ComponentOne

HeaderContent for ASP.NET AJAX

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)

This manual was produced using ComponentOne Doc-To-Help™.

Table of Contents

ComponentOne HeaderContent for ASP.NET AJAX Overview

ComponentOne HeaderContent for ASP.NET AJAX allows you to easily layout and display blocks of information. **ComponentOne HeaderContent for ASP.NET AJAX** includes one control, **C1HeaderContentControl**, which includes a header area where you can include title and context, and a content area which can contain text, images, controls, external Web sites, and more. With CSS styling and built-in Office 2007 and Vista themes, you can easily create sophisticated Web applications with **ComponentOne HeaderContent for ASP.NET AJAX**.

Getting Started

Get started with the following topics:

- Key Features (page 16)
- Quick Start (page 19)
- Samples (page 33)
- Task-Based Help (page 33)

Installing HeaderContent for ASP.NET AJAX

The following sections provide helpful information on installing **ComponentOne Studio for ASP.NET AJAX**:

HeaderContent for ASP.NET AJAX Setup Files

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

Bin	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET package.
C1WebUi	Contains files (at least a readme.txt) related to the product.
C1WebUi\VisualStyles	Contains all external file themes.

The **ComponentOne Studio for ASP.NET AJAX Help Setup** program installs integrated Microsoft Help 2.0 and Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for ASP.NET directory in the following folders:

H2Help	Contains Microsoft Help 2.0 integrated documentation for all Studio components.
HelpViewer	Contains Microsoft Help Viewer Visual Studio 2010 integrated documentation for all Studio components.

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\<username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<username>\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.
C1WebUi	Contains samples and tutorials for **HeaderContent for ASP.NET AJAX**.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for ASP.NET | Samples | Palomino Samples**.

System Requirements

System requirements for **ComponentOne Studio for ASP.NET AJAX** components include the following:

Operating Systems:	Windows 2000
	Windows Server® 2003
	Windows Server® 2008
	Windows XP SP2
	Windows Vista®
	Windows 7
Web Server:	Microsoft Internet Information Services (IIS) 5.0 or later
Environments:	.NET Framework 2.0 or later
	Visual Studio 2005 or later
	Internet Explorer® 6.0 or later
	Firefox® 2.0 or later
	Safari® 2.0 or later
Disc Drive:	CD or DVD-ROM drive if installing from CD

Note: HeaderContent for ASP.NET AJAX requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1HeaderContentControl control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 13). For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Uninstalling HeaderContent for ASP.NET AJAX

To uninstall **Studio for ASP.NET AJAX**:

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

To uninstall **Studio for ASP.NET AJAX** 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 ASP.NET AJAX Help** and click the **Remove** button.
3. Click **Yes** to remove the integrated help.

Deploying your Application in a Medium Trust Environment

Depending on your hosting choice, you may need to deploy your Web site or application in a medium trust environment. Often in a shared hosting environment, medium trust is required. In a medium trust environment several permissions are unavailable or limited, including OleDbPermission, ReflectionPermission,and FileIOPermission. You can configure your Web.config file to enable these permissions.

Note: ComponentOne controls will not work in an environment where reflection is not allowed.

ComponentOne ASP.NET controls include the AllowPartiallyTrustedCallers() assembly attribute and will work under the medium trust level with some changes to the Web.config file. Since this requires some control over the Web.config file, please check with your particular host to determine if they can provide the rights to override these security settings.

Modifying or Editing the Config File

In order to add permissions, you can edit the exiting web_mediumtrust.config file or create a custom policy file based on the medium trust policy. If you modify the existing web_mediumtrust.config file, all Web applications will have the same permissions with the permissions you have added. If you want applications to have different permissions, you can instead create a custom policy based on medium trust.

Edit the Config File

In order to add permissions, you can edit the exiting web_mediumtrust.config file. To edit the exiting web_mediumtrust.config file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Open the web_mediumtrust.config file.
3. Add the permissions that you want to grant. For examples, see Adding Permissions (page 4).

