
ComponentOne

SuperTooltip for WinForms

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

Web site: <http://www.componentone.com>

Sales

E-mail: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of ComponentOne LLC. All other trademarks used herein are the properties of their respective owners.

Warranty

ComponentOne warrants that the original CD (or diskettes) are free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective CD (or disk) to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. We are not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

This manual was produced using [ComponentOne Doc-To-Help™](#).

Table of Contents

ComponentOne SuperTooltip for WinForms Overview	1
Installing SuperTooltip for WinForms	1
SuperTooltip for WinForms Setup Files	1
System Requirements	2
Installing Demonstration Versions	2
Uninstalling SuperTooltip for WinForms	2
End-User License Agreement	2
Licensing FAQs	3
What is Licensing?	3
How does Licensing Work?	3
Common Scenarios	4
Troubleshooting	6
Technical Support	7
Redistributable Files	8
About This Documentation	8
Namespaces	9
Creating a .NET Project	10
Adding the SuperTooltip for WinForms Components to a Project	10
Key Features	11
SuperTooltip for WinForms Quick Start	13
Step 1 of 4: Adding a C1SuperLabel to a Project	13
Step 2 of 4: Creating a Vista-style C1SuperTooltip for a Control	15
Step 3 of 4: Creating a C1SuperTooltip with HTML Code	17
Step 4 of 4: Adding Code for the Buttons and Running the Project	18
SuperTooltip for WinForms Top Tips	21
Design-Time Support	23
C1SuperTooltip Menus	23
C1SuperTooltip Tasks and Context Menus	23
C1SuperLabel Tasks and Context Menus	24
C1SuperErrorProvider Tasks and Context Menus	24

C1SuperTooltip Editors	25
C1SuperTooltip Editor.....	25
C1SuperLabel Editor.....	29
Edit Image Collection Editor.....	30
C1SuperTooltip Elements	30
ToolTip Elements	31
Label Elements.....	32
ErrorProvider Elements.....	32
C1SuperTooltip Appearance	32
C1SuperTooltip Background Gradient	32
C1SuperTooltip Shape	33
C1SuperTooltip Shadow	34
SuperTooltip for WinForms Samples	35
SuperTooltip for WinForms Task-Based Help	35
Creating C1SuperTooltips.....	36
Creating C1SuperTooltips at Design Time.....	36
Creating a C1SuperTooltip Programmatically.....	37
Creating a C1SuperTooltip using a Cascading Style Sheet	39
Adding a C1SuperTootip using HTML.....	41
Adding Multiple C1SuperTooltips.....	42
Changing the C1SuperTooltip Appearance and Behavior Settings	43
Adding an Image to C1SuperTooltip.....	45
Creating C1SuperLabels.....	45
Creating C1SuperLabels at Design Time	46
Creating a C1SuperLabel Programmatically.....	46
Adding an Image to C1SuperLabel.....	47
Creating C1SuperErrorProvider Error Messages	48
Creating an Error Message	48
Changing the Error Message Icon.....	49
Changing the Error Message Blink Style	49
Showing an Image when the Error Icon is Hovered.....	50
Using C1SuperErrorProvider with Data Sources	50

ComponentOne SuperTooltip for WinForms Overview

ComponentOne SuperTooltip™ for WinForms allows you to create visually rich WinForms applications with Vista-style ToolTips and labels that can display HTML content. **SuperTooltip for WinForms** loads and displays HTML content much faster than its standard counterparts and without the security concerns often associated with the WebBrowser control since it doesn't depend on Internet Explorer.

SuperTooltip for WinForms supports virtually all HTML constructs, including cascading style sheets, mixed fonts and text colors, preformatted text, tables, bulleted and numbered lists, and more. With **SuperTooltip for WinForms**, the possibilities for your customized ToolTips and labels are endless.

For a list of the latest features added to **ComponentOne Studio for WinForms**, visit [What's New in Studio for WinForms](#).

Installing SuperTooltip for WinForms

The following sections provide helpful information on installing **SuperTooltip for WinForms**.

SuperTooltip for WinForms Setup Files

The installation program will create the following directory: C:\Program Files\ComponentOne\Studio for WinForms. This directory contains the following subdirectories:

bin	Contains copies of all ComponentOne binaries (DLLs, EXEs).
C1SuperTooltip	Contains files (at least a readme.txt) related to the SuperTooltip for WinForms product.

The **ComponentOne Studio for WinForms Help Setup** program installs integrated Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for WinForms\HelpViewer folder.

Samples

Samples for the product are installed in the **ComponentOne Samples** folder by default. The path of the **ComponentOne Samples** directory is slightly different on Windows XP and Windows 7/Vista machines:

Windows XP path: C:\Documents and Settings\\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\\Documents\ComponentOne Samples

The **ComponentOne Samples** folder contains the following subdirectories:

Common	Contains support and data files that are used by many of the demo programs.
C1SuperTooltip	Contains samples and tutorials for SuperTooltip for WinForms .

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **ComponentOne | Studio for WinForms | Samples | SuperTooltip Samples**.

System Requirements

System requirements include the following:

Operating Systems:	Windows 2000 Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista Windows 7
Environment:	.NET Framework 2.0 or later C# .NET Visual Basic .NET
Disc Drive:	CD or DVD-ROM drive if installing from CD

Installing Demonstration Versions

If you wish to try **SuperTooltip for WinForms** and do not have a serial number, follow the steps through the installation wizard and use the default serial number.

The only difference between unregistered (demonstration) and registered (purchased) versions of our products is that registered versions will stamp every application you compile so a ComponentOne banner will not appear when your users run the applications.

Uninstalling SuperTooltip for WinForms

To uninstall **SuperTooltip for WinForms**:

1. Open the **Control Panel** and select **Add or Remove Programs (Programs and Features)** in Windows 7/Vista).
2. Select **ComponentOne Studio for WinForms** and click the **Remove** button.
3. Click **Yes** to remove the program.

To uninstall **ComponentOne SuperTooltip for WinForms** integrated help:

1. Open the Control Panel and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
2. Select **ComponentOne Studio for WinForms Help** and click the **Remove** button.
3. Click **Yes** to remove the integrated help.

End-User License Agreement

All of the ComponentOne licensing information, including the ComponentOne end-user license agreements, frequently asked licensing questions, and the ComponentOne licensing model, is available online at <http://www.componentone.com/SuperPages/Licensing/>.

Licensing FAQs

This section describes the main technical aspects of licensing. It may help the user to understand and resolve licensing problems he may experience when using ComponentOne WinForms and ASP.NET products.

What is Licensing?

Licensing is a mechanism used to protect intellectual property by ensuring that users are authorized to use software products.

Licensing is not only used to prevent illegal distribution of software products. Many software vendors, including ComponentOne, use licensing to allow potential users to test products before they decide to purchase them.

Without licensing, this type of distribution would not be practical for the vendor or convenient for the user. Vendors would either have to distribute evaluation software with limited functionality, or shift the burden of managing software licenses to customers, who could easily forget that the software being used is an evaluation version and has not been purchased.

How does Licensing Work?

ComponentOne uses a licensing model based on the standard set by Microsoft, which works with all types of components.

Note: The **Compact Framework** components use a slightly different mechanism for run-time licensing than the other ComponentOne components due to platform differences.

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog.)

When a licensed component is added to a form or Web page, Visual Studio obtains version and licensing information from the newly created component. When queried by Visual Studio, the component looks for licensing information stored in the system and generates a run-time license and version information, which Visual Studio saves in the following two files:

- An assembly resource file which contains the actual run-time license
- A "licenses.licx" file that contains the licensed component strong name and version information

These files are automatically added to the project.

In WinForms and ASP.NET 1.x applications, the run-time license is stored as an embedded resource in the assembly hosting the component or control by Visual Studio. In ASP.NET 2.x applications, the run-time license may also be stored as an embedded resource in the App_Licenses.dll assembly, which is used to store all run-time licenses for all components directly hosted by WebForms in the application. Thus, the App_licenses.dll must always be deployed with the application.

The licenses.licx file is a simple text file that contains strong names and version information for each of the licensed components used in the application. Whenever Visual Studio is called upon to rebuild the application resources, this file is read and used as a list of components to query for run-time licenses to be embedded in the appropriate assembly resource. Note that editing or adding an appropriate line to this file can force Visual Studio to add run-time licenses of other controls as well.

Note that the licenses.licx file is usually not shown in the Solution Explorer; it appears if you press the **Show All Files** button in the Solution Explorer's Toolbox, or from Visual Studio's main menu, select **Show All Files** on the **Project** menu.

Later, when the component is created at run time, it obtains the run-time license from the appropriate assembly resource that was created at design time and can decide whether to simply accept the run-time license, to throw an

exception and fail altogether, or to display some information reminding the user that the software has not been licensed.

All ComponentOne products are designed to display licensing information if the product is not licensed. None will throw licensing exceptions and prevent applications from running.

Common Scenarios

The following topics describe some of the licensing scenarios you may encounter.

Creating components at design time

This is the most common scenario and also the simplest: the user adds one or more controls to the form, the licensing information is stored in the licenses.licx file, and the component works.

Note that the mechanism is exactly the same for Windows Forms and Web Forms (ASP.NET) projects.

Creating components at run time

This is also a fairly common scenario. You do not need an instance of the component on the form, but would like to create one or more instances at run time.

In this case, the project will not contain a licenses.licx file (or the file will not contain an appropriate run-time license for the component) and therefore licensing will fail.

To fix this problem, add an instance of the component to a form in the project. This will create the licenses.licx file and things will then work as expected. (The component can be removed from the form after the licenses.licx file has been created).

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the licenses.licx file. If desired, you can do this manually using notepad or Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

Inheriting from licensed components

If a component that inherits from a licensed component is created, the licensing information to be stored in the form is still needed. This can be done in two ways:

- Add a LicenseProvider attribute to the component.

This will mark the derived component class as licensed. When the component is added to a form, Visual Studio will create and manage the licenses.licx file, and the base class will handle the licensing process as usual. No additional work is needed. For example:

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

- Add an instance of the base component to the form.

This will embed the licensing information into the licenses.licx file as in the previous scenario, and the base component will find it and use it. As before, the extra instance can be deleted after the licenses.licx file has been created.

Please note, that C1 licensing will not accept a run time license for a derived control if the run time license is embedded in the same assembly as the derived class definition, and the assembly is a DLL. This restriction is necessary to prevent a derived control class assembly from being used in other applications without a design time license. If you create such an assembly, you will need to take one of the actions previously described create a component at run time.

Using licensed components in console applications

When building console applications, there are no forms to add components to, and therefore Visual Studio won't create a licenses.licx file.

In these cases, create a temporary Windows Forms application and add all the desired licensed components to a form. Then close the Windows Forms application and copy the licenses.licx file into the console application project.

Make sure the licenses.licx file is configured as an embedded resource. To do this, right-click the licenses.licx file in the Solution Explorer window and select **Properties**. In the property window, set the **Build Action** property to **Embedded Resource**.

Using licensed components in Visual C++ applications

There is an issue in VC++ 2003 where the licenses.licx is ignored during the build process; therefore, the licensing information is not included in VC++ applications.

To fix this problem, extra steps must be taken to compile the licensing resources and link them to the project. Note the following:

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:
`c1lc /target:MyApp.exe /complist:licenses.licx /i:C1.Win.C1FlexGrid.dll`
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select Properties, and go to the Linker/Command Line option. Enter the following:
`/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses`
6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

Automated testing products that load assemblies dynamically may cause them to display license dialogs. This is the expected behavior since the test application typically does not contain the necessary licensing information, and there is no easy way to add it.

This can be avoided by adding the string "C1CheckForDesignLicenseAtRuntime" to the AssemblyConfiguration attribute of the assembly that contains or derives from ComponentOne controls. This attribute value directs the ComponentOne controls to use design time licenses at run time.

For example:

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

Note that the AssemblyConfiguration string may contain additional text before or after the given string, so the AssemblyConfiguration attribute can be used for other purposes as well. For example:

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

THIS METHOD SHOULD ONLY BE USED UNDER THE SCENARIO DESCRIBED. It requires a design time license to be installed on the testing machine. Distributing or installing the license on other computers is a violation of the EULA.

Troubleshooting

We try very hard to make the licensing mechanism as unobtrusive as possible, but problems may occur for a number of reasons.

Below is a description of the most common problems and their solutions.

I have a licensed version of a ComponentOne product but I still get the splash screen when I run my project.

If this happens, there may be a problem with the licenses.licx file in the project. It either doesn't exist, contains wrong information, or is not configured correctly.

First, try a full rebuild (**Rebuild All** from the Visual Studio **Build** menu). This will usually rebuild the correct licensing resources.

If that fails follow these steps:

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 1: Follow these steps:

1. Open a new Visual Studio.NET project.
2. Add the updated component to the form.
3. Compile and run the new project.
4. Open the licenses.licx file in the new project.
5. Copy the line that starts with the namespace of the updated component (for example, C1.Win.C1Report) and ends with a public key token.
6. Open the existing, incorrectly licensed project.
7. Open the licenses.licx file in the new project.
8. Paste the line from step 5 into this file (replace the old licensing information with the new).
9. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 2: Follow these steps:

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a

website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

I have a licensed version of a ComponentOne product on my Web server but the components still behave as unlicensed.

There is no need to install any licenses on machines used as servers and not used for development.

The components must be licensed on the development machine, therefore the licensing information will be saved into the executable (.exe or .dll) when the project is built. After that, the application can be deployed on any machine, including Web servers.

For ASP.NET 2.x applications, be sure that the App_Licenses.dll assembly created during development of the application is deployed to the bin application bin directory on the Web server.

If your ASP.NET application uses WinForms user controls with constituent licensed controls, the run-time license is embedded in the WinForms user control assembly. In this case, you must be sure to rebuild and update the user control whenever the licensed embedded controls are updated.

I downloaded a new build of a component that I have purchased, and now I'm getting the splash screen when I build my projects.

Make sure that the serial number is still valid. If you licensed the component over a year ago, your subscription may have expired. In this case, you have two options:

Option 1 – Renew your subscription to get a new serial number.

If you choose this option, you will receive a new serial number that you can use to license the new components (from the installation utility or directly from the **About Box**).

The new subscription will entitle you to a full year of upgrades and to download the latest maintenance builds directly from <http://prerelease.componentone.com/>.

Option 2 – Continue to use the components you have.

Subscriptions expire, products do not. You can continue to use the components you received or downloaded while your subscription was valid.

Technical Support

ComponentOne offers various support options. For a complete list and a description of each, visit the ComponentOne Web site at <http://www.componentone.com/SuperProducts/SupportServices/>.

Some methods for obtaining technical support include:

- **[Online Resources](#)**
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Product Forums**
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and

technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.

- **Installation Issues**

Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.

- **Documentation**

Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

Note: You must create a ComponentOne Account and register your product with a valid serial number to obtain support using some of the above methods.

