的情况下，叫VisualStyle的枚举类型包含例如Office2010Blue, Office2010Black,等等的值。C1Themes旨在提供一个更强大和灵活的机制来调整控件的外观。C1Themes和VisualStyle属性这两种机制有很明显的重叠，有可能发生冲突。处理这些重叠的规则如下：

- 所有的C1控件的主题部分都有一个VisualStyle属性，这个属性可以被主题所定义。
- 默认情况下对所有提供的主题，这些VisualStyle属性都会设定为'Custom'，这样VisualStyle属性就不会影响其他属性。
- 在一个主题中将VisualStyle设定为除custom以外的其他值都会将这个属性的设定同步到目标控件，并且禁止应用主题的所有其他属性。（在C1ThemeDesigner,中，实际上禁用了控件的其他主题树）。

虽然我们认识到向后兼容性和其他方面的一些考虑，定制你的应用的外观可能需要使用VisualStyle属性而不是主题，但

是我们还是推荐你尽可能得使用C1Themes，因为它是更灵活和更强大的定制应用程序外观的机制。在新的控件中对视觉样式的支持会被淘汰并将被主题所替代。

边框样式

C1Input可以使用下列边框样式:

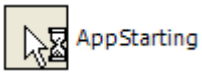
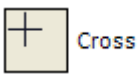
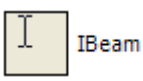
边框样式	预览
None	
FixedSingle	
Fixed3D (Default)	

边框颜色

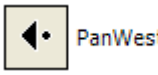
当边框样式设定为FixedSingle并且BorderColor属性的值被定义的时候，C1DateEdit,C1CheckBox,C1Label,C1NumericEdit,C1DropDownControl,和C1TextBox控件可以设定边框颜色。

光标样式

通过给Cursor属性设定一个值，你可以定制当鼠标箭头移动到控件的时候光标如何显示。你也可以通过设定ButtonCursor属性定制当鼠标移动到适用的输入类控件上的按钮的时候光标如何显示。光标样式适用于所有的C1Input控件。按钮光标样式适用于C1DropDownControl,C1DbNavigator,C1DateEdit,和C1NumericEdit控件。光标和按钮光标样式显示成下面的样子：

按钮光标样式	预览
AppStarting	
Arrow	
Cross	
Default	
IBeam	
No	

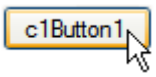
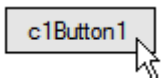
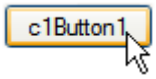
	 No
SizeAll	 SizeAll
SizeNESW	 SizeNESW
SizeNS	 SizeNS
SizeNWSE	 SizeNWSE
SizeWE	 SizeWE
UpArrow	 UpArrow
WaitCursor	 WaitCursor
Help	 Help
HSplit	 HSplit
VSplit	 VSplit
NoMove2D	 NoMove2D
NoMoveHoriz	 NoMoveHoriz
NoMoveVert	 NoMoveVert
PanEast	 PanEast

PanNE	
PanNorth	
PanNW	
PanSE	
PanSouth	
PanSW	
PanWest	
Hand	

扁平样式

当你移动鼠标到一个按钮控件并点击的时候，C1Button提供不同的扁平样式供选择。

ButtonBase.FlatStyle属性包含下面几个值：

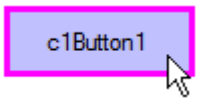
扁平样式	预览
Standard	
Flat	
Popup	
System	

当ButtonBase.FlatStyle属性设定为"Flat"，你可以使用ButtonBase.FlatAppearance属性来修改边框颜色，边框厚度，鼠

标移上去时的背景色，以及鼠标按下的背景色。ButtonBase.FlatAppearance 提供下面几个属性：

- BorderColor–定义按钮四周边框的颜色。
- BorderSize–定义按钮四周边框的大小（以像素为单位）
- MouseDownBackColor–定义当鼠标在控件的范围内按下的时候，按钮的背景色。
- MouseOverBackColor–定义当鼠标移到控件范围内的时候，按钮的背景色。

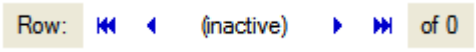
下图就是当一个C1button控件设定了BorderColor,BorderSize,和MouseOverBackColor属性的扁平样式。



按钮颜色

在C1DBNavigator控件中，使用ColorButtons你能定义是否显示蓝色的按钮。你使用ColorWhenHover属性，也能定义当鼠标移到按钮上时是否显示有颜色的按钮。

下图显示出了在C1DBNavigator控件上当ColorButtons属性设定为true的时候如何出现带颜色的按钮。



Input for WinForms的基于任务的帮助

基于任务的帮助部分假设你已经熟悉如何在VisualStudio .NET环境下编程并且已经对如何使用C1Input控件有了一般地了解。如果你对theComponentOneInputforWinForms产品是个新手，请先阅读适用于WinForm上输入的入门。

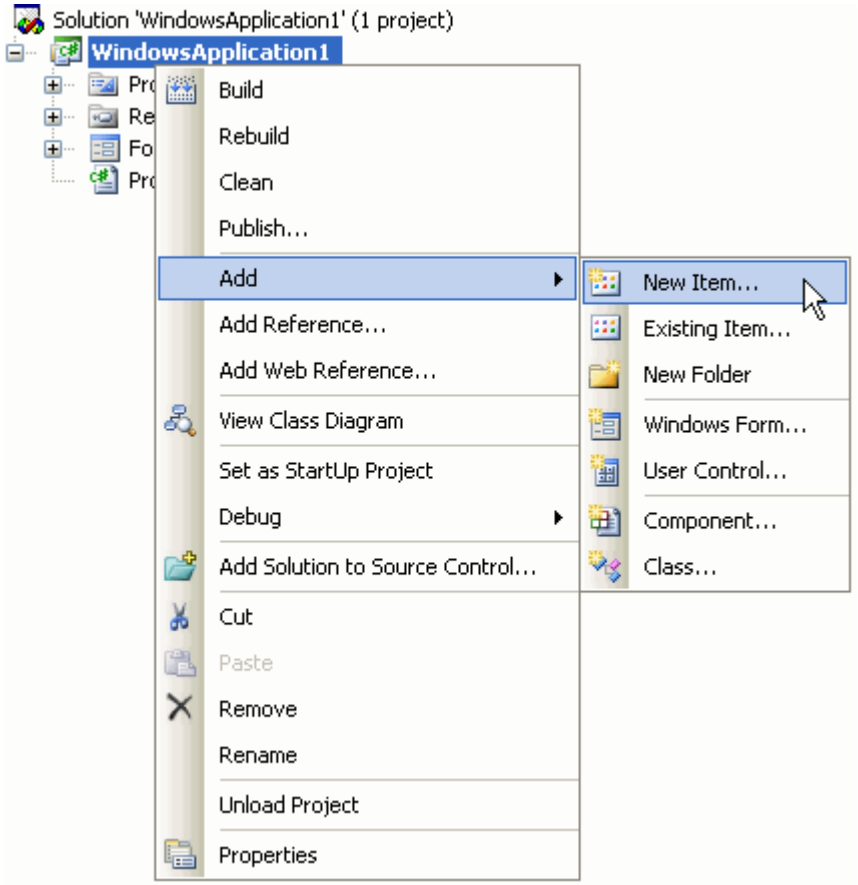
.每个主题都对使用InputforWinForms产品的特殊任务提供了一个解决方案。通过下面每个主题的一步一步描述，你将能够使用各种各样的InputforWinForms特性创建项目。