Create a Custom Policy Based on Medium Trust

In order to add permissions, you can create a custom policy file based on the medium trust policy. To create a custom policy file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Copy the web_mediumtrust.config file and create a new policy file in the same directory.

   Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.
3. Add the permissions that you want to grant. For examples, see Adding Permissions (page 4).
4. Enable the custom policy file on your application by modifying the following lines in your web.config file under the `<system.web>` node:

```
<system.web>
<trust level="CustomMedium" originUrl=""/>

 <securityPolicy>
               <trustLevel name="CustomMedium"
policyFile="AllowReflection_Web_MediumTrust.config"/>
```

```
        </securityPolicy>
        ...
</system.web>
```

Note: Your host may not allow trust level overrides. Please check with your host to see if you have these rights.

Allowing Deserialization

To allow the deserialization of the license added to App_Licenses.dll by the Microsoft IDE, you should add the SerializationFormatter flag to security permission to the Web.config file. Complete the steps in the Modifying or Editing the Config File (page 3) topic to create or modify a policy file before completing the following.

Add the `SerializationFormatter` flag to the `<IPermission class="SecurityPermission">` tag so that it appears similar to the following:

```
<NamedPermissionSets>
     <PermissionSet
     class="NamedPermissionSet"
     version="1"
     Name="ASP.Net">
        <IPermission
                class="SecurityPermission"
                version="1"
                Flags="Assertion, Execution, ControlThread,
ControlPrincipal, RemotingConfiguration, SerializationFormatter"/>
        ...
     </PermissionSet>
</NamedPermissionSets>
```

Adding Permissions

You can add permission, including ReflectionPermission, OleDbPermission, and FileIOPermission to the web.config file. Note that ComponentOne controls will not work in an environment where reflection is not allowed. Complete the steps in the Modifying or Editing the Config File (page 3) topic to create or modify a policy file before completing the following.

ReflectionPermission

By default ReflectionPermission is not available in a medium trust environment. ComponentOne ASP.NET controls require reflection permission because LicenseManager.Validate() causes a link demand for full trust.

To add reflection permission, complete the following:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
    <SecurityClass Name="ReflectionPermission"
Description="System.Security.Permissions.ReflectionPermission,
mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
     <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
        <IPermission
```

```
            class="ReflectionPermission"
            version="1"
            Flags="ReflectionEmit,MemberAccess" />
        ...
    </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web_mediumtrust.config file.

OleDbPermission

By default OleDbPermission is not available in a medium trust environment. This means you cannot use the ADO.NET managed OLE DB data provider to access databases. If you wish to use the ADO.NET managed OLE DB data provider to access databases, you must modify the web_mediumtrust.config file.

To add OleDbPermission, complete the following steps:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
    <SecurityClass Name="OleDbPermission"
Description="System.Data.OleDb.OleDbPermission, System.Data,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
    <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
        <IPermission class="OleDbPermission" version="1"
Unrestricted="true"/>
        ...
    </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web_mediumtrust.config file.

FileIOPermission

By default, FileIOPermission is not available in a medium trust environment. This means no file access is permitted outside of the application's virtual directory hierarchy. If you wish to allow additional file permissions, you must modify the web_mediumtrust.config file.

To modify FileIOPermission to allow read access to a specific directory outside of the application's virtual directory hierarchy, complete the following steps:

1. Open the web_mediumtrust.config file or a file created based on the web_mediumtrust.config file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
    <SecurityClass Name="FileIOPermission"
Description="System.Security.Permissions.FileIOPermission, mscorlib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
    <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
    ...
    <IPermission class="FileIOPermission" version="1"
Read="C:\SomeDir;$AppDir$" Write="$AppDir$" Append="$AppDir$"
PathDiscovery="$AppDir$" />
  ...
</PermissionSet>
</NamedPermissionSets>
```

4. Save and close the web_mediumtrust.config file.

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 .NET 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 box.)

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 Properties 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 application 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 dialog boxes. 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 project and go to the Solution Explorer window.
2. Click the **Show All Files** button on the top of the window.
3. Find the licenses.licx file and open it. If prompted, continue to open the file.
4. Change the version number of each component to the appropriate value. If the component does not appear in the file, obtain the appropriate data from another licenses.licx file or follow the alternate procedure following.
5. Save the file, then close the licenses.licx tab.
6. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

Alternatively, follow these steps:

1. Open the project and go to the Solution Explorer window.

2. Click the **Show All Files** button on the top of the window.
3. Find the licenses.licx file and delete it.
4. Close the project and reopen it.
5. Open the main form and add an instance of each licensed control.
6. Check the Solution Explorer window, there should be a licenses.licx file there.
7. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

For ASP.NET 2.x applications, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Find the licenses.licx file and right-click it.
3. Select the **Rebuild Licenses** option (this will rebuild the App_Licenses.licx file).
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

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 Studio for ASP.NET AJAX 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.Web.UI.2.dll
- C1.Web.UI.Controls.2.dll
- C1.Web.UI.3.dll
- C1.Web.UI.Controls.3.dll
- C1.Web.UI.4.dll
- C1.Web.UI.Controls.4.dll

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

About This Documentation

Acknowledgements

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 general namespace for ComponentOne Web products is **C1.Web.UI.Controls**The following code fragment shows how to declare a **C1TreeView** (which is one of the core **Studio for ASP.NET AJAX** classes) using the fully qualified name for this class:

- Visual Basic
```
Dim TreeView As C1.Web.UI.Controls.C1TreeView
```
- C#
```
C1.Web.UI.Controls.C1TreeView TreeView;
```

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*.

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 C1TreeView = C1.Web.UI.Controls.C1TreeView
Imports MyTreeView = MyProject.Objects.C1TreeView

Dim wm1 As C1TreeView
Dim wm2 As MyTreeViewMenu
```
- C#

```
using C1TreeView = C1.Web.UI.Controls.C1TreeView;
using MyTreeView = MyProject.Objects.C1TreeView;

 C1TreeView wm1;
 MyTreeView wm2;
```

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 an AJAX-Enabled ASP.NET Project

ComponentOne HeaderContent for ASP.NET AJAX requires you to create an ASP.NET AJAX-Enabled project so that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the C1HeaderContentControl control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

When creating AJAX-Enabled ASP.NET projects, Visual Studios 2008 and 2005 both give you the option of creating a Web site project or a Web application project. MSDN provides detailed information on why you would choose one option over the other.

If you are using Visual Studio 2008 with .NET Framework 2.0 or .NET Framework 3.0 or if you are using Visual Studio 2005, you must install the ASP.NET AJAX Extensions 1.0, which can be found at http://ajax.asp.net/. Additionally for Visual Studio 2005 users, creating a Web application project requires installation of a Visual Studio 2005 update and add-in, which can be found at http://msdn.microsoft.com/; however, if you have Visual Studio 2005 SP1, Web application project support is included and a separate download is not required.

If you are using Visual Studio 2008 and .NET Framework 3.5, you can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.

Note: If you are using Visual Studio 2010, see http://www.asp.net/ajax/ for more information on creating an AJAX-Enabled ASP.NET Project.

The following table summarizes the installations needed:

Visual Studio Version	Additional Installation Requirements
Visual Studio 2008, .NET Framework 3.5	None
Visual Studio 2008 and .NET Framework 2.0 or 3.0 Visual Studio 2005 Service Pack 1	ASP.NET AJAX Extensions 1.0 http://www.asp.net/ajax/downloads/archive/
Visual Studio 2005	ASP.NET AJAX Extensions 1.0 Visual Studio update and add-in (2 installs for Web application project support)

The following topics explain how to create both types of projects in Visual Studio 2008 and 2005.