Redistributable Files

ComponentOne SuperTooltip for WinForms is developed and published by ComponentOne LLC. You may use it to develop applications in conjunction with Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s). You may also distribute, free of royalties, the following Redistributable Files with any such application you develop to the extent that they are used separately on a single CPU on the client/workstation side of the network:

- C1.Win.C1SuperTooltip.2.dll
- C1.Win.C1SuperTooltip.4.dll
- C1.Win.C1SuperTooltip.4.Design.dll

Site licenses are available for groups of multiple developers. Please contact Sales@ComponentOne.com for details.

About This Documentation

Acknowledgements

Microsoft, Windows, Windows Vista, Windows Server, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

ComponentOne

If you have any suggestions or ideas for new features or controls, please call us or write:

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue

3rd Floor

Pittsburgh, PA 15206 • USA

412.681.4343

412.681.4384 (Fax)

<http://www.componentone.com/>

ComponentOne Doc-To-Help

This documentation was produced using [ComponentOne Doc-To-Help® Enterprise](#).

Namespaces

Namespaces organize the objects defined in an assembly. Assemblies can contain multiple namespaces, which can in turn contain other namespaces. Namespaces prevent ambiguity and simplify references when using large groups of objects such as class libraries.

The namespace for the **SuperTooltip for WinForms** components is **C1.Win.C1SuperTooltip**. The following code fragment shows how to declare a C1SuperTooltip component, for example, using the fully qualified name for this class:

- Visual Basic

```
Dim tooltip1 As C1.Win.C1SuperTooltip.C1SuperTooltip
```
- C#

```
C1.Win. C1SuperTooltip.C1SuperTooltip tooltip1;
```

Namespaces address a problem sometimes known as *namespace pollution*, in which the developer of a class library is hampered by the use of similar names in another library. These conflicts with existing components are sometimes called *name collisions*.

For example, if you create a new class named **C1SuperTooltip**, you can use it inside your project without qualification. However, the **C1SuperTooltip** assembly also implements a class called **C1SuperTooltip**. So, if you want to use the **C1SuperTooltip** class in the same project, you must use a fully qualified reference to make the reference unique. If the reference is not unique, Visual Studio .NET produces an error stating that the name is ambiguous. The following code snippet demonstrates how to declare these objects:

- Visual Basic

```
' Define a new C1SuperTooltip object (custom C1SuperTooltip class)  
Dim Mytooltip as C1SuperTooltip  
' Define a new C1SuperTooltip.C1SuperTooltip object.  
Dim C1ToolTip as C1.Win.C1SuperTooltip.C1SuperTooltip
```
- C#

```
// Define a new C1SuperTooltip object (custom C1SuperTooltip class)  
C1SuperTooltip Mytooltip;  
// Define a new C1SuperTooltip.C1SuperTooltip object.  
C1.Win.C1SuperTooltip.C1SuperTooltip C1ToolTip;
```

Fully qualified names are object references that are prefixed with the name of the namespace where the object is defined. You can use objects defined in other projects if you create a reference to the class (by choosing Add Reference from the Project menu) and then use the fully qualified name for the object in your code.

Fully qualified names prevent naming conflicts because the compiler can always determine which object is being used. However, the names themselves can get long and cumbersome. To get around this, you can use the **Imports** statement (**using** in C#) to define an alias — an abbreviated name you can use in place of a fully qualified name. For example, the following code snippet creates aliases for two fully qualified names, and uses these aliases to define two objects:

- Visual Basic

```
Imports C1ToolTip = C1.Win.C1SuperTooltip.C1SuperTooltip  
Imports Mytooltip = MyProject.C1SuperTooltip  
  
Dim t1 As C1ToolTip  
Dim t2 As Mytooltip
```
- C#

```
using C1ToolTip = C1.Win.C1SuperTooltip.C1SuperTooltip;  
using Mytooltip = MyProject.C1SuperTooltip;  
  
C1ToolTip t1;  
Mytooltip t2;
```

If you use the **Imports** statement without an alias, you can use all the names in that namespace without qualification, provided they are unique to the project.

Creating a .NET Project

To create a new .NET project, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic Projects** or **Visual C# Projects**. Note that one of these options may be located under **Other Languages**.
3. Select **Windows Application** from the list of **Templates** in the right pane.
4. Enter or browse for a location for your application in the **Location** field and click **OK**. A new Microsoft Visual Studio .NET project is created in the specified location. In addition, a new Form1 is displayed in the Designer view.
5. Double-click the C1SuperLabel, C1SuperTooltip, or C1SuperErrorProvider component from the Toolbox to add it to Form1. For information on adding a component to the toolbox, see [Adding the SuperTooltip for WinForms Components to a Project](#) (page 10).

Adding the SuperTooltip for WinForms Components to a Project

When you install ComponentOne Studio for WinForms, the **Create a ComponentOne Visual Studio 2005/2008 Toolbox Tab** checkbox is checked, by default, in the installation wizard. When you open Visual Studio, you will notice a **ComponentOne Studio for .NET 2.0** tab containing the ComponentOne controls has automatically been added to the Toolbox.

If you decide to uncheck the **Create a ComponentOne Visual Studio 2005/2008 Toolbox Tab** checkbox during installation, you can manually add ComponentOne controls to the Toolbox at a later time.

ComponentOne **SuperTooltip for WinForms** provides the following controls:

- C1SuperLabel
- C1SuperTooltip
- C1SuperErrorProvider

To use **SuperTooltip for WinForms**, add these controls to the form or add a reference to the C1.Win.C1SuperTooltip.2 assembly to your project.

Manually Adding SuperTooltip for WinForms Controls to the Toolbox

To add the components to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.
2. To make **SuperTooltip for WinForms** components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and enter the tab name, **C1SuperTooltip**, for example.
3. Right-click the tab where the components are to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the **Namespace** column header) and check the check boxes for all components belonging to namespace C1.Win.C1SuperTooltip. Note that there may be more than one component for each namespace.

Adding SuperTooltip for WinForms Components to the Form

To add one of the components to a form:

1. Add it to the Visual Studio Toolbox.
2. Double-click the control or drag it onto your form.

Adding a Reference to the Assembly

To add a reference to the C1.Win.C1SuperTooltip.2 assembly:

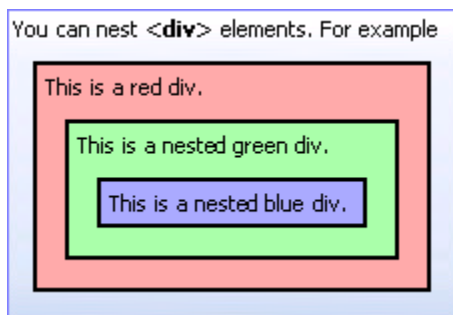
1. Select the **Add Reference** option from the **Project** menu of your project.
2. Select the **ComponentOne C1SuperTooltip** assembly from the list on the **.NET** tab or browse to find the C1.Win.C1SuperTooltip.2.dll file and click **OK**.
3. Double-click the form caption area to open the code window. At the top of the file, add the following **Imports** statements (**using** in C#):

```
Imports C1.Win.C1SuperTooltip
```

Key Features

You can use **ComponentOne SuperTooltip for WinForms** to create the following items within the C1SuperTooltip, C1SuperLabel, and C1SuperErrorProvider components in your application:

- **Nested <div> elements**



- **Mixed fonts and text colors using tags**

Use spans to mix different fonts in a paragraph and change text colors. **For example, this is 10pt Times New Roman, and it is red. And this is 12pt Verdana, and it is green.**

- **Preformatted text using the <pre> tags**

Html tooltips are great for showing quick samples. You can format the code with <pre> tags and give it a background and indentation:

```
string tipText = "Hello <b>World</b>";  
C1SuperTooltip.SetTooltip(control, tipText);
```

That would create a tooltip with a bold word in it.

- Paragraph alignment

This is a paragraph that is aligned to the **left**. This is a paragraph that is aligned to the **left**. This is a paragraph that is aligned to the **left**.

This is a paragraph that is aligned to the **center**. This is a paragraph that is aligned to the **center**. This is a paragraph that is aligned to the **center**.

This is a paragraph that is aligned to the **right**. This is a paragraph that is aligned to the **right**. This is a paragraph that is aligned to the **right**.

This is a paragraph that is **justified**. This is a paragraph that is **justified**. This is a paragraph that is **justified**.

- Formatted tables, nested tables and spanning rows and columns

Tables

Here's a table with some sales numbers:

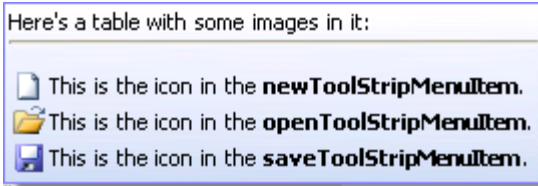
This cell spans three rows.	Semester 1					
	Quarter 1			Quarter 2		
	Jan	Feb	Mar	Apr	May	Jun
Widgets	12	23	23	43	23	34
Sprockets	45	32	23	23	34	43
Gaskets	45	55	53	34	45	78
Doodads	23	3	3	11	23	3

- Nested, bulleted and numbered lists

Here's a list of the main features :

1. Lists
 - Nested lists
 - Numbered and bullet lists
2. Tables
3. Images
4. Inline styles
5. Cascading style sheets

- Images loaded from the application resources



- Richly formatted HTML error messages



- Plain text and cascading style sheets

SuperTooltip for WinForms Quick Start

This quick start guide explains how to add two Microsoft Vista-style **C1SuperTooltips** and a **C1SuperLabel** control to your form.

Step 1 of 4: Adding a C1SuperLabel to a Project

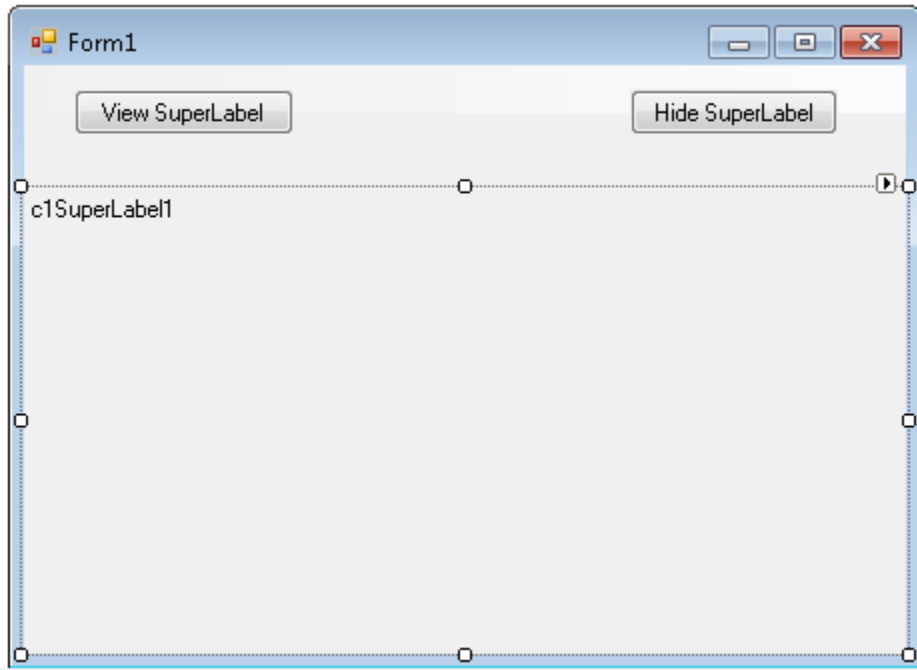
Start by adding a **C1SuperLabel** control to your project and adding some HTML code for it.

To add a **C1SuperLabel** to a project, complete the following steps:

1. [Create a new .NET project](#) (page 10) and add two button controls to the form.



2. Right-click the **Button1** control and select **Properties** to view the Properties window.
3. Enter "View C1SuperLabel" next to the **Text** property and resize the button so the text is visible.
4. Select **Button2** and enter "Hide C1SuperLabel" next to its **Text** property. Resize the button so the text is visible.
5. [Add the SuperTooltip for WinForms components](#) (page 10) to the Toolbox.
6. Double-click the **C1SuperLabel** control in the Toolbox to add it to your form so it looks like the following:



7. Click the **ellipsis** button next to C1SuperLabel's Text property in the Properties window. The **C1SuperLabel Editor** appears.
8. Click here for the HTML code to add in the text box.

```

These are just some of the things you can show using <b>C1SuperTooltip</b>:
<ol>
  <li>Lists:</li>
  <ul>
    <li>Nested lists</li>
    <li>Numbered and bulleted lists</li>
  </ul>
  <li>Tables:<p>Here's a table with some sales numbers:</p>
  <table border='1' bordercolor='black' cellpadding='2' cellspacing='0'
  style='border:solid 1 black;'>
    <tr bgcolor='LightSteelBlue'>
      <td style="width:150px; text-align: center;" rowspan='3'>
        <strong>This cell spans<br />three rows.</strong></td>
      <td align='center' colspan='6'><strong>Semester 1</strong></td>
    </tr>
    <tr bgcolor='LightSteelBlue'>
      <td align='center' colspan='3'><strong>Quarter 1</strong></td>
      <td align='center' colspan='3'><strong>Quarter 2</strong></td>
    </tr>
    <tr bgcolor='LightSteelBlue'>
      <td align='center'>Jan</td>
      <td align='center'>Feb</td>
      <td align='center'>Mar</td>
      <td align='center'>Apr</td>
      <td align='center'>May</td>
      <td align='center'>Jun</td>
    </tr>
    <tr>
      <td align='right' style='width: 150px'><strong>Widgets</strong></td>
      <td>12</td>
      <td>23</td>
      <td>23</td>
      <td>43</td>
      <td>23</td>
      <td>34</td>
    </tr>
  </table>
  </li>
</ol>

```

```

        <tr>
            <td align='right' style='width: 150px'><strong>Sprockets</strong></td>
            <td>45</td>
            <td>32</td>
            <td>23</td>
            <td>23</td>
            <td>34</td>
            <td>43</td>
        </tr>
    </table>
</li>
<li>Images: The background of this C1SuperLabel is an image. </li>
<li>Borders:<p style='border: #336633 thick'>Here's a thick border.</p>
</li>
<li>Preformatted Text:
    <pre style="background-color:#dddddd; margin:0 20pt 0 20pt;">
string tipText = "Hello &lt;b>World&/b>";
<b>C1SuperTooltip</b>.SetToolTip(control, tipText);
    </pre>
</li>
<li>Nesting:
    <div style="background-color:#ffaaaa;border:solid thin
black;margin:10px;padding:4px">
        This is a red div.
        <div style="background-color:#aaffaa;border:solid thin
black;margin:10px;padding:4px">
            This is a nested green div.
            <div style="background-color:#aaaaff;border:solid thin
black;margin:10px;padding:4px">
                This is a nested blue div.
            </div>
        </div>
    </div>
</li>
</ol>