每个基于任务的帮助主题同时假设你已经创建了一个新的.NET项目。

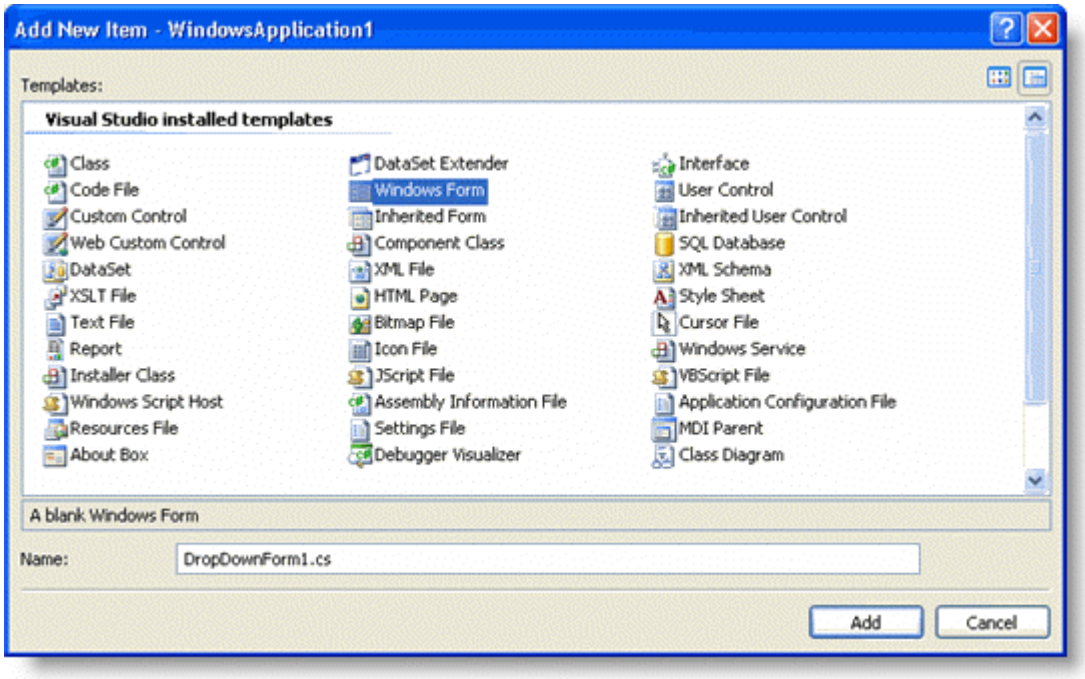
添加一个下拉列表

要给你的程序添加一个下拉列表，需要完成下面几个步骤：

1. 右键点击你的程序（位置在SolutionExplorer中）并且从Add二级菜单中选择Add New Item。



2. 在Add NewItem对话框里，从右边方框中Templates列表里选择WindowsForm。然后在Name文本框中输入DropDownForm1.cs。



3. 下面的步骤是在DropDownForm的代码中用下列类定义行替换原来的代码：

Visual Basic

```
Visual Basic
Public Class DropDownForm1
    Inherits System.Windows.Forms.Form
```

C#

```
C#
Public Class DropDownForm1:System.Windows.Forms.Form
```

替换成:

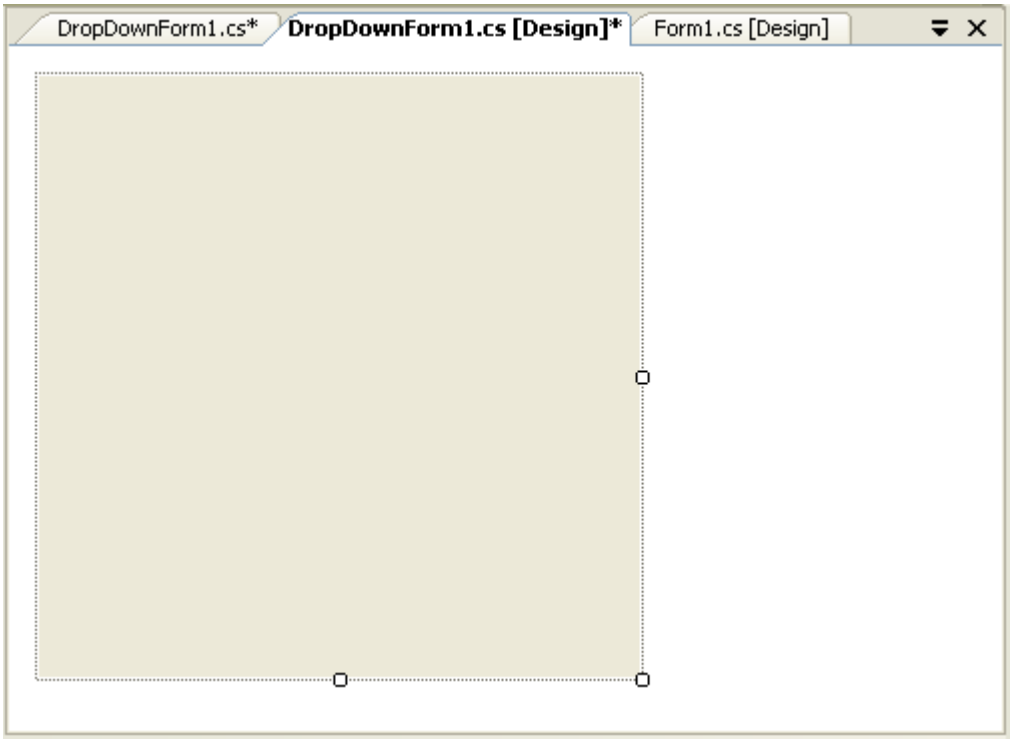
Visual Basic

```
Visual Basic
Public Class DropDownForm1
    Inherits C1.Win.C1Input.DropDownForm
```

C#

```
C#
Public Class DropDownForm1: C1.Win.C1Input.DropDownForm
```

表单编辑前应该类似下面的样子:



在导航器中改变导航

要在导航器中改变导航，象下面所示的改变索引：

Visual Basic

Visual Basic

```
Private Sub c1DbNavigator1_BeforeAction(sender As Object, e As
C1.Win.C1Input.NavigatorBeforeActionEventArgs)
        If e.Button = C1.Win.C1Input.NavigatorButtonEnum.First Then
              ' Goto second record instead of the first
               e.Index = 1
        End If

         ' Go to the last row if user entered too large position
        If e.Button = C1.Win.C1Input.NavigatorButtonEnum.Position AndAlso e.Cancel
Then
               e.Cancel = False
        End If
End Sub
```

C#

C#

```
private void c1DbNavigator1_BeforeAction(object sender,
C1.Win.C1Input.NavigatorBeforeActionEventArgs e)
    {
               if (e.Button == C1.Win.C1Input.NavigatorButtonEnum.First)
        {
```

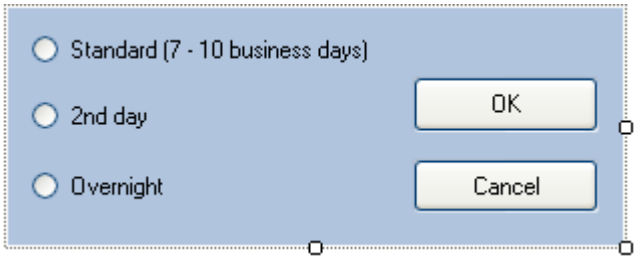
```
                // Goto second record instead of the first
            e.Index = 1;
        }

                // Go to the last row if user entered too large position
        if (e.Button == C1.Win.C1Input.NavigatorButtonEnum.Position && e.Cancel)
        {
            e.Cancel = false;
        }
    }
```

定制下拉编辑器

下面的下拉列表包含选择按钮和按钮控件以供用户从C1DropDownControl.中做选择。

下拉列表外观属性已经被编辑过所以显示成下面的样子：



在你的C1DropDownControl控件的DropDownFormClassName属性中选择列表的类名（例如，WindowsApplication1.DropDownForm1）。注意当你运行程序并且选择下拉箭头，下拉列表就会显示出来了。

让下拉列表上的按钮控件有效

1. 分别设定你的DropDownForm1上的button1和button2的AcceptButton和CancelButton属性。
2. 选择OK按钮并且设定DialogResult属性为OK。同样，选择Cancel按钮并且设定DialogResult属性为Cancel。
3. 当用户点击一个条目后，想要下拉列表关闭时改变控件的文字，添加下面的事件处理程序到PostChanges事件：