- Creating an AJAX-Enabled Web Site Project in Visual Studio 2008

  To create a Web site project in Visual Studio 2008, complete the following steps:

  a. From the File menu, select **New** | Web Site. The New Web Site dialog box opens.

  b. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.

  c. In the list of templates, select **AJAX 1.0-Enabled ASP.NET 2.0 Web Site**.

d. Click **Browse** to specify a location and then click **OK**.

> **Note:** The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new AJAX-Enabled Web Site is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Application Project in Visual Studio 2008

To create a new Web application project in Visual Studio 2008, complete the following steps.

a. From the **File** menu, select **New** | Project. The New Project dialog box opens.

b. Select .NET Framework 3.5 or the desired framework in the upper right corner. Note that if you choose .NET Framework 2.0 or 3.0, you must install the extensions first.

c. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.

d. Select **AJAX 1.0-Enabled ASP.NET 2.0 Web Application** from the list of **Templates** in the right pane.

e. Enter a URL for your application in the **Location** field and click **OK**.

> **Note:** The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Site Project in Visual Studio 2005

To create a Web site project in Visual Studio 2005, complete the following steps:

a. From the **File** menu in Microsoft Visual Studio .NET, select **New Web Site**. The **New Web Site** dialog box opens.

b. Select **ASP.NET AJAX-Enabled Web Site** from the list of Templates.

c. Enter a URL for your site in the **Location** field and click **OK**.

> **Note:** The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

- Creating an AJAX-Enabled Web Application Project in Visual Studio 2005

  To create a new Web application project in Visual Studio 2005, complete the following steps.

  a. From the **File** menu in Microsoft Visual Studio 2005, select **New Project**. The **New Project** dialog box opens.

  b. Under **Project Types**, choose either **Visual Basic Projects** or **Visual C# Projects**. Note that one of these options may be located under **Other Languages**.

  c. Select **ASP.NET AJAX-Enabled Web Application** from the list of **Templates** in the right pane.

  d. Enter a URL for your application in the **Location** field and click **OK**.

  Note: The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

  A new Web Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called Default.aspx is displayed and a **ScriptManager** control is placed on the form. The **ScriptManger** is needed to enable certain features of ASP.NET AJAX such as partial-page rendering, client-script functionality of the Microsoft AJAX Library, and Web-service calls.

Adding the HeaderContent for ASP.NET AJAX Component to a Project

When you install **ComponentOne Studio for ASP.NET AJAX**, the **Create a ComponentOne Visual Studio Toolbox Tab** check box is checked, by default, in the installation wizard. When you open Visual Studio 2005, you will notice a **ComponentOne Studio for ASP.NET AJAX Projects** tab containing the ComponentOne controls that have automatically been added to the Toolbox.

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

Manually Adding the Studio for ASP.NET AJAX controls to the Toolbox

When you install **ComponentOne Studio for ASP.NET AJAX**, the following **HeaderContent for ASP.NET AJAX** component will appear in the Visual Studio Toolbox customization dialog box:

- C1HeaderContentControl

To manually add the **Studio for ASP.NET AJAX** controls 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 the **Studio for ASP.NET AJAX** components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **Studio for ASP.NET AJAX**, for example.
3. Right-click the tab where the component is 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.Web.UI.Controls.2.dll. Note that there may be more than one component for each namespace.
5. Click **OK** to close the dialog box.

   The controls are added to the Visual Studio Toolbox.

Adding Studio for ASP.NET AJAX Controls to the Form

To add **Studio for ASP.NET AJAX** controls to a form:

1. Add them to the Visual Studio Toolbox.

2. Double-click each control or drag it onto your form.

Note: HeaderContent for ASP.NET AJAX requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1HeaderContentControl control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 13). For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Adding a Reference to the Assembly

To add a reference to the C1.Web. UI.Conttrols.2 assembly:

1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the Project menu of your Web Application project.
2. Select the most recent version of the **ComponentOne Studio for ASP.NET AJAX** assembly from the list on the **NET** tab or browse to find the C1.Web.UI.Controls.2.dll file and click **OK**.
3. Select the **Form1.vb** tab or go to **View|Code** to open the Code Editor. At the top of the file, add the following **Imports** directive (**using** in C#):

```
Imports C1.Web.UI.Controls
```

Note: This makes the objects defined in the **C1.Web.UI.Controls.2** assembly visible to the project. See Namespaces (page 12) for more information.