```

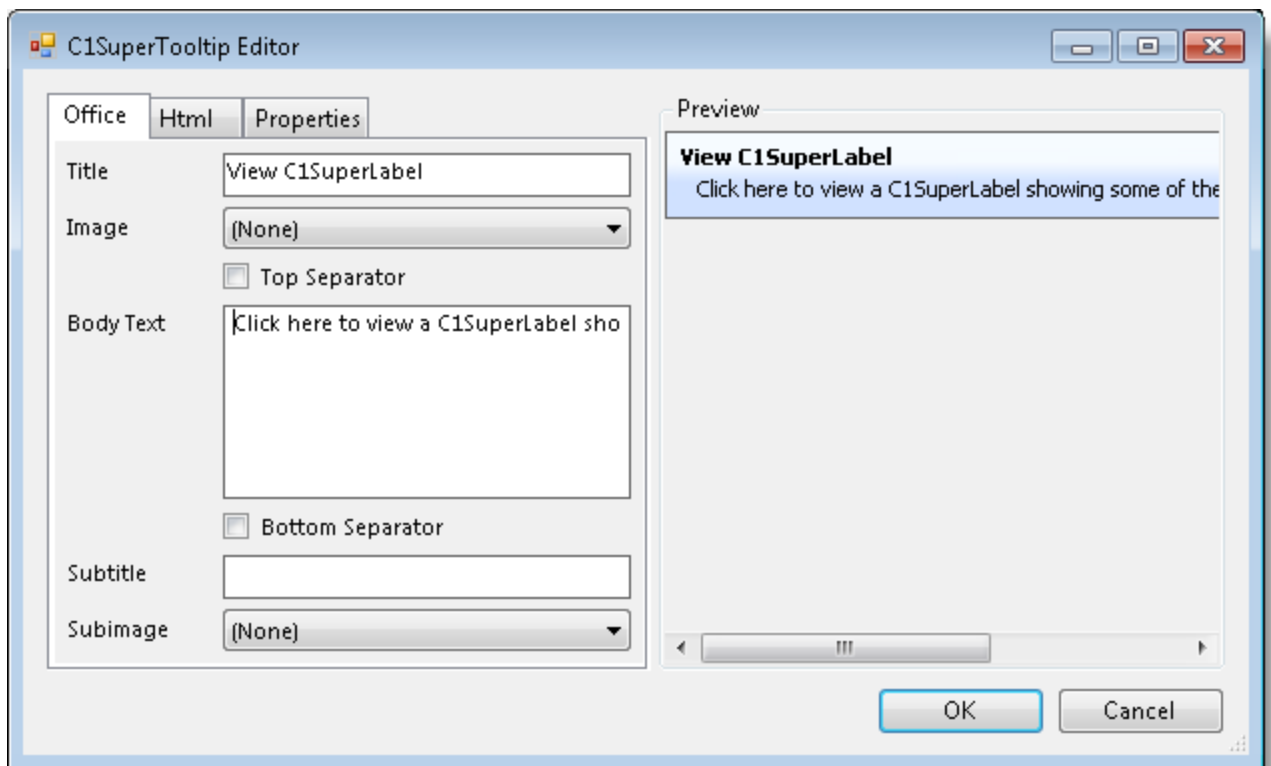
9. In the Properties window, click the drop-down arrow next to the **BackColor** property, select the **Web** tab, and select **White**.
10. Add an image to the C1SuperLabel control:
 - Click the **ellipsis** button next to the BackgroundImage property. The **Select Resource** dialog box appears.
 - Select **Local resource** and click the **Import** button. The **Open** dialog box appears.
 - Locate and select the **TipBackground.png** or another graphic of your choice and then click **Open**. The TipBackground.png image is installed, by default, with the SuperTooltip for WinForms samples and is located in C:\Documents and Settings\\My Documents\ComponentOne Samples\Studio for WinForms\C1SuperTooltips\Resources folder (Windows XP) or C:\Users\\Documents\ComponentOne Samples\Studio for WinForms\C1SuperTooltip\CS\SuperTooltips\Resources folder (Windows 7/Vista).
 - Click **OK**.
11. Click the drop-down arrow next to the BackgroundImageLayout property and select **Stretch**.
12. Set the **Visible** property to **False**.

Step 2 of 4: Creating a Vista-style C1SuperTooltip for a Control

Next, create a Microsoft Vista-style C1SuperTooltip using the **Office** tab in the **C1SuperTooltip Editor**. To do this, complete the following steps:

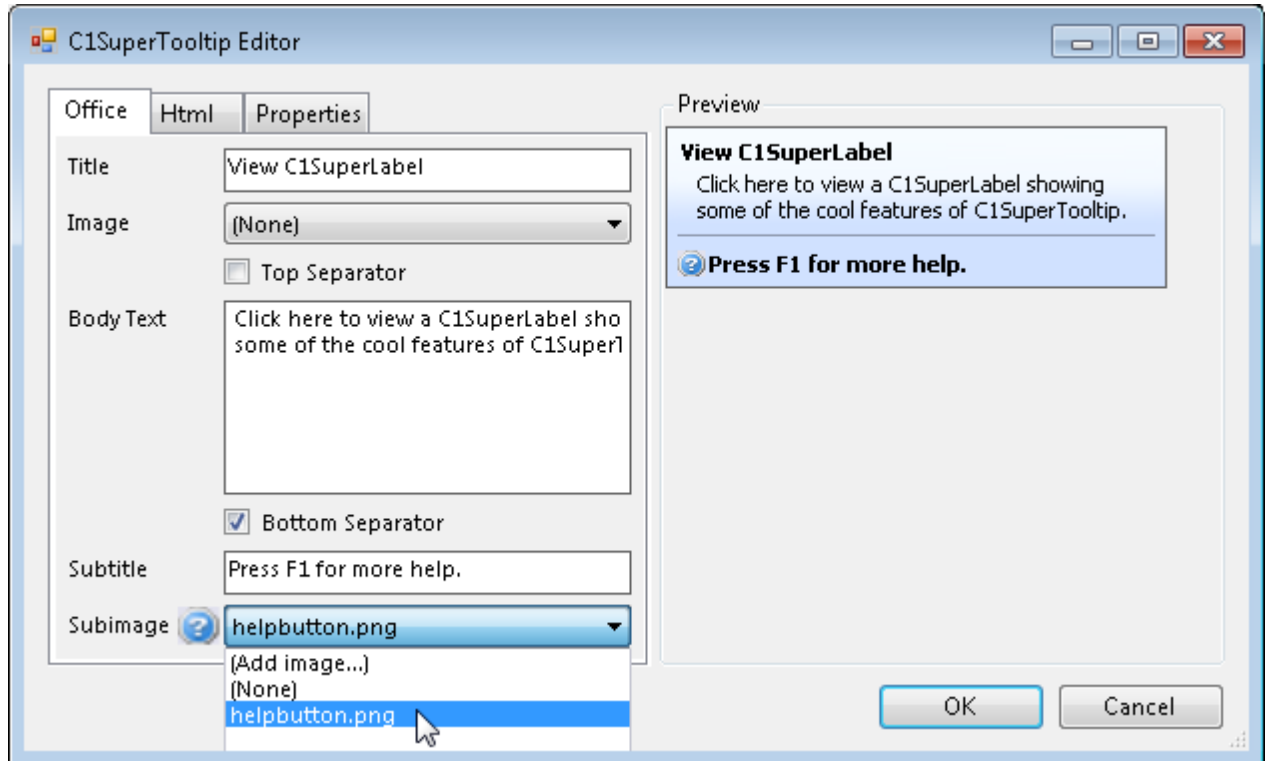
1. Double-click the C1SuperTooltip control in the Toolbox to add it to your form.

2. First, we will add an image to the ToolTip's image collection. We are using a small Help graphic to be used as the Subimage in our ToolTip.
 - a. Select **C1SuperTooltip1** and click the **ellipsis** button next to the Images property in the Properties window. The **Edit Image Collection** editor appears.
 - b. Click the **Add** button, and find and select the desired image file.
 - c. Once the image is added to the collection, click **OK**. We will specify this image in the ToolTip a little later.
3. Right-click the **Button1** control and select **Properties** to view the Properties window.
4. Click the **ellipsis** button next to the **ToolTip on C1SuperTooltip1** property and the **C1SuperTooltip Editor** appears.
5. On the **Office** tab, enter **View C1SuperLabel** in the **Title** text box.
6. Enter the following text in the **Body Text** text box:
[Click here to view a C1SuperLabel showing some of the cool features of C1SuperTooltip.](#)
 Notice a preview of the C1SuperTooltip appears in the **Preview** window.



7. To make the ToolTip appear on multiple lines, place the cursor after the word *showing* in the **Body Text** and press **ENTER**.
8. You can also format the body text from within the **Office** tab. Select and right-click **C1SuperLabel**, and then choose **Bold** from the context menu or click **Ctrl+B** on the keyboard. Do the same for **C1SuperTooltip**.
9. Check the **Bottom Separator** check box.

10. Enter the following text in the **Subtitle** text box:
Press F1 for more help.
11. Click the **Subimage** drop-down arrow and select the image you added to the collection in the previous step 2.



Note that C1SuperTooltip automatically creates the HTML used to format your Tooltip so you don't have to. To view the HTML code, click the **Html** tab. We will create a Tooltip using only HTML code in the **Creating a C1SuperTooltip with HTML Code** topic of this quick start.

12. Click the **Properties** tab in the editor.
13. Click the drop-down arrow next to the BackgroundGradient property and choose **Vista**. The background and layout of the Tooltip now have the Microsoft Vista Tooltip style.
14. Click **OK** to close the **C1SuperTooltip Editor**.

Step 3 of 4: Creating a C1SuperTooltip with HTML Code

Then create a C1SuperTooltip in the **C1SuperTooltip Editor** using only HTML code. Complete the following steps:

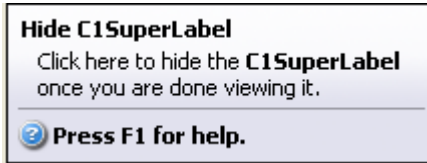
1. Select the **Button2** control.
2. Click the **ellipsis** button next to the **Tooltip on C1SuperTooltip1** property to open the **C1SuperTooltip Editor**.
3. Click the **Html** tab and enter the following HTML markup:

```

<table><tr>
  <td><parm></parm>
  <td><b><parm>Hide C1SuperLabel</parm></b>
</table>
<parm></parm>
<div style='margin:1 12'><parm>
```

```
Click here to hide the <b>C1SuperLabel</b><br>once you are done viewing it.
</parm></div>
<parm><hr noshade size=1 style='margin:2' color=Darker></parm>
<table><tr>
  <td><parm><img src='HelpButton.png'></parm>
  <td><b><parm>Press F1 for help.</parm></b>
</table>
```

4. Click **OK** to close the **C1SuperTooltip Editor**. Note that the ToolTip will appear similar to the following:



Step 4 of 4: Adding Code for the Buttons and Running the Project

Once you have created the C1SuperLabel and **C1SuperTooltips**, you can add code for the buttons on the form and run the project. To do this, complete the following steps:

1. Double-click **Button1** and add the following code to the **Button1_Click** event:

- Visual Basic

```
C1SuperLabel1.Visible = True
```

- C#

```
c1SuperLabel1.Visible = true;
```

2. Double-click **Button2** and add the following code to the **Button2_Click** event:

- Visual Basic

```
C1SuperLabel1.Visible = False
```

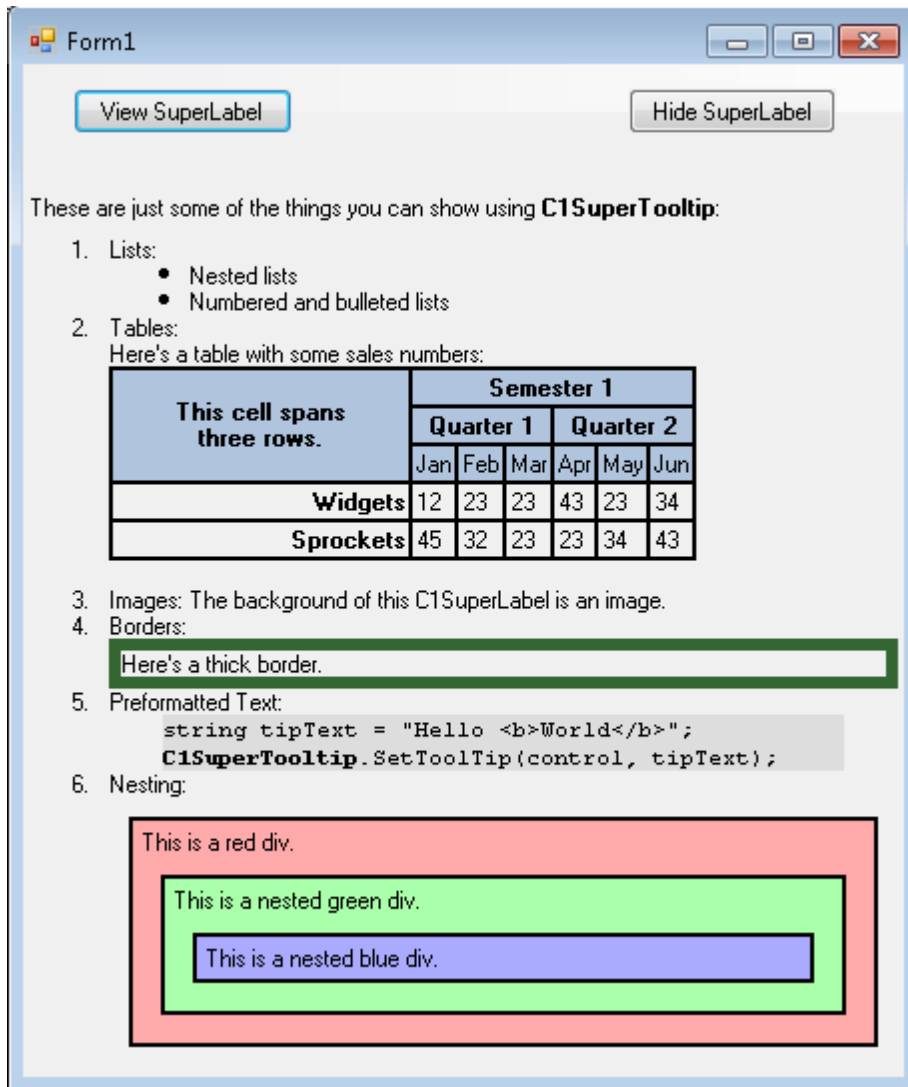
- C#

```
c1SuperLabel1.Visible = false;
```

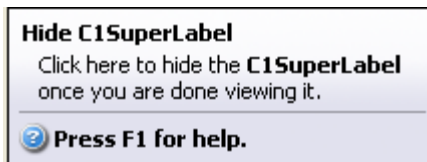
3. Run the project and mouse over **Button1**. Notice the ToolTip is a Vista-style C1SuperTooltip:



4. Click **View C1SuperLabel**. The C1SuperLabel appears.



5. Mouse over **Button2**. The Vista-style C1SuperTooltip looks like this:



The text in the ToolTip for **Button2** was applied through HTML markup, but you will notice the C1SuperTooltip Vista background style appears in both ToolTips. Any properties you set via the **C1SuperTooltip Editor** will be the same in all ToolTips for controls associated with that C1SuperTooltip.

6. Click **Hide C1SuperLabel**, and the C1SuperLabel is hidden.

SuperTooltip for WinForms Top Tips

The following tips were compiled from frequently asked user questions posted in the [Studio for ASP.NET forum](#).