Visual Basic

Visual Basic

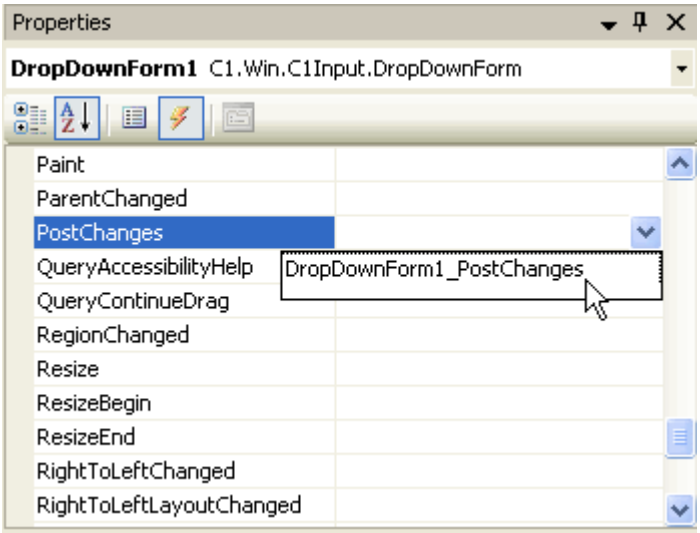
```
Private Sub DropDownForm1_PostChanges(sender As Object, e As System.EventArgs)
    If (MyBase.DialogResult = DialogResult.OK) Then
        Dim control1 As Control
        For Each control1 In MyBase.Controls
            If (TypeOf control1 is RadioButton AndAlso CType(control1,
RadioButton).Checked) Then
                MyBase.OwnerControl.Value = CType(control1, RadioButton).Text
            End If
        Next
    End If
End Sub
```

C#

C#

```
private void DropDownForm1_PostChanges(object sender, System.EventArgs e)
{
    if (DialogResult == DialogResult.OK)
    {
        foreach (Control control1 in Controls)
        {
            if (control1 as RadioButton != null &&
((RadioButton)control1).Checked)
            {
                OwnerControl.Value = ((RadioButton)control1).Text;
            }
        }
    }
}
```

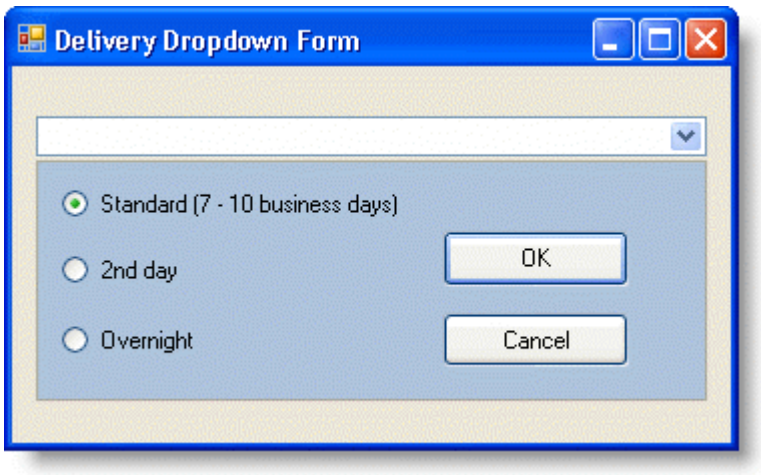
4. 在设计时，选择DropDownForm1在属性窗口查看它的属性，并且从属性工具栏框选择Events按钮 
5. 设定DropDownForm1.PostChanges事件为DropDownForm1_PostChanges.



6. 当列表打开时，想要OK按钮（button1）获得焦点，设定DropDownForm1.FocusControl属性为button1。
7. 要让Standard选择按钮选择上，在设计时选择radiobutton1按钮并且设定它的Checked属性为True。

这个主题的结果如下图：

你的列表应该显示成下面的样子。



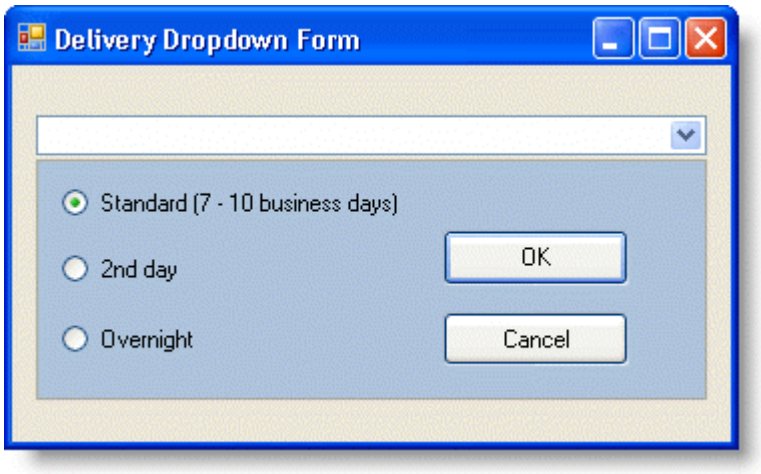
自定义C1DropDownControl

C1DropDownControl

这个主题说明了如何自定义C1Input.C1DropDownControl.

要让只有下拉按钮可见：

1. 展开VisibleButtons属性节点。
2. 设定UpDown为False。注意DropDown默认设定为True。控件显示成下图的样子。



想要下来列表的宽度和控件的宽度相同：

1. 选择下拉列表。
2. 设定Options.AutoResize为True.

绑定C1CheckBox

下面的主题说明了如何绑定C1CheckBox到一个布尔型的字段，字符串型的字段或者数字型的字段。

绑定C1CheckBox 到一个布尔型字段

使用下面的代码，在编程的时候绑定C1CheckBox 到一个布尔型的字段：

Visual Basic

```
Visual Basic
C1CheckBox1.DataSource = dt
C1CheckBox1.DataField = "ColumnBoolean"
```

C#

```
C#
C1CheckBox1.DataSource = dt;
C1CheckBox1.DataField = "ColumnBoolean";
```

绑定C1CheckBox到一个字符串型的字段

使用下面的代码，在编程的时候绑定C1CheckBox 到一个字符串型的字段：

Visual Basic

```
Visual Basic
c1CheckBox1.DataSource = dt
c1CheckBox1.DataField = "ColumnString"
c1CheckBox1.DataType = GetType(String)
' Use TranslateValues property to translate string values to/from the check box
states.
c1CheckBox1.TranslateValues.Checked = "Yes"
c1CheckBox1.TranslateValues.Unchecked = "No"
```

C#

```
C#
c1CheckBox1.DataSource = dt;
c1CheckBox1.DataField = "ColumnString";
c1CheckBox1.DataType = typeof(string);
// Use TranslateValues property to translate string values to/from the check box
states.
c1CheckBox1.TranslateValues.Checked = "Yes";
c1CheckBox1.TranslateValues.Unchecked = "No";
```

绑定C1CheckBox到一个数字型的字段

使用下面的代码，在编程的时候绑定C1CheckBox 到一个字符串型的字段：

Visual Basic

```
Visual Basic
c1CheckBox1.DataSource = dt
c1CheckBox1.DataField = "ColumnInt"
```

```
c1CheckBox1.DataType = GetType(Integer)
'Use TranslateValues property to translate string values to/from the check box
states.
c1CheckBox1.TranslateValues.Checked = 1
c1CheckBox1.TranslateValues.Unchecked = 0
```

C#

C#

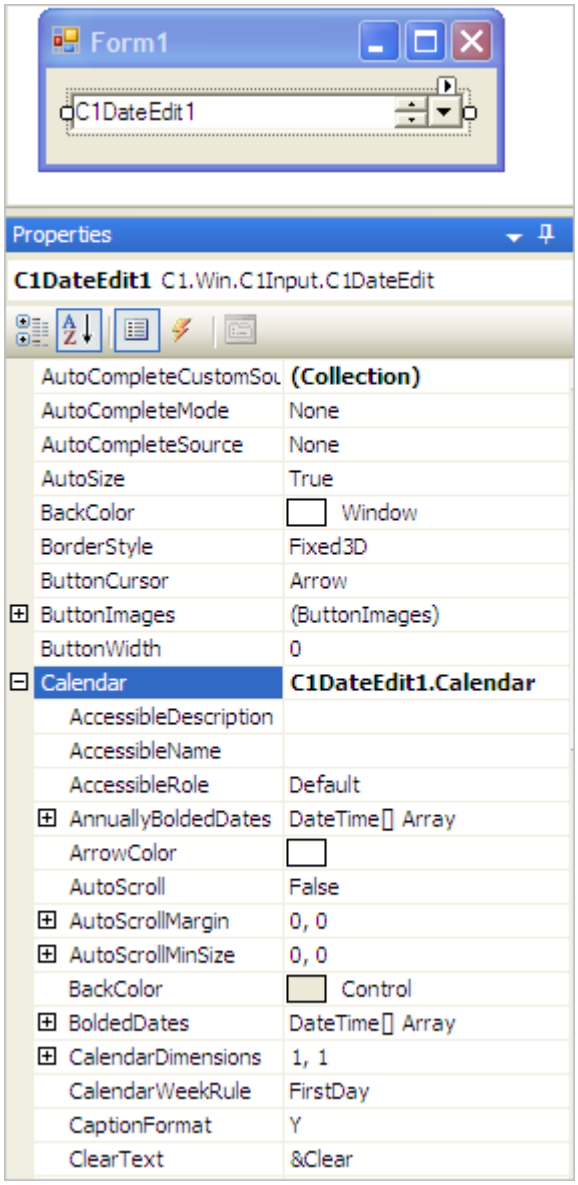
```
c1CheckBox1.DataSource = dt;
c1CheckBox1.DataField = "ColumnInt";
c1CheckBox1.DataType = typeof(int);
// Use TranslateValues property to translate string values to/from the check box
states.
c1CheckBox1.TranslateValues.Checked = 1;
c1CheckBox1.TranslateValues.Unchecked = 0;
```