Key Features

C1HeaderContentControl includes several unique features, including the following:

- **CSS Styling**

  C1HeaderContentControl includes CSS supported styling so that you can use cascading style sheets to easily style the **C1HeaderContentControl** control to match the design of your current Web site.

- **Customized Content**

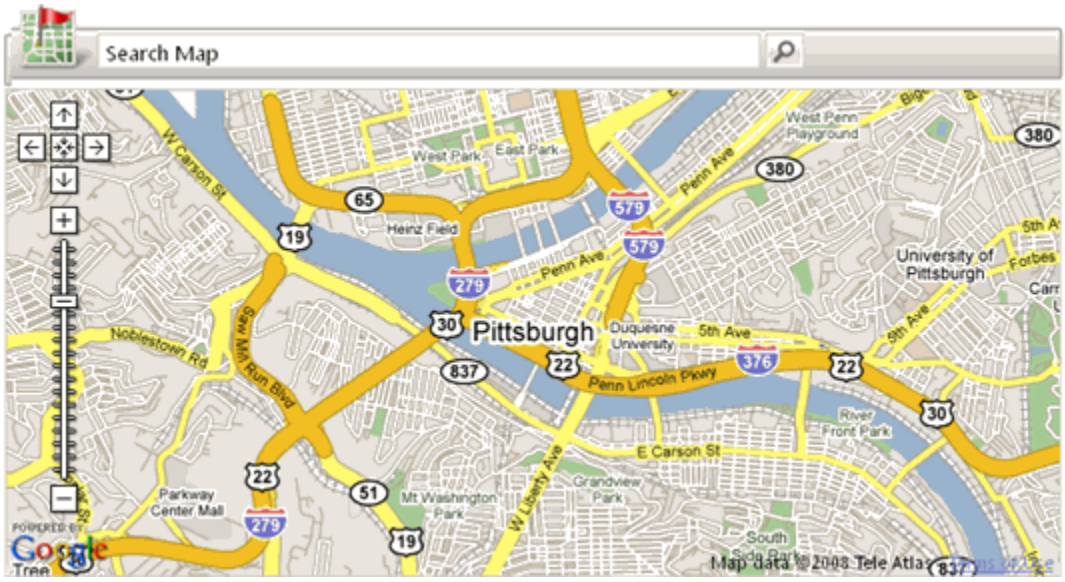
  You can add images, text, and controls – standard and third-party controls – to pages in the **C1HeaderContentControl** control.

- **Template Support**

  You can add templates to the content area of the dialog window through the Content property. Dynamic Templates can be used in the content area of the dialog window for achieving rich presentation of the **C1HeaderContentControl**. See Header and Content Templates (page 31) for more information.

- **External Content**

  You can show external content in the control using the ContentUrl property. This means that you can have the content of another Web page in your project or even the content of a Web site outside of your project appear in a **C1HeaderContentControl**. For more information see Displaying External Content (page 48).



- **AJAX Support**

  Built-in AJAX support lets users interact with the **C1HeaderContentControl** control without the control performing a postback operation back to the server.

- **Set Postbacks**

  Use the AutoPostBack property to determine whether **C1HeaderContentControl** should perform a postback to the server each time a user interacts with the control.

- **Client-Side Object Model**

  The **C1HeaderContentControl** control's client-side object model is exposed so that you can easily customize the control with client-side script.

- **Browser Support**

  C1HeaderContentControl includes support for the Internet Explorer (6.0 or later), Firefox (2 or later), and Safari Web browsers.

- **XHTML Compliant**

  C1HeaderContentControl provides complete XHTML compliance. The output that is generated is fully XHTML 1.1 compliant.