Tip 1: Use Visual Studio to create the HTML that goes into the super tooltips.

The C1SuperTooltip has a designer that makes it easy to create standard Office 2007-style tooltips with images, headers, and footers.

But you have a lot more flexibility than that. The C1SuperTooltip has a powerful HTML parser, so you can use it to display virtually any HTML you want. For example, you can create HTML content in Visual Studio (**File | New | File... | HTML Page**), with style sheets, lists, tables, and so on, and use that as your tooltip contents. Just paste the resulting HTML into the “Tooltip on c1SuperTooltip” extender property for any control.

See [Adding a C1SuperTooltip using HTML](#) (page 41) for an example.

Tip 2: Use the MaximumWidth property to make the tooltip content wrap.

By default, the C1SuperTooltip will break lines only at the end of paragraphs or at explicit line breaks. By setting the MaximumWidth property, you can have the content wrap automatically, which makes maintaining the content much easier.

Tip 3: Use the Images collection to add images to the C1SuperTooltip and C1SuperLabel.

Both the C1SuperTooltip and C1SuperLabel have an **Images** property that you can use to define a list of images that you want to display in the controls. Simply add as many images as you want to the **Images** collection, and then refer to the images in your HTML content by using IMG tags with a “res://” qualifier. For example, if you add an image called “MyBitmap.png” to the **Images** collection, you can use it in your HTML text as follows:

```
<img src= "res://mybitmap.png" />
```

See [Adding an Image to C1SuperTooltip](#) (page 45) and [Adding an Image to C1SuperLabel](#) (page 47) for examples.

Tip 4: Use the Opacity property to provide see-through tips.

The C1SuperToolTip has an Opacity property that takes a value between zero (transparent) and one (solid). By setting the Opacity property to 0.5 for example, you can provide tooltips that allow users to see the content underneath the tip. This can be especially useful if the tooltips are large.

Tip 5: Use the IsBalloon and BackgroundGradient properties to provide tips that stand out.

In addition to the rich HTML rendering, the C1SuperToolTip has several properties that allow you to customize the appearance of the tooltips themselves. The IsBalloon and BackgroundGradient properties for example are very easy to use and allow you to provide tooltips that really stand out.

See [C1SuperToolTip Background Gradient](#) (page 32) and [C1SuperToolTip Shape](#) (page 33) for more information on these properties.

Design-Time Support

ComponentOne SuperTooltip for WinForms provides visual editing to make it easier to create a ToolTip. The following sections describe how to use **C1SuperTooltip's** design-time environment to configure the **SuperTooltip for WinForms** controls:

Tasks Menus

A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each control. You can invoke each control's tasks menu by clicking on the smart tag (☐) in the upper-right corner of the control. For more information, see [C1SuperTooltip Menus](#) (page 23).

Context Menus

You can also access some of the short-cuts found in the tasks menu through each control's context menu. You can invoke each control's context menu by right-clicking on the control.

Properties Window

You can also easily configure C1SuperTooltip at design time using the **Properties** window in Visual Studio. You can access the **Properties** window by right-clicking the control and selecting **Properties**.

Editors

You can also quickly configure C1SuperTooltip at design time using the C1SuperTooltip Editors. You can access the Editors through the **Properties** window. For more information, see [C1SuperTooltip Editors](#) (page 25).

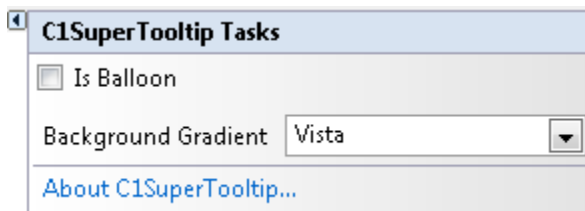
C1SuperTooltip Menus

The following sections describe how to use **C1SuperTooltip's** tasks and context menus to configure the **SuperTooltip for WinForms** controls.

C1SuperTooltip Tasks and Context Menus

C1SuperTooltip Tasks Menu

To access the **C1SuperTooltip Tasks** menu, click the smart tag in the upper-right corner of the C1SuperTooltip control. The **C1SuperTooltip Tasks** menu appears. You can set the **IsBalloon** and **BackgroundGradient** properties through the **C1SuperTooltip Tasks** menu.



- **Is Balloon**
Check this checkbox if you want the ToolTip to appear within a balloon shape rather than in a rectangular box.
- **Background Gradient**
Specifies the background gradient for the ToolTip. You can choose **Automatic** (current system Visual Style), **Blue**, **Gold**, **None**, **Olive**, **Silver** or **Vista** (Microsoft Vista style).

- **About C1SuperTooltip**
Displays the **About ComponentOne C1SuperTooltip** dialog box containing the version number, along with licensing, registration and purchasing information, and additional online resources.

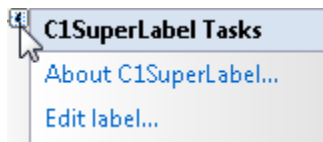
C1SuperTooltip Context Menu

To access C1SuperTooltip's context menu, right-click the C1SuperTooltip component. The **C1SuperTooltip** context menu appears.

C1SuperLabel Tasks and Context Menus

C1SuperLabel Tasks Menu

To access the **C1SuperLabel Tasks** menu, click the smart tag in the upper-right corner of the C1SuperLabel control. The **C1Label Tasks** menu appears. By clicking **Edit label** you can open the **C1SuperLabel Editor**.



- **About C1SuperLabel**
Displays the **About ComponentOne C1SuperTooltip** dialog box containing the version number, along with licensing, registration and purchasing information, and additional online resources.
- **Edit label**
Opens the **C1SuperLabel Editor** so you can begin adding HTML code to create a C1SuperLabel.

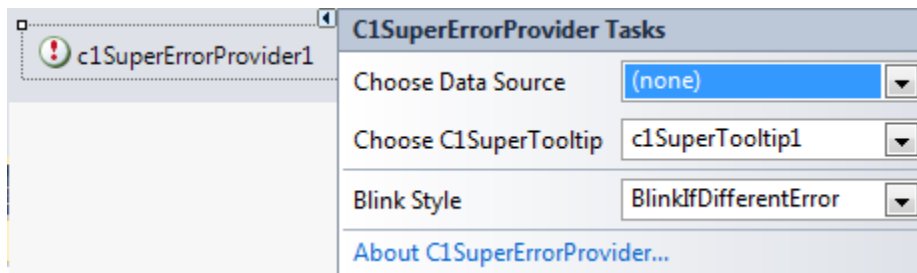
C1SuperTooltip Context Menu

To access **C1SuperTooltip's** context menu, right-click on the C1SuperTooltip component. The C1SuperTooltip context menu appears. By selecting **Edit label** you can open the **C1SuperLabel Editor**.

C1SuperErrorProvider Tasks and Context Menus

C1SuperErrorProvider Tasks Menu

To access the **C1SuperErrorProvider Tasks** menu, click the smart tag in the upper-right corner of the C1SuperErrorProvider control. The **C1SuperErrorProvider Tasks** menu appears.



- **Choose Data Source**
Use this drop-down to select a data source that you can attach to a control and that you want to monitor for errors.

- **Choose C1SuperTooltip**
Click this drop-down and select a C1SuperTooltip to associate it with the error provider component. The C1SuperTooltip will be used for displaying the error description text. It must be set in order for the error message to appear when the mouse pointer hovers the error icon, although it can be blank. If it is blank, the error icon won't have the associated tool tip.
- **Blink Style**
Select the way you would like the default error message icon to appear: **BlinkIfDifferentError**, **AlwaysBlink**, or **NeverBlink**.
- **About C1SuperErrorProvider**
Displays the **About ComponentOne C1SuperTooltip** dialog box containing the version number, along with licensing, registration and purchasing information, and additional online resources.

C1SuperErrorProvider Context Menu

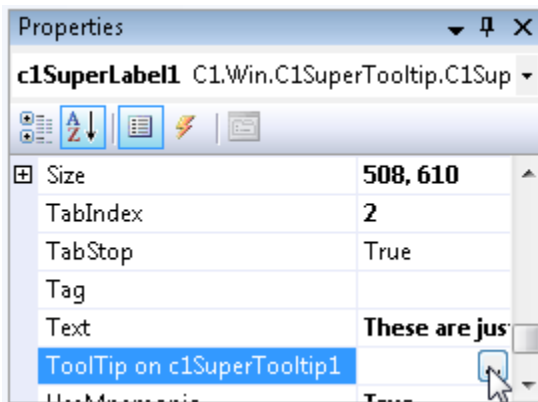
To access C1SuperErrorProvider's context menu, right-click the C1SuperErrorProvider component. The C1SuperErrorProvider context menu appears.

C1SuperTooltip Editors

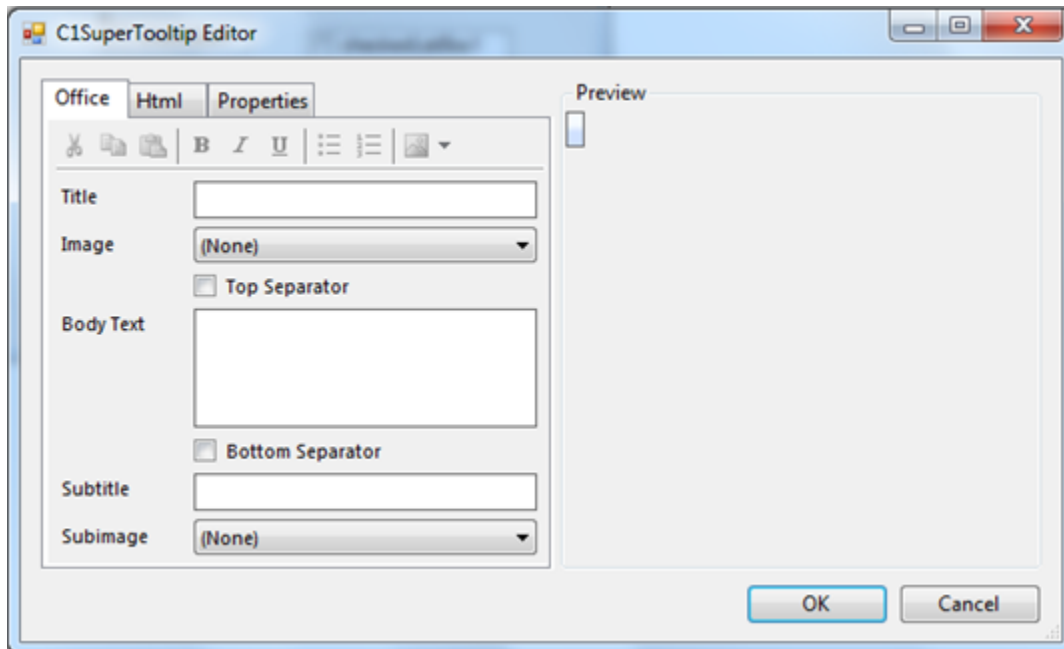
The following sections describe how to use **C1SuperTooltip's** Editors to configure the **C1SuperToolTip** controls.

C1SuperTooltip Editor

To access the **C1SuperTooltip Editor**, access the Properties window for the control that will contain the ToolTip and select the **ellipsis** button next to **ToolTip on C1SuperTooltip**, as in the image below. The **C1SuperTooltip Editor** appears.



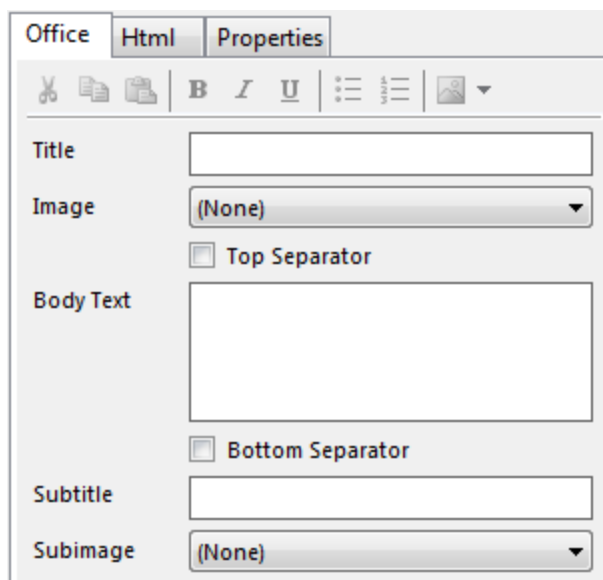
The **C1SuperTooltip Editor** consists of a tabbed design pane on the left, where you can change the appearance and content of the ToolTip, and a preview pane on the right, where you can view changes that have been made to the ToolTip.



At design time, there are two ways you can create the content of a Tooltip: using the **Office** tab or using **Html** tab to manually enter your own HTML code. You can use the **Properties** tab to change the Tooltip's appearance and behavior settings.

Office tab

In the **C1SuperTooltip Editor** you can use the **Office** tab to add images, a title, a subtitle, and the body text of the Tooltip. C1SuperTooltip automatically creates all of the HTML code behind the Tooltip, saving you time and work. The various elements in the **Office** tab allow you to quickly customize the content of your Tooltip, for example as in the image below.



Title

This is the text that appears at the top of the ToolTip; you can add HTML code to customize the appearance of the title text.

Image

This is the image that appears to the left of the Title. You can add an image by clicking the drop-down arrow and selecting **Add image**. Images that you have added using the [Edit Image Collection Editor](#) (page 30) are listed and can be added to the ToolTip.

Top Separator

By checking this check box, you will add the top separator to the ToolTip. The top separator is a horizontal, dark grey rule that appears between the ToolTip's title and the body text.

Body Text

The body text is the main content of the ToolTip; you can add HTML to customize the appearance of the body text.

Bottom Separator

By checking this check box, you will add the bottom separator to the ToolTip. The top separator is a horizontal, dark grey rule that appears between the ToolTip's body text and subtitle text.

Subtitle

This is text that appears below the ToolTip's body text; you can add HTML code to customize the appearance of the title text.

Subimage

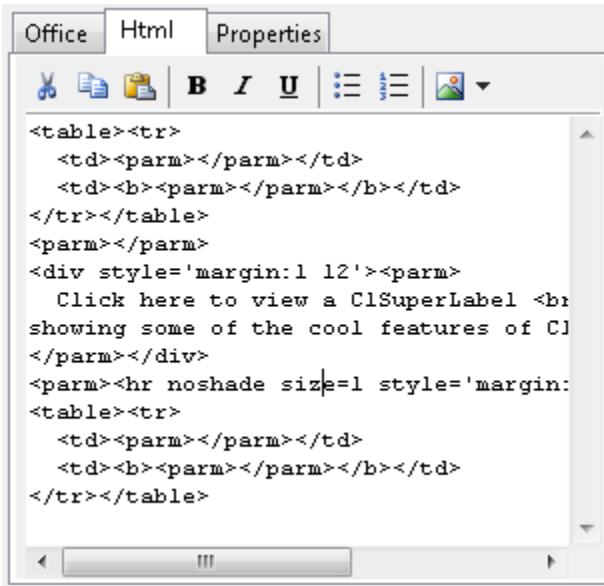
This is the image that appears to the left of the subtitle. You can add an image by clicking the drop-down arrow and selecting **Add image**. Images that you have added using the [Edit Image Collection Editor](#) (page 30) are listed and can be added to the ToolTip.