设定下拉日历

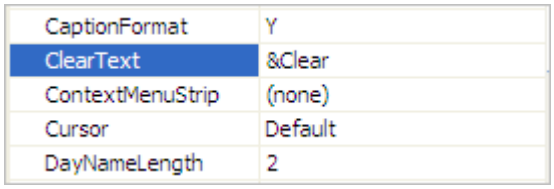
在InputforWinForms以前的版本中，C1DateEdit控件的日历特性允许你通过操作Calendar.UIString属性设定"Today"和"Clear"按钮。在C1Input.DateEdit控件新的版本中，你可以通过访问C1DateEdit属性菜单设置这些按钮。

设定"Today"和"Clear"按钮：

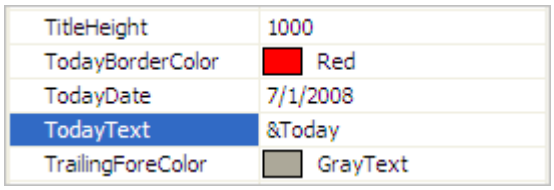
1. 添加C1DateEdit控件到你的窗体：
2. 从属性菜单选择C1.DateEdit1
3. 在左边的列中找到Calendar并且展开Calendar属性。



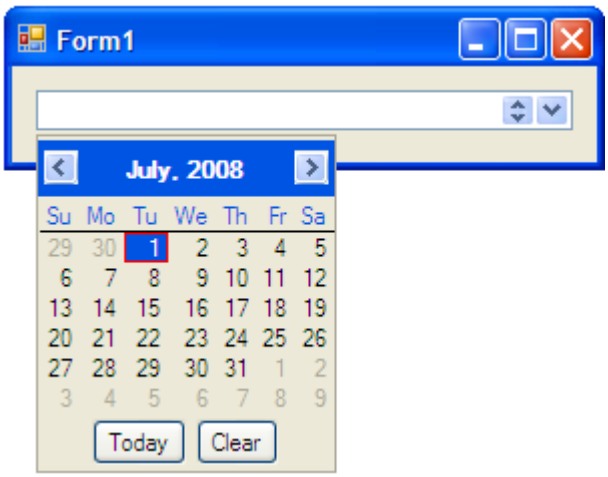
4. 在左边的列中找到ClearText并且在右边的列中输入"&Reset"。



5. 在左边的列中找到TodayText并且在右边的列中输入"&Now"。



6. 按下F5来编译和运行程序



现在当你打开C1DateEdit下拉菜单，就会出现"Today"和"Clear"按钮。

使用视觉样式定义控件外观

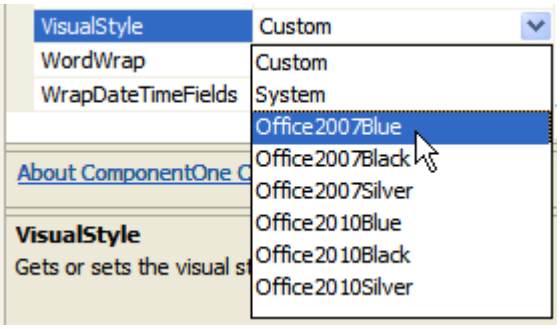
设定 C1Input 控件的 VisualStyle属性可以改变控件的背景以及边框，这些控件包括C1TextBox，C1Label，C1DbNavigator，C1DropDownControl，C1DateEdit（包括下拉的日历控件），C1NumericEdit（包括下拉的计算器控件）以及C1Button。

C1Input控件的视觉样式定义了控件的外观，可以设定的样式有：Custom，Office2007Black，Office2007Blue，Office2007Silver，System，Office2010Blue，Office2010Black，以及Office2010Silver.该属性在设计时和代码中都可以设定，下面的表格详细介绍了每一种样式：

视觉样式	描述
Custom	没有应用视觉样式，通常使用 styles 和 appearance 属性来设定外观。
Office2007Black	Office2007 黑色主题样式。
Office2007Blue	Office2007 蓝色主题样式。
Office2007Silver	Office2007 银色主题样式。
System	当前操作系统主题样式。
Office2010Blue	Office2010 蓝色主题样式。
Office2010Black	Office2010 黑色主题样式。
Office2010Silver	Office2010 银色主题样式

使用设计器更改样式

在设计时选中控件，打开属性窗口，定位到VisualStyle属性，可以设置该属性为Custom，Office2007Black，Office2007Blue，Office2007Silver，System，Office2010Blue，Office2010Black，和Office2010Silver。下面的例子中将C1TextBox 的VisualStyle属性设定为 Office2007Blue。



使用代码更改样式

在页面的Form_Load事件中设置控件的VisualStyle属性为Custom，Office2007Black，Office2007Blue，Office2007Silver，System，Office2010Blue，Office2010Black，和Office2010Silver。下面的例子中将 C1TextBox 的VisualStyle属性设定为 Office2007Blue。

Visual Basic

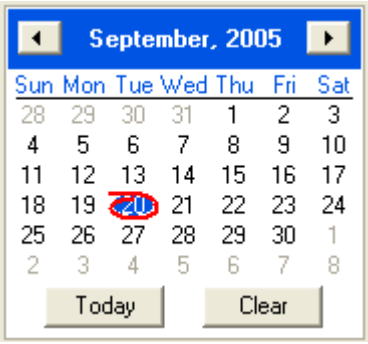
```
Visual Basic
Me.C1TextBox1.VisualStyle = C1.Win.C1Input.VisualStyle.Office2007Blue
```

C#

```
C#
Me.C1TextBox1.VisualStyle = C1.Win.C1Input.VisualStyle.Office2007Blue
```

定义C1DateEdit控件的外观

本主题介绍了如何自定义C1DateEdit控件的下拉日历的样式，默认情况下下拉日历如下图：



按钮是否可见

按照下面的步骤开发人员可以隐藏下拉日历中的 Clear 和 Today 按钮：

1. 选中 C1DateEdit控件。
2. 在属性窗口，展开 Calendar 属性节点。
3. 将ShowClearButton和ShowTodayButton属性设置为False。

日期显示格式

通过FormatType属性开发人员可以编辑 C1DateEdit 控件的显示文本。

默认情况下日期和时间都会显示，如果只希望显示日期，参考下面的步骤：

1. 选中 C1DateEdit 控件。
2. 在属性窗口中，将 FormatType 属性更改为 ShortDate。

在文本框中显示点击的 C1DropDown 控件按钮

为了能够显示点击的 C1DropDown 控件按钮，需要使用C1DropDownControl 的 UpDownButtonClick事件，在下面的例子中点击的按钮会显示在文本框中。

Visual Basic

Visual Basic

```
Private Sub C1DropDownControl1_UpDownButtonClick(ByVal sender As Object, ByVal e As
C1.Win.C1Input.UpDownButtonClickEventArgs) Handles
C1DropDownControl1.UpDownButtonClick
    If (e.Delta = 1) Then
        Me.TextBox1.AppendText("Up " & ControlChars.CrLf)
    ElseIf (e.Delta = -1) Then
        Me.TextBox1.AppendText("Down " & ControlChars.CrLf)
    End If
End Sub
```