- **Visual Styles**

  Use the built-in Visual Styles to quickly change the **C1HeaderContentControl** control's appearance. Built-in styles include Office 2007 and Vista styles. For more information about Visual Styles, see Visual Styles (page 29) and Setting the Visual Style (page 36).

HeaderContent for ASP.NET AJAX Quick Start

The following quick start is designed to quickly familiarize you with the features for the C1HeaderContentControl control. In this quick start you'll create a simple Web site that uses the C1HeaderContentControl control to organize content in four information blocks that display external Web sites.

Step 1 of 3: Adding C1HeaderContentControl to the Page

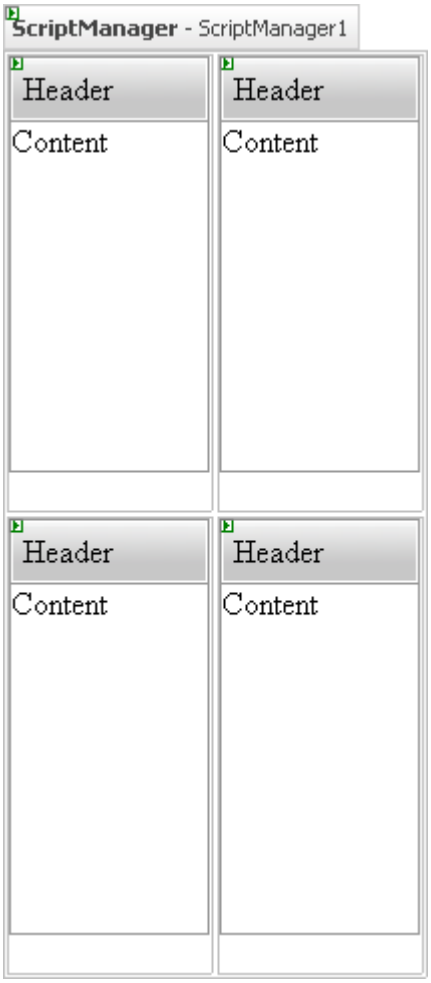
In this quick start guide, you'll create four blocks of information using **HeaderContent for ASP.NET AJAX**. In this first step, you'll create a table with two columns and two rows and add four **C1HeaderControls**, one in each cell of the table.

Complete the following steps:

1. Create a new ASP.NET AJAX-Enabled Website project. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 13).

   Note that as you've created an AJAX-Enabled Web site, a **ScriptManager** control initially appears on the page.
2. Click the **Design** tab located below the Document window to switch to Design view.
3. On the page, add a table (select **Layout | Insert Table**) with two columns and two rows. Each table cell will contain an information block.
4. Click once inside the top left table cell to select it and from the Visual Studio Toolbox add a **C1HeaderContentControl** control to the table cell. Repeat this step with each table cell to add a total of four **C1HeaderContentControl** controls to the form.

   The page will now look similar to the following image:



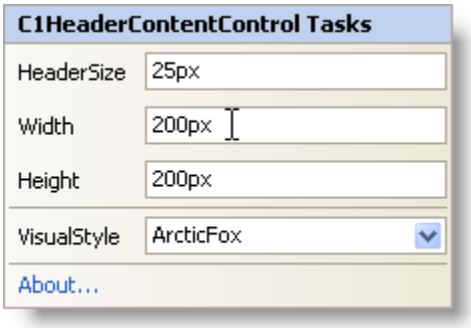
You've completed adding **C1HeaderContentControl**s to your page. In the next step you'll customize the appearance of the controls.