For more information about creating a ToolTip using the **Office** tab, see the [Creating C1SuperTooltips at Design Time](#) (page 36) topic.

Html tab

In the **Html** tab of the **C1SuperToolTip Editor** you can view and edit HTML code that reflects changes you have made in the **Office** tab. You can also use the **Html** tab to create a ToolTip by entering all of your own HTML code if you choose not to have the editor do it for you. By creating a ToolTip in the **Html** tab you have more control over each line of the ToolTip.

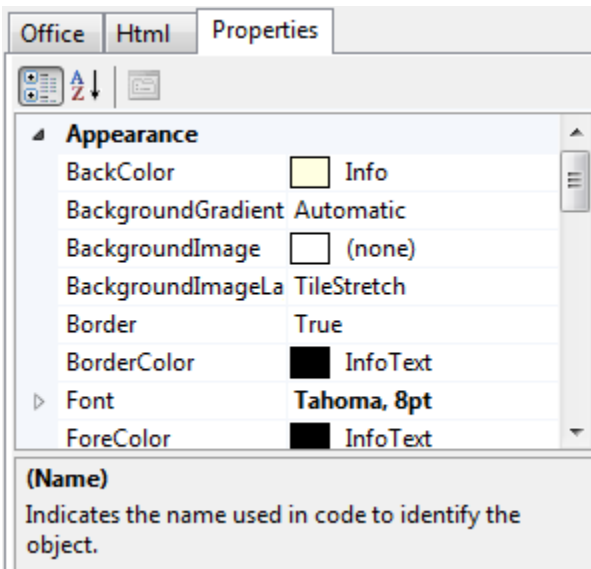
The **Html** tab provides a formatting toolstrip with the following buttons: **Cut**, **Copy**, **Paste**, **Bold**, **Italic**, **Underline**, **Bulleted List**, **Numbered List**, and **Insert Image**.



For more information about creating a Tooltip using the **Html** tab, see the [Adding a C1SuperTooltip using HTML](#) (page 41) topic.

Properties tab

The **Properties** tab of the **C1SuperTooltip Editor** allows you to customize the overall appearance and behavior of the Tooltip, which will be applied to all controls associated with it. While the settings available in the **Properties** tab of the editor are the same as in the Properties window of the C1SuperTooltip, the advantage of the Properties tab is that you can quickly preview content and appearance changes in the [Preview pane](#) (page 29).



For more information about changing Tooltip appearance and behavior settings using the **Properties** tab, see the [Changing the C1SuperTooltip Appearance and Behavior Settings](#) (page 43) topic.

Preview pane

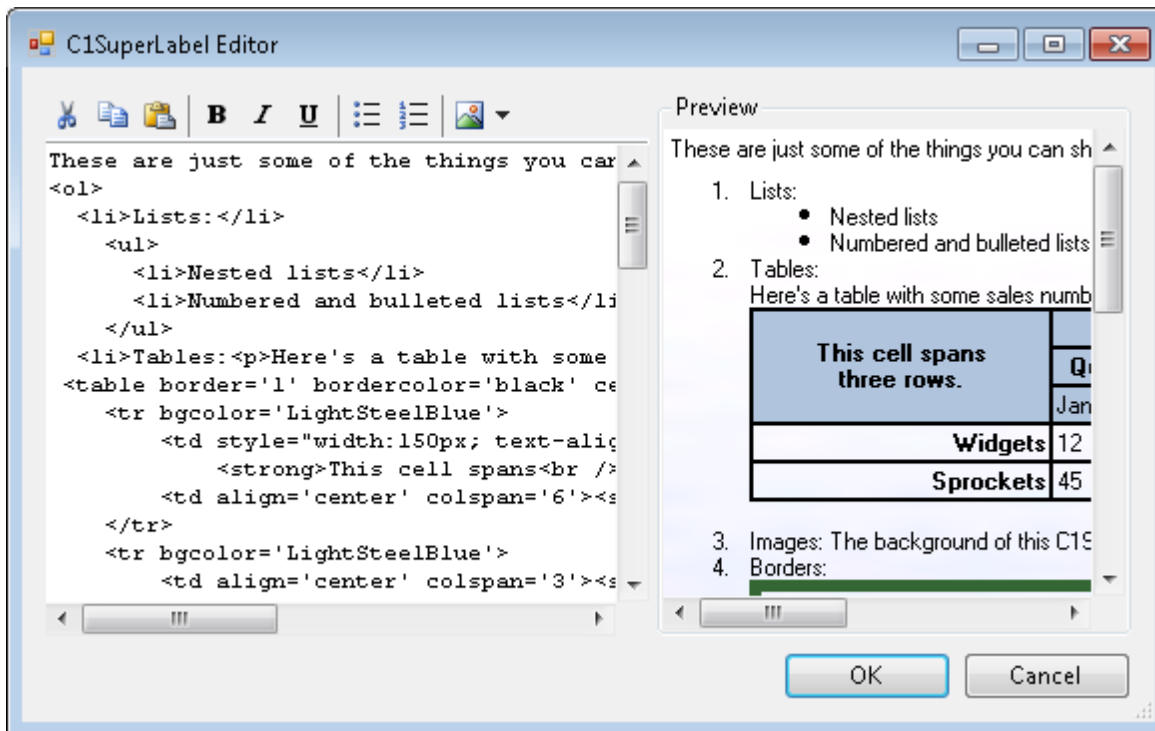
The **Preview** pane of the **C1SuperToolTip Editor** allows you to preview the ToolTip. The **Preview** pane appears on the right side of the **C1SuperToolTip Editor** and reflects any changes that you make to the ToolTip allowing you to more easily visualize and change the ToolTip's content and appearance.



C1SuperLabel Editor

To access the **C1SuperLabel Editor**, select **Edit label** from the **C1SuperLabel Tasks** menu. The **C1SuperLabel Editor** appears.

Using the **C1SuperLabel Editor**, you can control the content and appearance of the **C1SuperLabel**. You can enter text and HTML code in the left-side text area of the **C1SuperLabel Editor** and preview the content in the right-side preview area, as in the example below.



The **C1SuperLabel Editor** also provides a formatting toolstrip with the following buttons: **Cut**, **Copy**, **Paste**, **Bold**, **Italic**, **Underline**, **Bulleted List**, **Numbered List**, and **Insert Image**.

Edit Image Collection Editor

You can add images to be used in the **C1SuperToolTip** components through the **Edit Image Collection** editor. You can access the **Edit Image Collection** editor through the Properties window of the **C1SuperToolTip** component and the **C1SuperLabel** control.

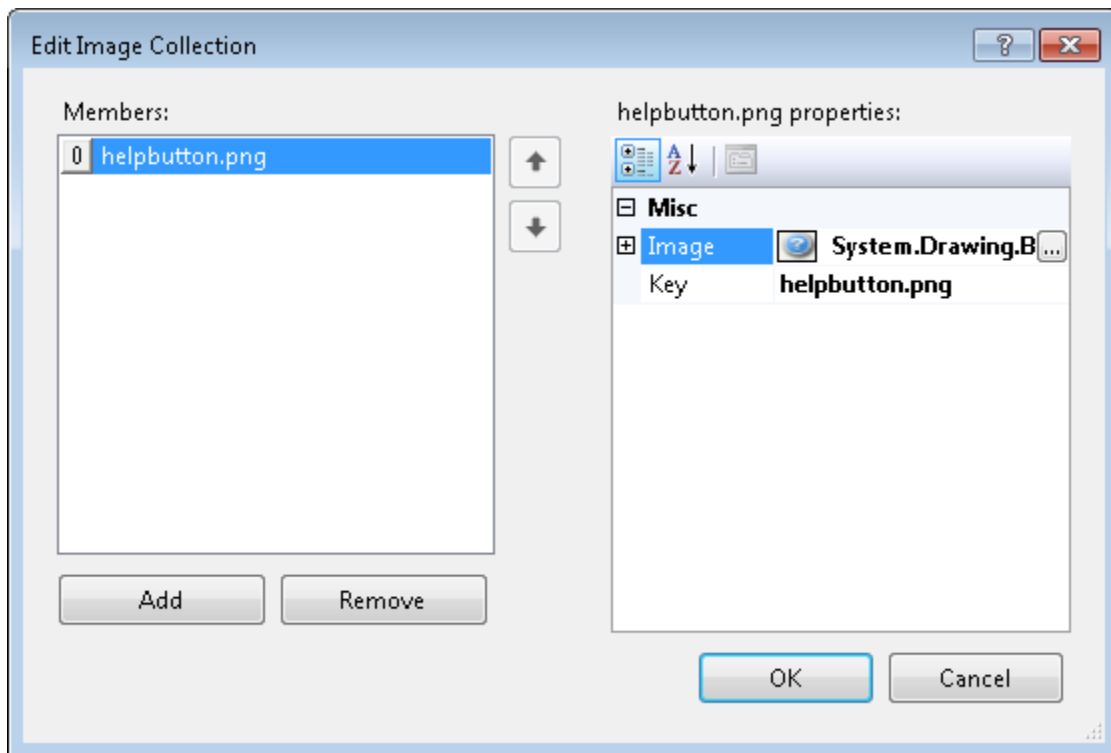
Accessing the Edit Image Collection editor from C1SuperToolTip

To open the **Edit Image Collection** editor select the **C1SuperToolTip** control and click the **ellipsis** button next to the Images property in the Properties window. The **Edit Image Collection** editor appears.

Accessing the Edit Image Collection editor from C1SuperLabel

To open the **Edit Image Collection** editor select the **C1SuperLabel** control and click the **ellipsis** button next to the Images property in the Properties window. The **Edit Image Collection** editor appears.

In the left pane of the **Edit Image Collection** editor you can see what image members have been added and can add and remove images from the collection. In the right pane you can view and change each image's properties, as in the image below.



Adding an image to **C1SuperToolTip** controls is easy; for more information see [Adding an Image to C1SuperToolTip](#) (page 45) and [Adding an Image to C1SuperLabel](#) (page 47).

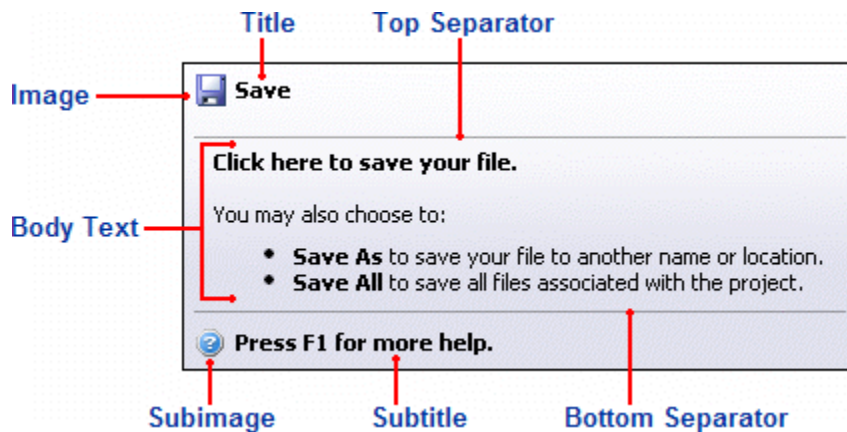
C1SuperToolTip Elements

This section provides a visual and descriptive overview of the elements that comprise the **C1SuperToolTip** and **C1SuperLabel** controls.

ToolTip Elements

The C1SuperToolTip control consists of several elements that can be modified through the **Office** tab of the **C1SuperToolTip Editor**. For more information about the **C1SuperToolTip Editor** and these elements, please see the [Office tab](#) (page 26) topic.

Note: If you choose to add HTML code directly to the C1SuperToolTip or add a C1SuperToolTip programmatically, elements will not be automatically added and must be formatted manually through the HTML code (for example adding a separator by adding the <hr> horizontal rule tag).



Title

This is the text that appears at the top of the ToolTip; you can add HTML code to customize the appearance of the title text.

Image

This is the image that appears to the left of the Title. You can add an image through the **C1SuperToolTip Editor** or you can add an image to the collection by using the **Edit Image Collection Editor**.

Top Separator

The top separator is a horizontal rule that appears between the ToolTip's title and the body text. You can add a top separator through the **Office** tab of the **C1SuperToolTip Editor** or you can add a horizontal rule to the HTML code by using the <hr> tag.

Body Text

The body text is the main content of the ToolTip; you can add HTML to customize the appearance of the body text.

Bottom Separator

The bottom separator is a horizontal rule that appears between the ToolTip's body text and subtitle. You can add a bottom separator through the **Office** tab of the **C1SuperToolTip Editor** or you can add a horizontal rule to the HTML code by using the <hr> tag.

Subtitle

This is text that appears below the ToolTip's body text; you can add HTML code to customize the appearance of the subtitle text.

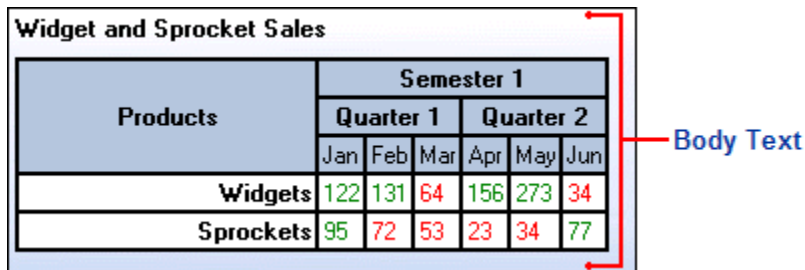
Subimage

This is the image that appears to the left of the subimage. You can add an image through the **C1SuperToolTip Editor** or you can add an image to the collection by using the **Edit Image Collection Editor**.

For more information about creating a ToolTip using the **Office** tab, see the [Creating C1SuperTooltips at Design Time](#) (page 36) topic.

Label Elements

The C1SuperLabel control consists of a main body in which HTML content can be added. In the following example, a table was added to the label using HTML. For more information, see the [Creating C1SuperLabels](#) (page 45) topic.



Products	Semester 1					
	Quarter 1			Quarter 2		
	Jan	Feb	Mar	Apr	May	Jun
Widgets	122	131	64	156	273	34
Sprockets	95	72	53	23	34	77

You can also quickly add and preview content using the **C1SuperLabel Editor**. For more information about the editor, see the [C1SuperLabel Editor](#) (page 29) topic.

ErrorProvider Elements

The C1SuperErrorProvider control consists of an icon image, along with any formatted HTML text you want to include to indicate an error.



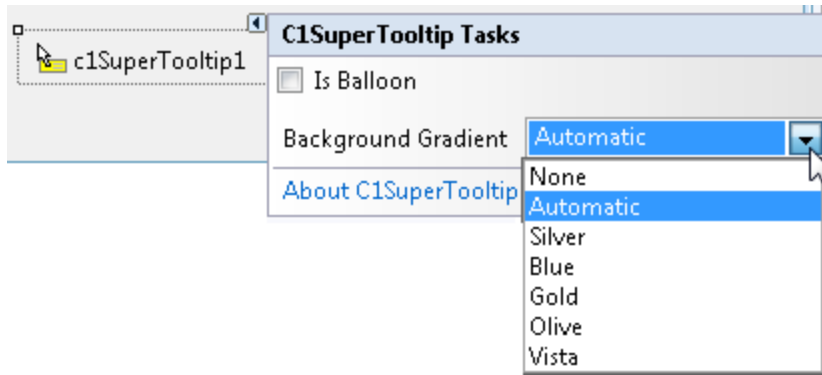
For more information on providing error messages, see the [Creating C1SuperErrorProvider Error Messages](#) (page 48) topic.