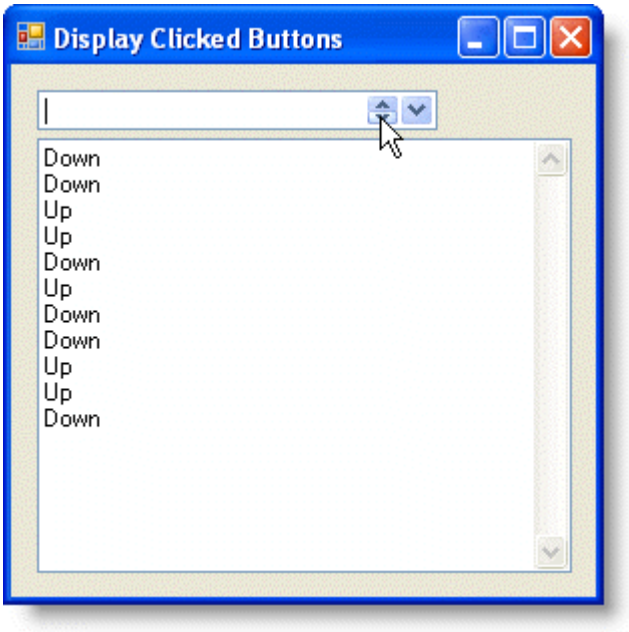
C#

C#

```
private void c1DropDownControl1_UpDownButtonClick(object sender,
C1.Win.C1Input.UpDownButtonClickEventArgs e)
{
    if ((e.Delta == 1))
    {
        this.textBox1.AppendText("Up\r\n");
    }
    else if ((e.Delta == -1))
    {
        this.textBox1.AppendText("Down\r\n");
    }
}
```

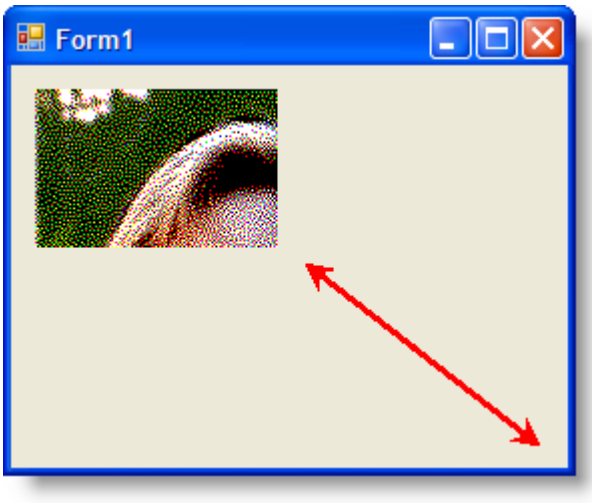
本主题介绍内容如下：

当C1DropDownControl 控件的上或者下按钮被点击的时候，文本“Up”或者“Down”会显示在文本框中，用来显示那个按钮被点击。

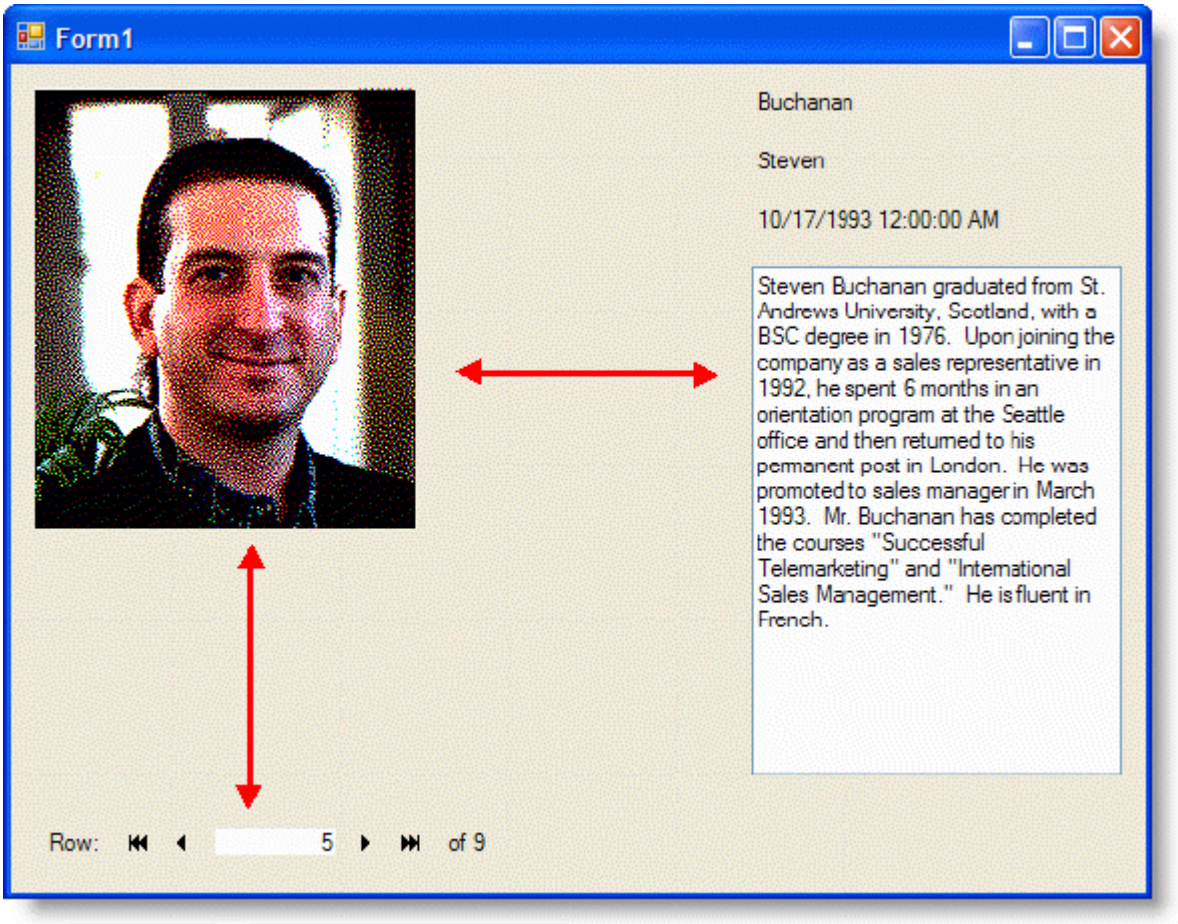


定义PictureBox控件的外观

像大多数控件一样开发人员可以使用C1.Input.PictureBox.Size 属性来更改控件的大小，但是 PictureBox中呈现的图片可能会被切去一部分或者有一些留白。



或者...



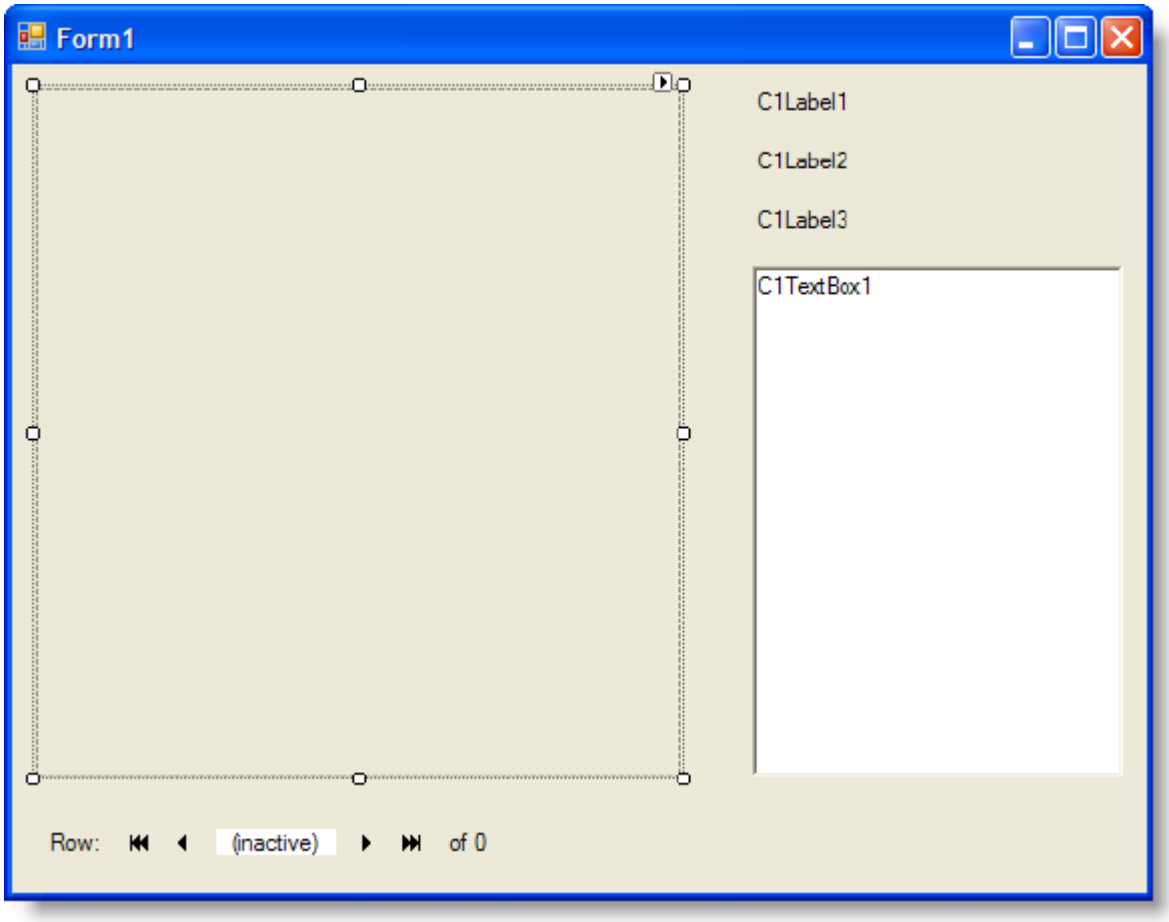
在程序运行时开发人员可能想增大图片的大小，如果你增加了 PictureBox 控件的大小，你还需要更改PictureBox的 SizeMode 属性，以便于内部图片能够被拉伸来匹配控件大小。

在PictureBox 中扩展图片

1. 创建 .NET 项目，添加下面的控件到表单上：

   - C1ExpressTable1（C1.Data.Express.C1ExpressTable）
   - C1Label1-3(C1.Win.C1Input.C1Label)
   - C1PictureBox1(C1.Win.C1Input.C1PictureBox)
   - C1TextBox1(C1.Win.C1Input.C1TextBox)
   - C1DbNavigator1(C1.Win.C1Input.C1DbNavigator)

2. 按照下图放置控件：



3. 在C1ExpressTable控件的ConnectionString
属性中输入以下字符串:

"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="C:\Program Files\ComponentOne Studio.NET 2.0\Common\Nwind.mdb"

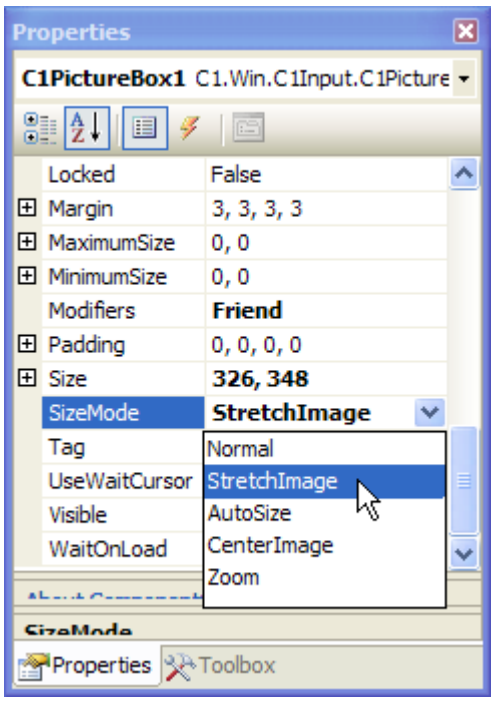
注意：步骤三假设在ComponentOne 控件的安装位置有一个数据库文件 NWind.mdb，如果你在其他位置有一个数据库文件，需要调整以下代码。

4. 使用属性窗口，设定控件的数据源属性：

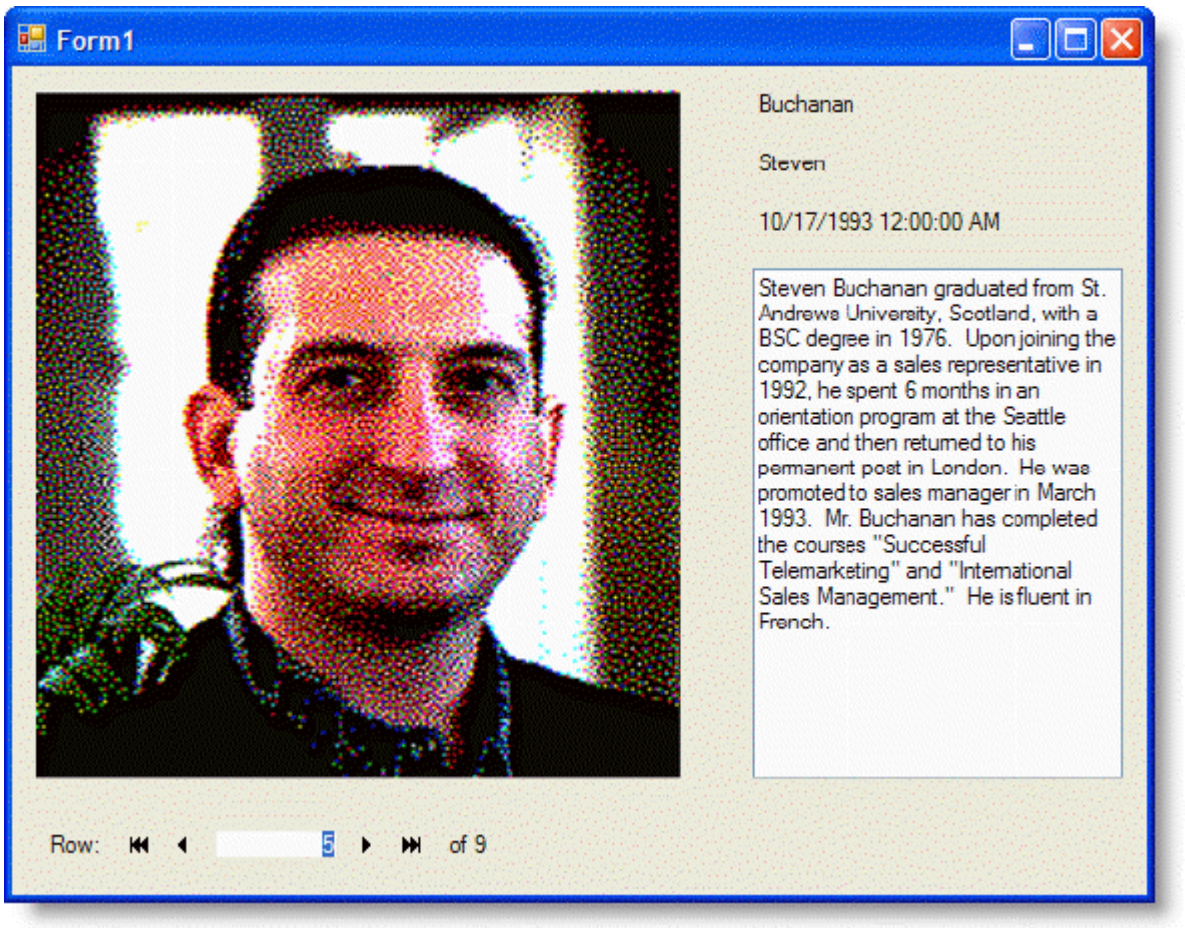
视觉样式	描述
C1DbNavigator1.DataSource	C1ExpressTable1
C1Label1.DataSource	C1ExpressTable1
C1Label1.DataField	LastName
C1Label2.DataSource	C1ExpressTable1
C1Label2.DataField	FirstName
C1Label3.DataSource	C1ExpressTable1
C1Label3.DataField	HireDate
C1PictureBox1.DataSource	C1ExpressTable1
C1PictureBox1.DataField	Photo
C1TextBox1.DataSource	C1ExpressTable1