C1SuperTooltip Appearance

This section provides a visual and descriptive overview of the settings available to customize the appearance of the **SuperTooltip for WinForms** controls.

C1SuperTooltip Background Gradient

You can customize the appearance of the C1SuperTooltip quickly and easily by changing the BackgroundGradient property. You can access the BackgroundGradient property at design time through the Properties window or by selecting the **Background Gradient** drop-down box from the **C1SuperTooltip Tasks** menu. By default the BackgroundGradient property is set to **Automatic**.



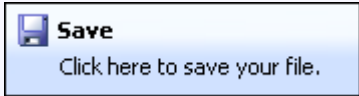
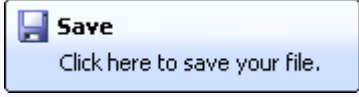
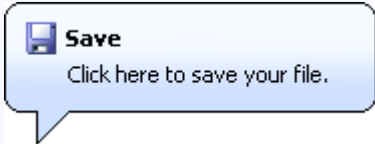
The following background gradients are available for **C1SuperTooltip**:

Background Gradient	Preview
None	
Automatic	
Silver	
Blue	
Gold	
Olive	
Vista	

C1SuperTooltip Shape

By using the `IsBalloon` and `RoundedCorners` properties you can customize the appearance of the `C1SuperTooltip`. You can access the `IsBalloon` and `RoundedCorners` properties through the Properties window. You can also access the `IsBalloon` property through the **C1SuperTooltip Tasks** menu.

The following shape settings are available for **C1SuperTooltip**:

Shape Settings	Preview
IsBalloon = False RoundedCorners = False	
IsBalloon = False RoundedCorners = True	
IsBalloon = True RoundedCorners = False	

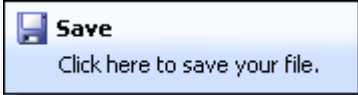
If both the `IsBalloon` and `RoundedCorners` properties are set to **False** (default), the Tooltip appears as a rectangle. If `RoundedCorners` is set to **True**, the Tooltip appears as a rounded rectangle. If `IsBalloon` is set to **True**, the Tooltip appears in a balloon shape.

C1SuperTooltip Shadow

By setting the `Shadow` property you can determine whether a shadow will appear below a **C1SuperTooltip**. By default the `Shadow` property is set to **True**.

You can access the `Shadow` property through the Properties window.

The following shadow settings are available for **C1SuperTooltip**:

Shape Settings	Preview
Shadow = True	
Shadow = False	

SuperTooltip for WinForms Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studios.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on the desktop, click the **Start** button and then click **ComponentOne | Studio for WinForms | Samples | SuperTooltip Samples**. The following tables provide a short description for each sample.

Visual Basic and C# Samples

ComponentOne SuperTooltip for WinForms includes the following Visual Basic and C# samples:

Sample	Description
SuperErrorProvider	This sample shows how to use the C1SuperErrorProvider component with data sources and individual controls.
SuperLabels	Demonstrates the different items that can be placed within C1SuperLabel controls, such as lists, tables, preformatted text and images. This sample uses the C1SuperLabel control.
SuperTooltips	Demonstrates the different items that can be placed within C1SuperTooltip controls, such as lists, tables, preformatted text and images. This sample uses the C1SuperTooltip control.
GridTips	Demonstrates two methods that can be used to show a C1SuperTooltip control on demand. This sample uses the C1SuperTooltip and C1FlexGrid controls.
ShowTooltips	Demonstrates the Show() and Hide() methods in the standard ToolTip and C1SuperTooltip controls. This sample uses the C1SuperTooltip control.
TextDrivenSuperTooltip (C# only)	Shows how you can load SuperTooltips from an XML file at run time.
ThumbnailTips	Demonstrates how to build ToolTips with dynamically generated thumbnail images. This sample uses the C1Chart , C1FlexGrid , and C1SuperTooltip controls.

SuperTooltip for WinForms Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET, and know how to use bound and unbound controls in general. If you are a novice to the **ComponentOne SuperTooltip for WinForms** product, please see the [SuperTooltip for WinForms Quick Start](#) (page 13) first.

Each topic provides a solution for specific tasks using the **SuperTooltip for WinForms** product. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **SuperTooltip for WinForms** features.

Each task-based help topic also assumes that you have created a new .NET project. For additional information on this topic, see [Creating a .NET Project](#) (page 10).

Creating C1SuperTooltips

The following topics explain how to create **C1SuperTooltips** at design time using the **C1SuperTooltip Editor**, in code, by using cascading style sheets, and using HTML, as well as how to add multiple ToolTips, adjust **C1SuperTooltip** appearance and behavior settings, and add images to ToolTips.

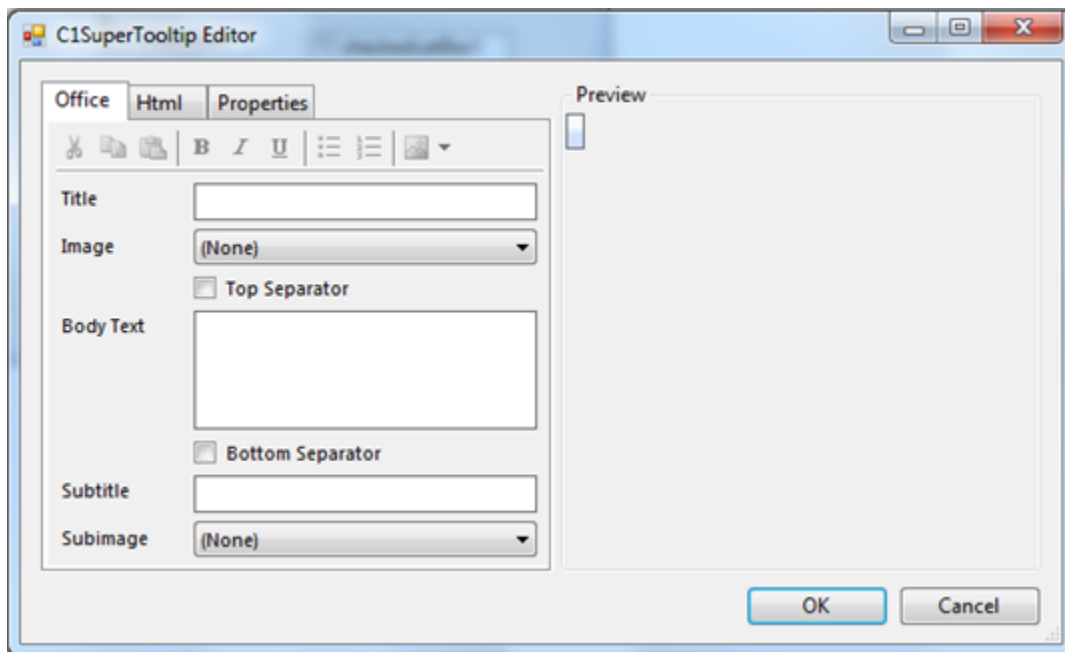
Creating C1SuperTooltips at Design Time

C1SuperTooltip provides a design-time editor, **C1SuperTooltip Editor**, to simplify the process of creating ToolTips in your applications. The following topic will show you how to create ToolTips and change their appearance and behavior using this editor. A Tooltip is used to display helpful information when a mouse hovers over an item in an application. Usually ToolTips contain only plain text. Using C1SuperTooltip, you can display HTML content, including images, tables, and numbered, bulleted, or nested lists.

You can associate C1SuperTooltip with any System.Windows.Forms control or System.Windows.Forms.ToolStripItem.

To add a C1SuperTooltip to your application:

1. Double-click the C1SuperTooltip component in the Toolbox to add it to your form.
2. Select the control you would like associated with the C1SuperTooltip.
3. In the Properties window, click the **ellipsis** button next to the **Tooltip on C1SuperTooltip1** property. The **C1SuperTooltip Editor** appears.



At design time, there are two ways you can create the content of your Tooltip: either using the **Office** tab or using **Html** tab to manually enter your own HTML code. In this example, we will use the **Office** tab, but click the **Html** tab and enter your code there, if desired. When you use the **Office** tab, **C1SuperTooltip** automatically creates the HTML code behind the Tooltip for you.

4. To set up your Tooltip:
 - a. Enter a title for the Tooltip in the **Title** field. The title will appear to the right of the image, if any.
 - b. Click the drop-down arrow next to the **Image** property to find and select an image to appear next to the Tooltip title.

- c. Check the **Top Separator** checkbox if you want a divider line to appear after the title text, separating the title from the body text of the ToolTip.
- d. Enter the text for the C1SuperTooltip in the **Body Text** field.
- e. Check the **Bottom Separator** checkbox if you want a divider line to appear after the body text of the ToolTip, separating it from the subtitle.
- f. Enter a subtitle for the ToolTip in the Subtitle field. The subtitle will appear to the right of the subimage, if any.
- g. Click the drop-down arrow next to the **Subimage** property to find and select an image to appear next to the subtitle.

A preview of the ToolTip appears in the **Preview** window.

5. Select the **Properties** tab and set the desired properties for **C1SuperTooltip1**. You can do things such as: change ToolTip text and background color, add a background image, or modify how long and how quickly a ToolTip window is displayed. For a complete list of properties, see C1SuperTooltip Properties.

When you run the application and mouse over the control associated with a C1SuperTooltip, the ToolTip appears.



Creating a C1SuperTooltip Programmatically

The following topic explains how to create **C1SuperTooltips** in code. You can specify the C1SuperTooltip text and associate it with a control using the SetToolTip method.

In the SetToolTip method, specify the control or **Windows.Forms.ToolStripItem** to associate with the ToolTip first, and then add the string, or the HTML code used to create the ToolTip text.

The code for the SetToolTip method looks like this:

```
C1SuperTooltip1.SetToolTip(Control, String)
```

or

```
C1SuperTooltip1.SetToolTip(ToolStripItem, String)
```

Note: In the following examples an embedded resource containing an image is used. To embed a resource, select **Project | YourProjectName Properties**. Select **Add Resource** and choose to add an existing file, *NewDoc.png* in this example, or add a new one. Then, in the Solution Explorer, select the resource file and set **Build Action** to **Embedded Resource** in the Properties window.

To create a C1SuperTooltip for a control programmatically:

1. Add a C1SuperTooltip control and the control to associate it with to your form. In this example, we will use a Button1 control.
2. Add the following code to the **Form_Load** event. This code uses an embedded resource that contains an image, *NewDoc.png*, but any image can be used.

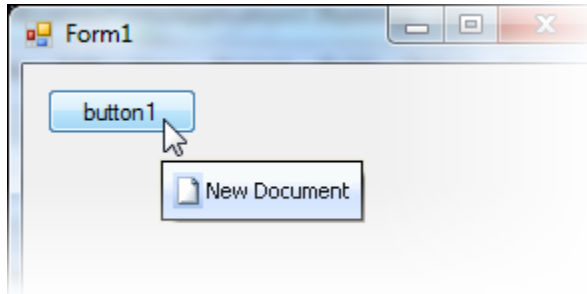
- Visual Basic

```
C1SuperTooltip1.SetToolTip(Button1, "<table><tr>" + _
"<td><img src='NewDoc.png'></td>" + _
"<td>New Document</td>" + "</tr></table>")
```

- C#

```
c1SuperTooltip1.SetToolTip(button1, "<table><tr>" +
"<td><img src='NewDoc.png'></td>" +
"<td>New Document</td>" + "</tr></table>");
```

3. Run your project. The code creates a C1SuperTooltip like the following.



To create a C1SuperTooltip for a ToolStripItem programmatically:

1. Add a C1SuperTooltip control and a **System.Windows.ToolStrip** control to your form. In this example, we have added buttons to the **ToolStrip**.
2. Add the following code to the **Form_Load** event. This code uses an embedded resource that contains an image, NewDoc.png, but any image can be used.

- Visual Basic

```
' hide the default ToolTip so only the C1SuperTooltip is visible
ToolStripButton1.AutoToolTip = False
```

```
C1SuperTooltip1.SetToolTip(ToolStripButton1, "<table><tr>" + _
"<td><img src='NewDoc.png'></td>" + _
"<td>New Document</td>" + "</tr></table>")
```

- C#

```
// hide the default ToolTip so only the C1SuperTooltip is visible
toolStripButton1.AutoToolTip = false;
```

```
c1SuperTooltip1.SetToolTip(toolStripButton1, "<table><tr>" +
"<td><img src='NewDoc.png'></td>" +
"<td>New Document</td>" + "</tr></table>");
```

3. Run your project. The code creates a C1SuperTooltip like the following:



For more information, see the `SetToolTip` method.

Creating a C1SuperTooltip using a Cascading Style Sheet

The following topic explains how to apply a cascading style sheet to your ToolTips for complete control over how and where they appear within your application. **SuperTooltip for WinForms** supports most HTML features, including cascading style sheets, which offer you greater control over how and where ToolTips appear in your applications. Simply create a cascading style sheet within your code, create your Tooltip, and apply the style sheet styles to the Tooltip.

In the following example, we will create a Microsoft Vista-style Tooltip identical to the one created using the **C1SuperTooltip Editor** in the [Creating C1SuperTooltips at Design Time](#) (page 36) topic, only this Tooltip will be created in code and use a cascading style sheet. The code in the following steps was placed within the **Form_Load** event.