C1TextBox1.DataField	Notes

5. 运行程序你会发现 PictureBox 被没有被图片完全覆盖，会有一大片留白，还需要调整PictureBox 属性来拉伸图片。
6. 将C1PictureBox1的SizeMode 属性从Normal调整到StretchImage。



7. 运行程序，注意到图片被拉伸至完全填充 PictureBox。



C1DateEdit控件导航

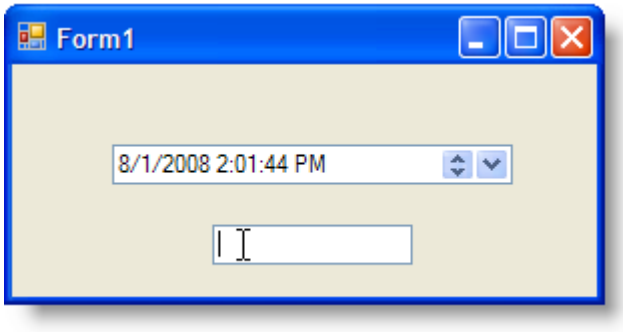
在程序运行时C1DateEdit 控件获得焦点的时候，如果在键盘上点击回车键或者Tab键，或者使用鼠标点击其他控件，C1DateEdit 控件会丢失焦点，当同时当前日期会被自动填充至控件。

如果不希望C1DateEdit 控件在丢失焦点的时候被填充，参照下面的步骤：

使用设计器：

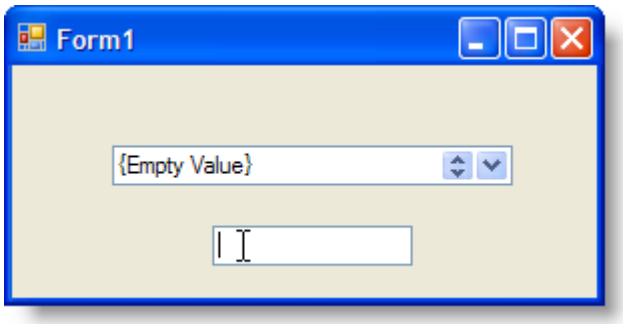
1. 创建一个新的.NET 程序，添加C1DateEdit控件和C1TextBox控件到表单上。
2. 在属性窗口设置C1DateEdit.NullText属性为“｛Empty Value｝”。

运行程序选中 C1Date Edit 控件，使用鼠标或者键盘让 C1TextBox 获得焦点，当前时间会显示在 C1DateEdit 控件中。



注意到焦点确实被移至C1TextBox 控件，但是当前日期被填充至C1DateEdit。

3. 使用属性窗口将C1DateEdit1的DateTimeInput属性更改为False，C1DateEdit1的EmptyAsNull属性更改为True.
4. 运行程序点击**C1DateEdit** 和**C1TextBox**控件。



注意到焦点被移至C1TextBox，C1DateEdit 控件将继续保留空值。

使用代码：

1. 创建一个 .NET工程，并且添加 C1Input 控件的引用。

   在代码中导入C1Input的命名空间。

   Visual Basic

```
Visual Basic
Imports C1.Win.C1Input
```

   C#

```
C#
using C1.Win.C1Input;
```

2. 在Form_Load 事件中添加C1DateEdit控件和C1TextBox 控件。

   Visual Basic

```
Visual Basic
Dim X As New C1DateEdit
Controls.Add(X)
X.Location = New Point(50, 40)
Dim Y As New C1TextBox
Controls.Add(Y)
Y.Location = New Point(100, 80)
```

   C#

```
C#
C1DateEdit X = new C1DateEdit();
Controls.Add(X);
X.Location = new Point(50, 40);
```

```
C1TextBox Y = new C1TextBox();
Controls.Add(Y);
Y.Location = new Point(100, 80);
```

3. 为了设置空值的时候控件显示的字符，添加下面的代码：

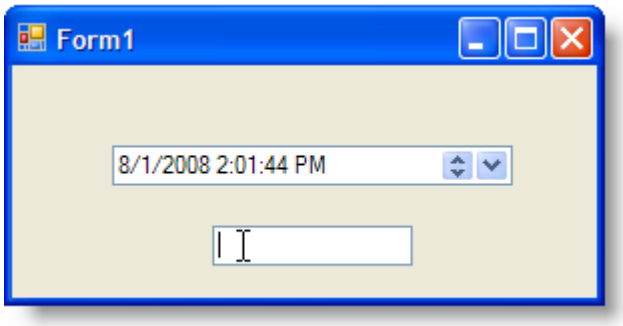
Visual Basic

```
Visual Basic
X.NullText = "{Empty Value}"
```

C#

```
C#
X.NullText = "{Empty Value}";
```

运行程序选中 C1Date Edit 控件，使用鼠标或者键盘让 C1TextBox 获得焦点，当前时间会显示在 C1DateEdit 中。



4. 如果需要在 C1Date Edit 丢失焦点的时候保留空值，使用下面的代码：

Visual Basic

```
Visual Basic
X.DateTimeInput = False
X.EmptyAsNull = False
```

C#

```
C#
X.DateTimeInput = False;
X.EmptyAsNull = False;
```

5. 运行程序点击C1TextBox控件，将焦点移至C1TextBox，C1DateEdit 控件将继续保留空值。

为C1Input控件显示带颜色边框

边框的颜色可以应用到C1DateEdit，C1Label，C1NumericEdit，C1DropDownControl，以及C1TextBox控件上。

在设计时为C1DropDownControl创建带颜色的边框

1. 添加一个C1DropDownControl至窗体。
2. 导航到C1DropDownControl的属性窗体，并设置BorderStyle属性为"FixedSingle"。
3. 设置C1DropDownControl的BorderColor属性的值为"Red"。

当边框颜色变化时显示一个消息框

使用以下代码为C1DropDownControl添加一个带有颜色的边框：

Visual Basic

Visual Basic

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        Dim dropdown As New C1DropDownControl
        Me.Controls.Add(dropdown)
        dropdown.BorderStyle = BorderStyle.FixedSingle
        dropdown.BorderColor = Color.Red
    End Sub
```

C#

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
            C1DropDownControl dropdown = new C1DropDownControl();
            this.Controls.Add(dropdown);
            dropdown.BorderStyle = BorderStyle.FixedSingle;
            dropdown.BorderColor = Color.Red;
        }
```

本主题说明如下：

FixedSingle样式的红色边框将添加到C1DropDownControl。

当边框颜色变化时显示一个消息框

您可以使用BorderColorChanged事件，该事件在BorderColor属性的值发生变化时触发。

为了在C1TextBox边框颜色发生变化时创建一个消息框，请完成以下步骤：

1. 向您的窗体添加一个C1TextBox。
2. 导航至C1TextBox的属性窗体，并修改BorderStyle属性为"FixedSingle"。
3. 向C1TextBox控件添加对MouseClick事件的处理，以改变C1TextBox的边框颜色为紫色。

Visual Basic

```
Visual Basic
Private Sub C1TextBox1_MouseClick(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles C1TextBox1.MouseClick
    C1TextBox1.BorderColor = Color.Purple
End Sub
```

C#

```
C#
private void c1TextBox1_MouseClick(object sender, MouseEventArgs e)
        {

            c1TextBox1.BorderColor = Color.Purple;
        }
```

4. 向C1TextBox1添加BorderColorChanged事件处理，以显示一个消息框提示用户边框的颜色发生了变化。

Visual Basic

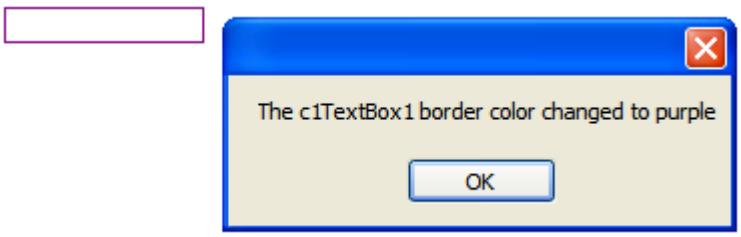
```
Visual Basic
Private Sub C1TextBox1_BorderColorChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles C1TextBox1.BorderColorChanged
   MessageBox.Show("The C1TextBox1 border color change to purple")
End Sub
```