1. Create the cascading style sheet.

- Visual Basic

```
Dim myCSS As String

'create the cascading style sheet
myCSS = "<style type='text/css'>" + _
".header{font-family: tahoma; font-weight: bold; margin-left: 2px; vertical-align:middle}" + _
".body{font-family: tahoma; margin-left: 8px}" + _
"img{vertical-align: middle}" + _
"td{vertical-align:middle}" + _
"p{border-bottom: medium solid #999999; border-bottom-width:1px}" + _
"</style>"
```

- C#

```
string myCSS;

//create the cascading style sheet
myCSS = "<style type='text/css'>" +
".header{font-family: tahoma; font-weight: bold; margin-left: 2px; vertical-align:middle}" +
".body{font-family: tahoma; margin-left: 8px}" +
"img{vertical-align: middle}" +
"td{vertical-align:middle}" +
"p{border-bottom: medium solid #999999; border-bottom-width:1px}" +
"</style>";
```

2. Create the header and body text of the Tooltip, and apply styles from the cascading style sheet.

- Visual Basic

```

Dim TipBuilder, TipBody, TipHeader As String

' create the header, or title, of the ToolTip
TipHeader = "<div class='header'>" + "Copy" + "</div>"

'create the body text of the ToolTip
TipBody = "<table width=160px>" + _
    "<tr>" + _
    "<td>" + _
    "<div class='body'>" + "Copy the selection and put" +
    "it<br>on the Clipboard." + "</div>" + _
    "</td>" + _
    "</tr>" + _
    "</table>" + _
    "<p></p>" + _
    "<table cellpadding=0>" + _
    "<tr>" + _
    "<td>" + _
    "<img src='HelpButton.png'>" + _
    "</td>" + _
    "<td>" + _
    "<div class='header'>" + _
    "Press F1 for help." + _
    "</div>" + _
    "</tr>" + _
    "</table>"

```

- C#

```

string TipBuilder, TipBody, TipHeader;

// create the header, or title, of the ToolTip
TipHeader = "<div class='header'>" + "Copy" + "</div>";

//create the body text of the ToolTip
TipBody = "<table width=160px>" +
    "<tr>" +
    "<td>" +
    "<div class='body'>" + "Copy the selection and put" +
    "it<br>on the Clipboard." + "</div>" +
    "</td>" +
    "</tr>" +
    "</table>" +
    "<p></p>" +
    "<table cellpadding=0>" +
    "<tr>" +
    "<td>" +
    "<img src='HelpButton.png'>" +
    "</td>" +
    "<td>" +
    "<div class='header'>" +
    "Press F1 for help." +
    "</div>" +
    "</tr>" +
    "</table>";

```

Note: In this example an embedded resource containing an image is used. To embed a resource, select **Project | YourProjectName Properties**. Select **Add Resource** and choose to add an existing file, *HelpButton.png* in this

example, or add a new one. Then, in the Solution Explorer, select the resource file and set **Build Action** to **Embedded Resource** in the Properties window.

3. Combine the separate parts of the ToolTip, and apply the cascading style sheet.

- Visual Basic

```
' Combine the ToolTip header and body, and apply the cascading style sheet.  
TipBuilder = myCSS + TipHeader + TipBody
```

- C#

```
// Combine the ToolTip header and body, and apply the cascading style sheet.  
TipBuilder = myCSS + TipHeader + TipBody;
```

4. Add the Vista formatting and associate the ToolTip with the button control.

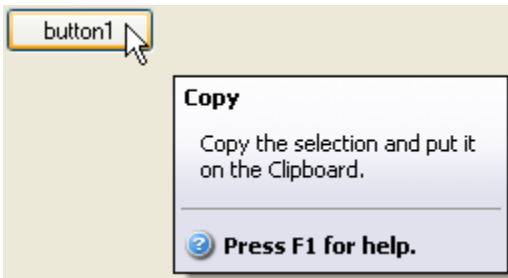
- Visual Basic

```
' apply the Vista background gradient and associate the ToolTip with Button1  
C1SuperTooltip1.BackgroundGradient =  
C1.Win.C1SuperTooltip.BackgroundGradient.Vista  
C1SuperTooltip1.SetToolTip(Button1, TipBuilder)
```

- C#

```
// apply the Vista background gradient and associate the ToolTip with Button1  
c1SuperTooltip1.BackgroundGradient =  
c1.Win.C1SuperTooltip.BackgroundGradient.Vista;  
c1SuperTooltip1.SetToolTip(button1, TipBuilder);
```

5. Run the project and mouse over the button associated with **C1SuperTooltip1**. The Vista-style ToolTip appears.



Adding a C1SuperTooltip using HTML

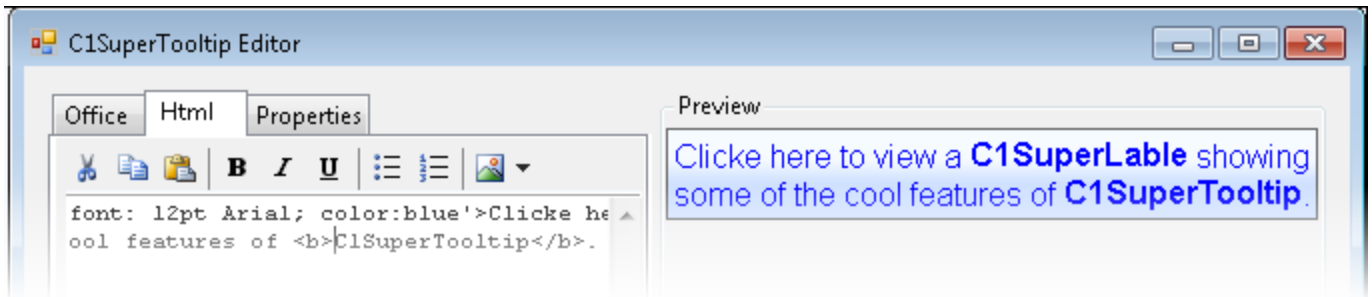
If you choose to create a ToolTip by using all of your own HTML code, you can enter it on the **Html** tab of the **C1SuperTooltip Editor**.

To add a **C1SuperTooltip** using HTML, complete the following steps:

1. In your project, select the control for which you are creating a C1SuperTooltip.
2. In the Properties window, click the **ellipsis** button next to the **ToolTip on C1SuperTooltip1** property for the control to open the **C1SuperTooltip Editor**.
3. Select the **Html** tab.
4. Enter the following HTML code in the **Html** text box:

```
<span style='font: 12pt Arial; color:blue'>Click here to view a <b>C1SuperLabel</b>  
showing<br> some of the cool features of <b>C1SuperTooltip</b>.</span>
```

A preview of the C1SuperTooltip appears in the **Preview** window.



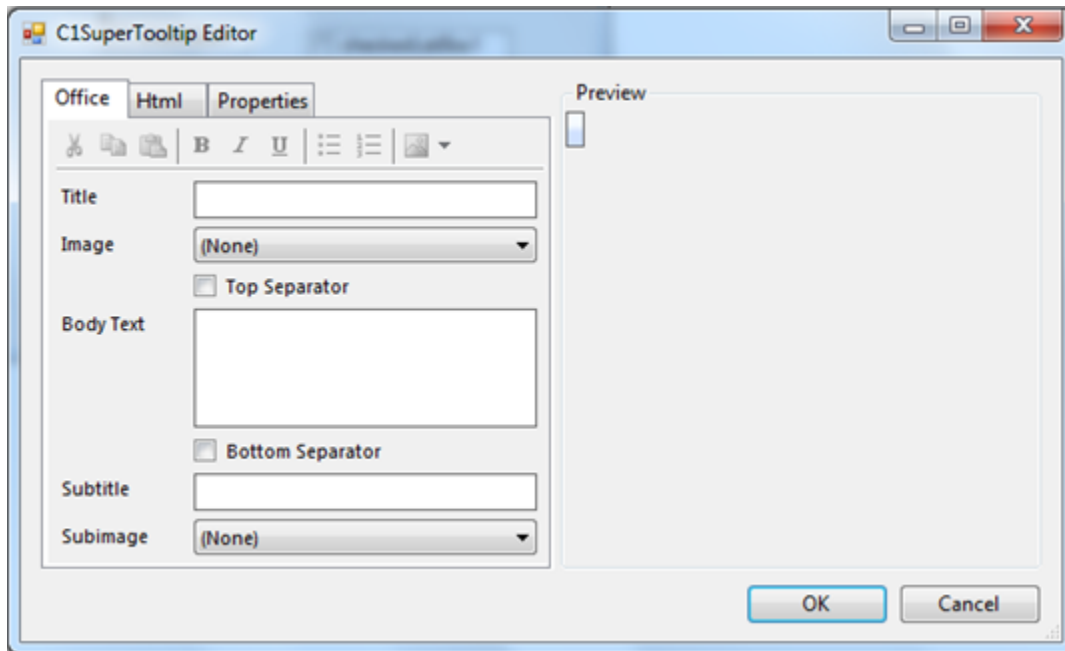
5. Click the **Properties** tab. Notice the ForeColor property is set to **InfoText** and the Font property is set to **Tahoma, 8pt**, by default, but the preview of the Tooltip shows the forecolor as blue and the font as Arial, 12pt. This is because when `` tags are used to format the text in your HTML code, they cannot be overwritten by the properties specified in the **Properties** tab of the editor.
6. Click **OK** to close the editor.

Adding Multiple C1SuperTooltips

You can also add multiple ToolTips to a project. You may want to do this if you are not formatting the ToolTips with your own HTML code and you want the appearance properties of the ToolTips to be different. You could also do this when you want the behavior properties of the ToolTips to be different, regardless of how you created them.

When you add more than one Tooltip, make sure you specify the correct Tooltip for the control with which you want it to be associated. If you specify multiple ToolTips for one control, all specified ToolTips appear when you run the project and mouse over the control.

1. Suppose you have a project with two buttons on the form. Add two C1SuperTooltip controls.
2. Select **Button1** and click the **ellipsis** button next to the **Tooltip on C1SuperTooltip1** property in the Properties window. The **C1SuperTooltip Editor** appears.



3. Create your ToolTip using the **Office** tab or by adding HTML code on the **Html** tab.
4. Set some of the properties on the **Properties** tab and click **OK**.
5. Select **Button2** and click the **ellipsis** button next to the **ToolTip on C1SuperTooltip2** property in the Properties window.
6. In the **C1SuperTooltip Editor**, create the second ToolTip and click the **Properties** tab.
7. Set some properties using different values than you used for **C1SuperTooltip1**.
8. Click **OK** to close the editor. The **ToolTip on C1SuperTooltip1** property will remain empty for **Button2**.
9. Run the project and mouse over each button. Notice how the **C1SuperTooltip1** appears when you mouse over **Button1**, and **C1SuperTooltip2** appears when you mouse over **Button2**. The styles and behaviors will be different, depending on the settings you used.

Changing the C1SuperTooltip Appearance and Behavior Settings

You can change the appearance and behavior settings of your ToolTips two different ways: using the **C1SuperTooltip Editor** or using the C1SuperTooltip control, through its smart tag and the Properties window.

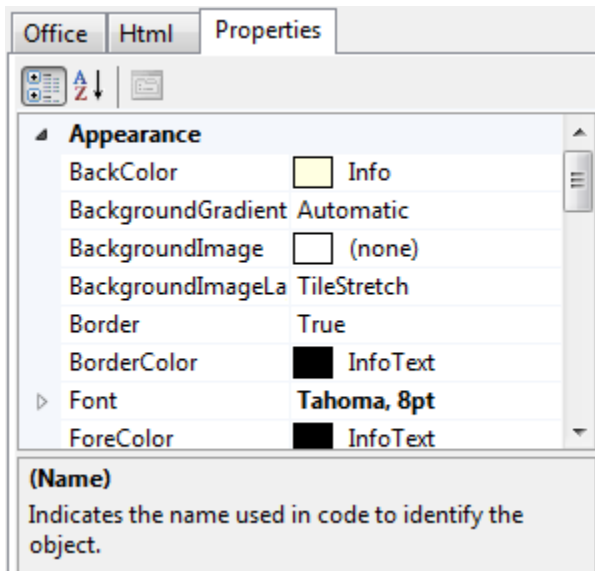
Changing the settings in the C1SuperTooltip Editor

In the **C1SuperTooltip Editor**, you can use the **Office** tab to add images, a title, a subtitle, and the body text of the ToolTip. C1SuperTooltip automatically creates all of the HTML code behind the ToolTip, saving you time and work. You can create the same ToolTip by entering all of your own HTML code on the **Html** tab of the editor if you choose not to have the editor do it for you. When using these two tabs, the changes you make and the settings you specify are applied only to the ToolTip for the control you have selected in your form.

The **Properties** tab, however, allows you to change the overall appearance and behavior of the ToolTip, which will be applied to all controls associated with it.

1. In your project, select the control for which you are creating a C1SuperTooltip.
2. In the Properties window, click the **ellipsis** button next to the **ToolTip on C1SuperTooltip1** property for the control to open the **C1SuperTooltip Editor**.

3. Select the **Properties** tab.



4. Set the following properties:
 - Click the drop-down arrow next to the BackColor property, select the **Web** tab, and choose a color.
 - Set the BackgroundGradient property to **None**. The background color will not appear if this property is set to a value other than **None**.
 - Click the drop-down arrow next to the ForeColor property, select the **Web** tab, and choose a color.
 - Set the Shadow property to **False**.
 - Set the Border property to **False**.
 - Expand the Font property node and set the **Size** to **14**.
5. To change the amount of time, in milliseconds, the ToolTip remains visible when the mouse hovers over each button, set the AutoPopDelay property to **1000**.
6. Run the project and mouse over the control with the associated ToolTip. The ToolTip will appear something like the following image, depending on the settings you used.

[Click here to view a C1SuperLabel showing some of the cool features of C1SuperTooltip.](#)

Changing the settings using the C1SuperTooltip Control

You can use the C1SuperTooltip smart tag to determine the background gradient for the ToolTip and whether it appears within a balloon shape or a rectangular box. For more information, see [C1SuperTooltip Tasks and Context Menus](#) (page 23).

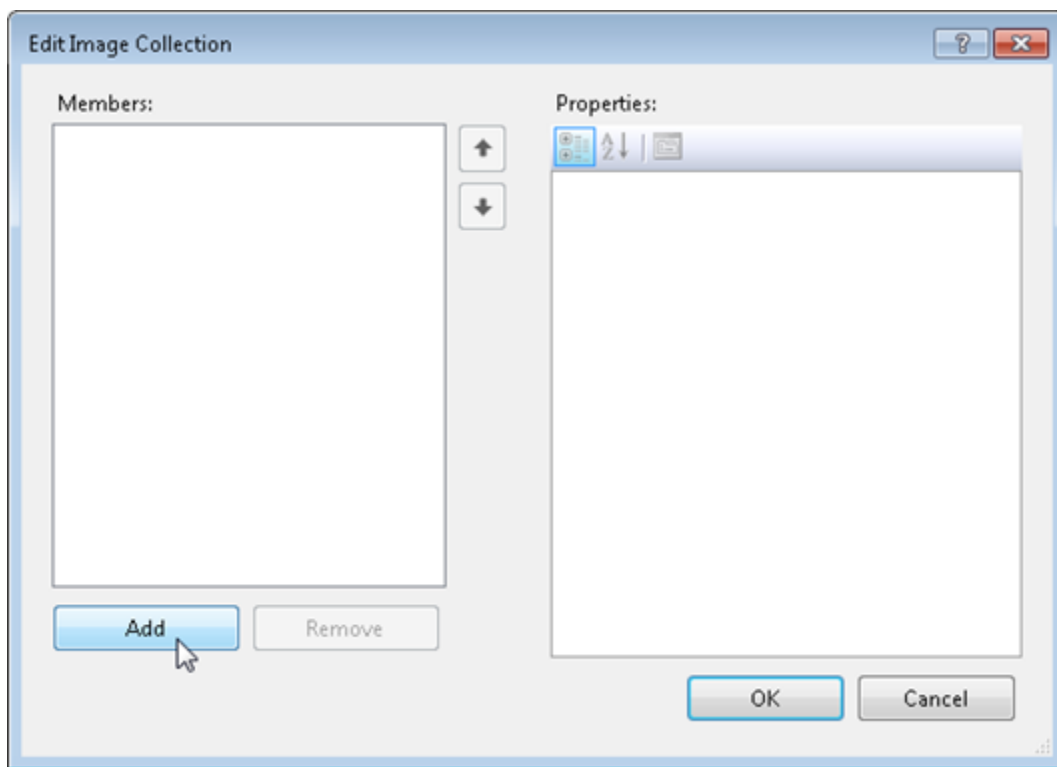
You can use the Visual Studio Properties window to change all of the C1SuperTooltip properties. This is the same list of properties you will find on the **Properties** tab of the **C1SuperTooltip Editor**.