C#

```
C#
private void c1TextBox1_BorderColorChanged(object sender, EventArgs e)
        {
            MessageBox.Show("The c1TextBox1 border color changed to purple");
        }
```

本主题说明如下：

当您鼠标单击C1TextBox控件时，边框颜色将变为紫色。一旦该颜色变为紫色，将触发BorderColorChanged，此时将出现一个消息框，提示用户边框的颜色发生了变化。



设置输入法模式

ImeMode属性可以被用作设置C1Input控件的输入法编辑器（IME）模式。输入法编辑器是一个程序，允许用户输入复杂的字符或符号，如日文汉字字符，至输入控件，使用一个基础的键盘。

下表介绍ImeMode属性可用的属性值。

值	描述
On	表示IME为开启状态。中文或日文特定的字符或符号可以输入。仅针对日文，简体中文以及繁体中文有效。
Off	表示IME为关闭状态。该对象和英文输入模式行为一致。仅针对日文，简体中文以及繁体中文有效。
Disable	表示输入法是禁用的。这意味着IME窗口被隐藏，并且用户无法通过键盘切换IME为打开状态。
Hiragana	平假名双字节字符。仅针对日文输入法有效。
Katakana	片假名双字节字符。仅针对日文输入法有效。
KatakanaHalf	单字节片假名字符。仅针对日文输入法有效。
AlphaFull	双字节字符的字母和数字。仅针对韩文和日文输入法有效。
Alpha	单字节字符的字母和数字。仅针对韩文和日文输入法有效。
HangulFull	韩文双字节字符。只针对韩文输入法有效。
NoControl	None （默认值）。
Inherit	表示输入法模式继承自父控件。
Close	表示IME为关闭状态。只适用于中文输入法。
Hangul	韩文单字节字符。只适用于韩文输入法。
OnHalf	表示输入法为HalfShape。只适用于中文输入法。

完成以下步骤，以更改C1Input控件的输入法模式：

1. 创建一个新的Windows应用程序工程。放置一个C1Input控件（C1TextBox，C1ComboBox，C1DateEdit，C1DropDownControl 或者 C1NumericEdit）至窗体。
2. 从属性窗体，设置ImeMode属性为您需要的值。

移动焦点

该功能使得在不同控件之间导航更容易，同时允许您向C1Input（C1TextBox，C1ComboBox，C1DateEdit，C1DropDownControl 以及 C1NumericEdit）控件添加键盘导航支持。以下属性允许您移动焦点至C1Input控件或者从C1Input控件移出焦点。

- ExitOnLastChar:当输入的文本长度达到最大长度限制，如MaxLength属性定义的值，或者当输入掩码填充完毕时，从C1Input控件移出焦点。其默认值为False。
- ExitOnLeftRightKey:当左或者右方向键按下时，将焦点从C1Input控件移出。其默认值是None。
- TabStop:指示当焦点在C1Input控件的上一个控件时，按下TAB键是否允许接收焦点至C1Input控件。其默认值是True。

下表描述了上述属性可用的选项值和行为。

属性	可能的值	描述
ExitOnLastChar	**True**	当输入的文本长度达到使用MaxLength定义的最大长度显示，或者当输入掩码填充完毕时，允许从C1Input控件

		移动焦点至下一个控件。
	False	当输入的文本长度达到使用MaxLength定义的最大长度显示，或者当输入掩码填充完毕时，不会自动地从C1Input控件移动焦点至下一个控件。
ExitOnLeftRightKey	**None**	当方向键按下时，不会从C1Input控件移动焦点到上一个或下一个控件。
	Left	当左方向键被按下时，允许位于C1Input控件左侧的焦点移动到C1Input控件的上一个控件。
	Right	当右方向键被按下时，允许位于C1Input控件右侧的焦点移动到C1Input控件的下一个控件。
	Both	当相关的按键被按下时，允许焦点从C1Input的最左侧或者最右侧移动到上一个或者下一个控件。
TabStop	**True**	当焦点在C1Input控件的上一个控件时，按下TAB键允许接收焦点至C1Input控件。
	False	当焦点在C1Input控件的上一个控件时，按下TAB键允许禁止焦点至C1Input控件。

完成以下步骤来启用或禁用C1Input控件的功能：

1. 创建一个新的Windows应用程序工程。放置一个C1Input控件（C1TextBox，C1ComboBox，C1DateEdit，C1DropDownControl 或者 C1NumericEdit）至窗体。
2. 从属性窗体中，设置一个或全部的以下属性为您期望的设置。

   - 设置ExitOnLastChar为True或False，启用或禁用当输入的文本达到最大长度限制时，从控件移出焦点。
   - ExitOnLeftRightKey为Left，Right或者Both，以启用当按下相应的方向按键时，从控件移出焦点。
   - 设置TabStop属性为True或False，以启用或禁用当Tab键按下时，是否移动焦点到C1Input控件上。

选择特定的日历类型

位于C1DateEdit以及C1NumericEdit 控件上的CalendarType属性，允许您选择特定的非默认日历。C1DateEdit 以及C1NumericEdit 支持以下日历类型：

- Default
- ChineseLuniSolarCalendar
- EastAsianLunisolarCalendar
- GregorianCalendar
- HebrewCalendar
- HijriCalendar
- JapaneseCalendar
- JapaneseLunisolarCalendar
- JulianCalendar
- KoreanCalendar
- KoreanLunisolarCalendar
- TaiwanCalendar
- TaiwanLunisolarCalendar
- ThaiBuddhistCalendar
- UmAlQuraCalendar

完成以下步骤以修改C1Input控件的日历类型：

1. 创建一个新的Windows应用程序工程。在窗体上放置一个C1Input控件（C1DateEdit或者C1NumericEdit）。
2. 从属性窗口设置CalendarType属性为您期望的值。

以编程方式执行Spin Up/Spin Down

当焦点没有位于控件上时，SpinUp以及SpinDown方法允许您增加或减少C1Input控件输入的值（C1ComboBox，C1DropDownControl，C1DateEdit 以及 C1NumericInput）。

除了上述方法之外，C1ComboBox 和 C1DateEdit 也包含额外的AllowSpinLoop属性，当其设置为true时，调用SpinUp或SpinDown方法可以在所有项目之间循环滚动。

以下是一个使用SpinUp方法的示例：

1. 创建一个新的Windows应用程序工程。向窗体放置一个C1NumericEdit 控件。
2. 从属性窗口中，设置Value属性的值为0。
3. 添加一个文本框控件至窗体，并双击该文本框以便在代码中生成TextChanged事件。
4. 将以下代码添加到文本框的TextChanged事件。

Visual Basic

Visual Basic

```
Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles
TextBox1.TextChanged
      c1NumericEdit1.SpinUp(1)
End Sub
```

C#

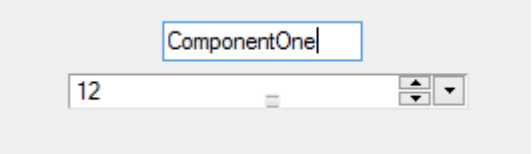
C#

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
      c1NumericEdit1.SpinUp(1);
}
```

5. 运行该工程。

您完成了什么

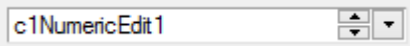
当您在文本框控件中键入字符时，C1NumericEdit控件的值将增加1，每次输入一个新的字符，将返回控件中全部输入的字符个数。



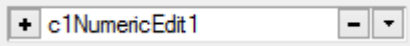
改变Up-Down按钮的对齐方式

UpDownButtonAlignment属性允许您改变存在于C1DropDown，C1DateEdit 以及 C1NumericEdit一侧的Up和Down按钮的对齐方式。该属性可以有以下值：

- **Default**: 这两个按钮放在右边，紧挨着下拉按钮。



- **UpLeftDownRight**: Up按钮放在左边，Down按钮放在右边。



- **UpRightDownLeft**: Up按钮放在右边，Down按钮放在左边。

完成以下步骤来改变Up和Down按钮的对齐方式：

1. 创建一个新的Windows应用程序工程。向窗体放置一个C1Input控件（C1DropDown，C1DateEdit 或者C1NumericEdit）。
2. 从属性窗口中，按照您的需求设置UpDownButtonAlignment属性的值。