1. Right-click the C1SuperTooltip control and select **Properties** to access the Properties window.
2. Set the desired properties.

Adding an Image to C1SuperTooltip

SuperTooltip for WinForms supports adding images, including animated images, at design time. First, add the image to the C1SuperTooltip Image collection, and then specify the image in the **C1SuperTooltip Editor**.

1. Add a C1SuperTooltip control to your form.
2. Select the C1SuperTooltip control and click the **ellipsis** button next to the Images property in the Properties window. The **Edit Image Collection** editor appears.



3. Click the **Add** button and browse to find the image you want to use.
4. Select the image and click **Open**. The image is added to the ToolTip's image collection.
5. Click **OK** to close the **Edit Image Collection** editor.
6. Select the control being associated with the C1SuperTooltip.
7. Click the **ellipsis** button next to the **ToolTip on C1SuperTooltip1** property.

The image you added to the ToolTip's image collection can be selected from the **Image** or **Subimage** drop-down list on the **Office** tab. If you are using your own HTML code, click the **Html** tab and reference the image in your code like this:

```
<img src= "res://mybitmap.png" />
```

Creating C1SuperLabels

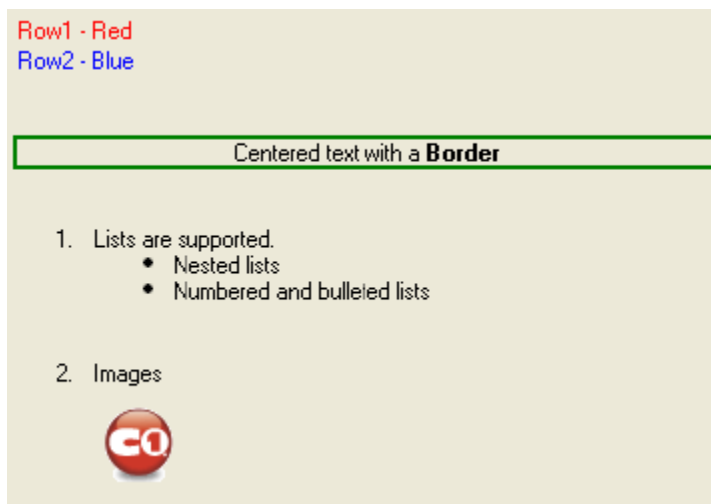
The following topics explain how to create **C1SuperLabels** at design time using the **C1SuperLabel Editor** and programmatically in code, as well as how to add images labels.

Creating C1SuperLabels at Design Time

SuperTooltip for WinForms provides a design-time editor, the **C1SuperLabel Editor**, to simplify the process of creating labels in your applications. This topic will show you how to create labels and change their appearance and behavior using this editor. The C1SuperLabel component is very similar to the **Label** control, except it can display HTML content instead of just plain text. You can display an HTML page including tables, images, lists or preformatted text, for example, right within the label.

To add C1SuperLabel to your application:

1. Double-click the C1SuperLabel component in the Toolbox to add it to your form.
2. Click the **ellipsis** button next to the Text property. The **C1SuperLabel Editor** appears.
3. Enter your HTML code.
4. Run the application and the rendered HTML appears within the C1SuperLabel.



Creating a C1SuperLabel Programmatically

You can specify the C1SuperLabel text and associate it with a control using the Text property. All you need to do is add the text as a string. You can add plain text or HTML code.

To create a C1SuperLabel programmatically:

1. Add a C1SuperLabel to your form.
 2. Add the following code to the **Form_Load** event. It adds a table containing two rows with two graphics and some text in the label:
- Visual Basic

```
'add two rows to the C1SuperLabel
C1SuperLabel1.Text = _
    "<table>" + _
    "<tr>" + _
        "<td><img src='search.png'>" + _
        "<td>This is the second cell in the top row" + _
    "<tr>" + _
        "<td><img src='up.png'>" + _
        "<td>This is the second cell in the bottom row." + _
    "</table>"

' automatically resize the label to show all contents
```

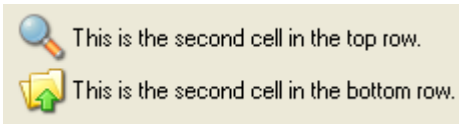
```
C1SuperLabel1.AutoSize = True
```

- C#

```
// add two rows to the C1SuperLabel
c1SuperLabel1.Text =
    "<table>" +
    "<tr>" +
    "    <td><img src='search.png'>" +
    "    <td>This is the second cell in the top row" +
    "</tr>" +
    "<tr>" +
    "    <td><img src='up.png'>" +
    "    <td>This is the second cell in the bottom row." +
    "</table>";

// automatically resize the label to show all contents
c1SuperLabel1.AutoSize = true;
```

3. Run your project. The code creates a C1SuperLabel like the following:



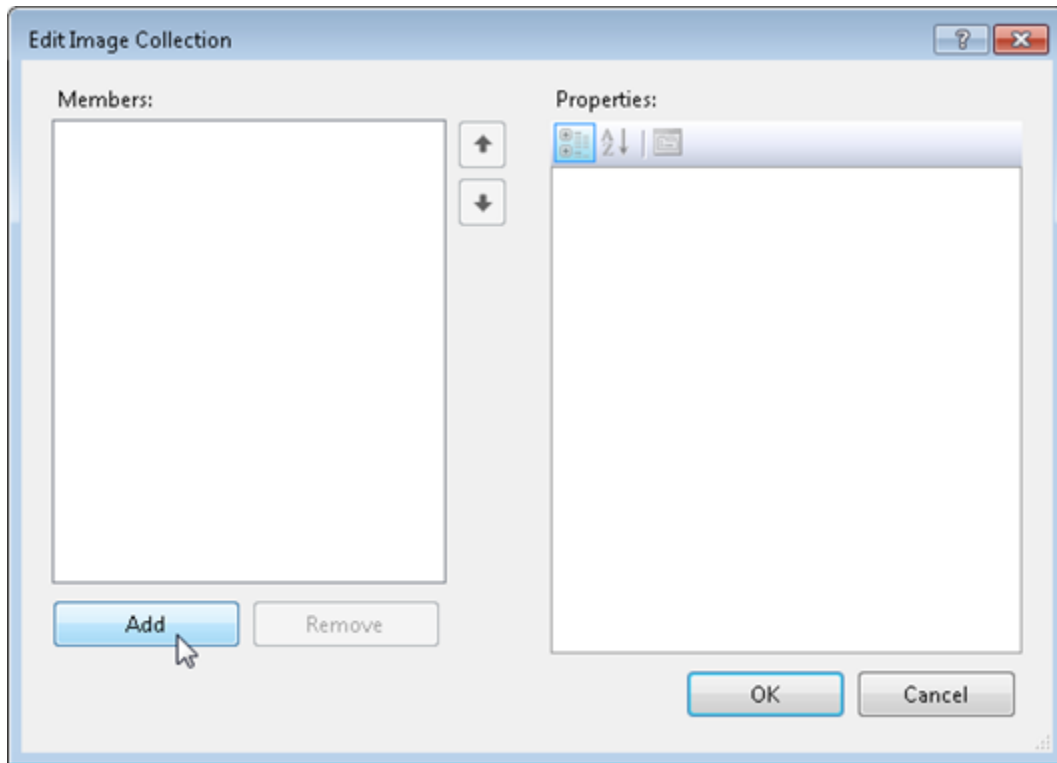
Note: In this example embedded resources containing images are used: *search.png* and *up.png*. To embed a resource, select **Project | YourProjectName Properties**. Select **Add Resource** and choose to add an existing file or add a new one. Then, in the Solution Explorer, select the resource file and set **Build Action** to **Embedded Resource** in the Properties window.

For more information, see the Text property.

Adding an Image to C1SuperLabel

SuperTooltip for WinForms supports adding images, including animated images, at design time. First, add the image to the C1SuperTooltip Image collection, and then specify the image in the **C1SuperTooltip Editor**.

1. Add a C1SuperLabel control to your form.
2. Select the C1SuperLabel control and click the **ellipsis** button next to the Images property in the Properties window. The **Edit Image Collection** editor appears.



3. Click the **Add** button and browse to find the image you want to use.
4. Select the image and click **Open**. The image is added to the ToolTip's image collection.
5. Click **OK** to close the **Edit Image Collection** editor.
6. Click the **ellipsis** button next to the Text property.

When you enter your HTML code, the image you added to the label's image collection can be referenced in the code like this:

```
<img src= "res://mybitmap.png" />
```

Creating C1SuperErrorProvider Error Messages

The following topics explain how to create an error message, change the error message icon, use the C1SuperErrorProvider control with a data source, and more.

Creating an Error Message

You can create an HTML formatted error message to pop up for a control. In this example, we'll add an error message for a text box named **txtCountry**.

1. Add a C1SuperErrorProvider control to your form. A C1SuperTooltip is automatically added to the form and connected with the C1SuperErrorProvider control.
2. Select **Code** in the Visual Studio **View** menu.
3. Add the following code to the **Form_Load** event:

- Visual Basic

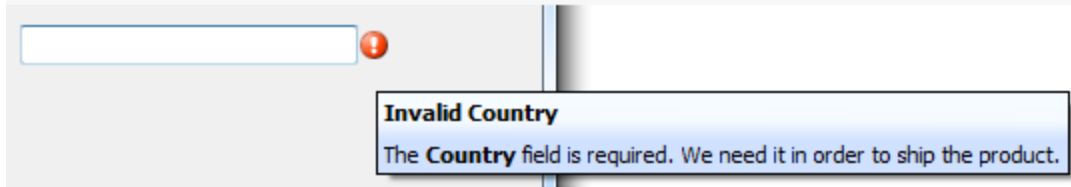
```
C1SuperErrorProvider1.SetError(txtCountry,  
"<b>Invalid Country</b><br/>" & "
```

```
<p>The <b>Country</b> field is required. We need it in order to ship the product.</p>"
```


- C#

```
c1SuperErrorProvider1.SetError(txtCountry, "Invalid Country<br/>" + "
```

When you run the project and mouse over the error message icon, the error message looks like the following example:



Changing the Error Message Icon

When you create an error message with C1SuperErrorProvider, a default warning icon  is used. You can change this to any icon file you like using the Icon property. Assuming you have a C1SuperErrorProvider control on your form, follow these steps to change the error message icon:

1. Right-click the C1SuperErrorProvider control and select **Properties** to open the Visual Studio Properties window.
2. Click the **ellipsis** button next to the Icon property.
3. Choose an icon file (*.ico) and click **Open**.

When you run your project, notice the new icon image used for the error message.



Changing the Error Message Blink Style

When you create an error message with C1SuperErrorProvider, by default, the error message icon blinks. You have the option of making it blink sometimes, always, or never. It stops blinking when you click on it. To specify the blink style, follow these steps:

1. Add a C1SuperErrorProvider control to your form.
2. Click the C1SuperErrorProvider smart tag to open the **Tasks** menu.
3. Click the drop-down arrow next to the BlinkStyle property and select **BlinkIfDifferentError**, **AlwaysBlink**, or **NeverBlink**.

Showing an Image when the Error Icon is Hovered

You can use the ImageHot property to provide feedback in the form of an image when the mouse pointer hovers over the error icon.

To show an image when the error icon is hovered, follow these steps:

1. Add a C1SuperErrorProvider control to your form. A C1SuperTooltip is automatically added to the form and connected with the C1SuperErrorProvider control.
2. Select **Code** in the Visual Studio **View** menu.
3. Add the following code to the **Form_Load** event:

- Visual Basic

```
C1SuperErrorProvider1.ImageHot =  
System.Drawing.Image.FromFile("c:\\MyFiles\\Level1Warning.bmp")
```

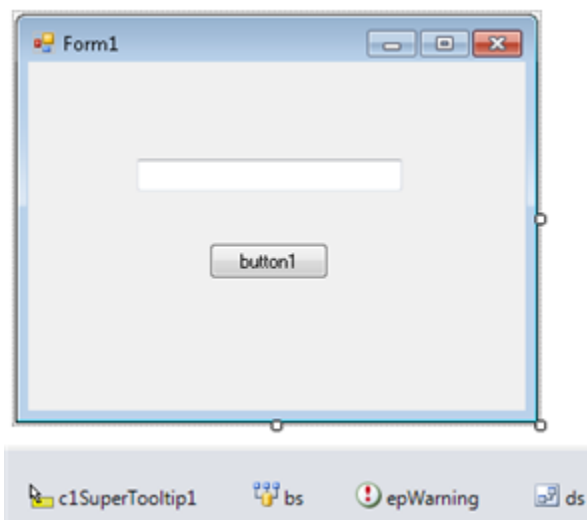
- C#

```
c1SuperErrorProvider1.ImageHot =  
System.Drawing.Image.FromFile("c:\\MyFiles\\Level1Warning.bmp");
```

Using C1SuperErrorProvider with Data Sources

Use the C1SuperErrorProvider control with a data source to indicate an error to users. The C1SuperErrorProvider must be associated with a C1SuperTooltip in order to appear, although the tooltip can be blank. When you add a C1SuperErrorProvider control to your form, a C1SuperTooltip is automatically added and connected with it.

This topic assumes you have a button, text box, and data source on your form, similar to the following image:



1. Add a C1SuperErrorProvider control to your form.
2. Click the C1SuperErrorProvider smart tag to open the **Tasks** menu.
3. Click the drop-down arrow next to **Choose DataSource** and select the data source on your form.
4. Use the **DataRow.SetColumnError** method to associate an error message with the given data field:

- Visual Basic

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles Button1.Click
```

```

        DirectCast(bs.Current,
DataRowView).Row.SetColumnError("LastName", "Here is the <b>warning</b>
message!")
End Sub

```

- C#

```

private void button1_Click(object sender, EventArgs e)
{
    ((DataRowView)bs.Current).Row.SetColumnError("LastName",
        "Here is the <b>warning</b> message!");
}

```

5. Add the necessary code to the **Form_Load** event to bind the text box to the data source. In this example, the text box is bound to an .xml file.

- Visual Basic

```

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    ds.DataSetName = "AuthorsDataSet"
    ds.ReadXml("../..\authors.xml", System.Data.XmlReadMode.Auto)
    bs.DataMember = "authors"

    TextBox1.DataBindings.Add(New Binding("Text", bs, "LastName"))
End Sub

```

- C#

```

private void Form1_Load(object sender, EventArgs e)
{
    ds.DataSetName = "AuthorsDataSet";
    ds.ReadXml(@"..\..\authors.xml",
System.Data.XmlReadMode.Auto);
    bs.DataMember = "authors";

    textBox1.DataBindings.Add(new Binding("Text", bs,
"LastName"));
}

```

6. Press F5 to run the project and then click the button. The warning icon appears, and if you mouse over it, the message appears.