Step 2 of 3: Changing the C1HeaderContentControl's Appearance

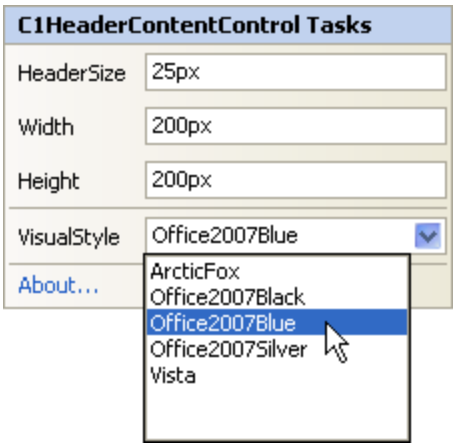
You can easily change the **C1HeaderContentControl**'s appearance at design time by setting properties. The following steps assume you've completed the Step 1 of 3: Adding C1HeaderContentControl to the Page (page 19) topic and added four **C1HeaderContentControl** controls to the page.

Complete the following steps:

1. Select each **C1HeaderContentControl** and click its smart tag to open the **C1HeaderContentControl Tasks** menu. Set each control's **Width** property to **200px** so the external content is more easily visible.



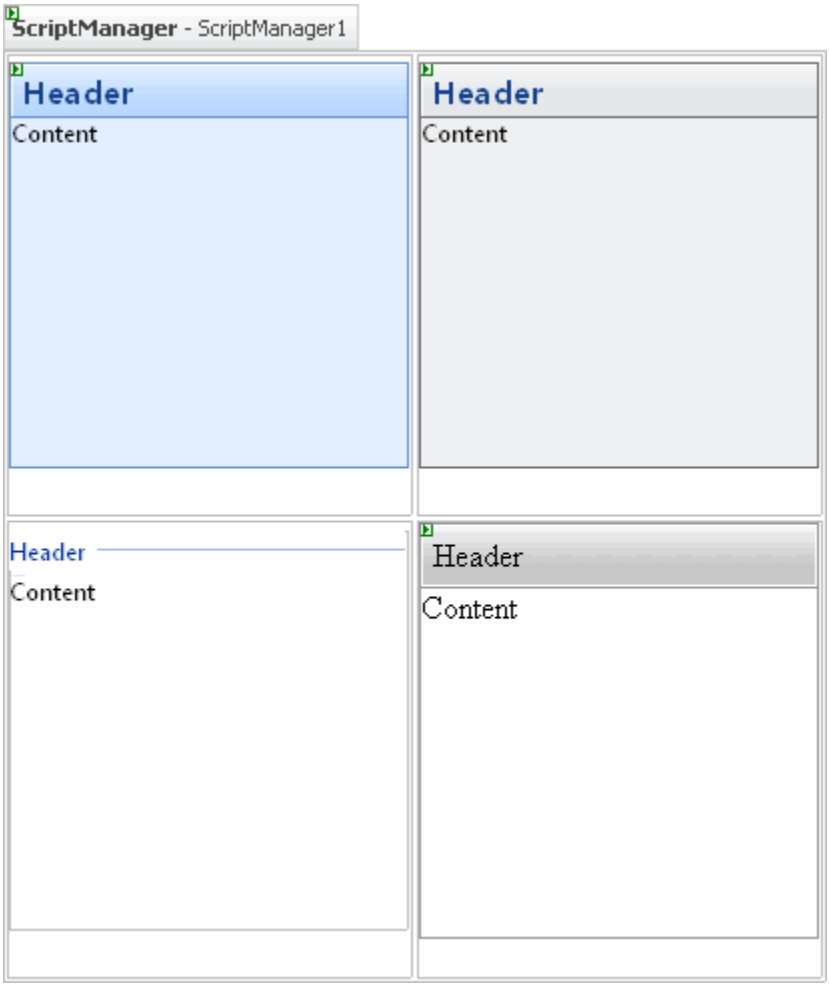
2. Select the **C1HeaderContentControl** in the upper-left cell of the table (**C1HeaderContentControl1**), and select its smart tag to open the **C1HeaderContentControl Tasks** menu.
3. In the **C1HeaderContentControl Tasks** menu, select the Visual Style drop-down arrow and select **Office2007Blue** to set the control's VisualStyle.



4. Repeat steps 2 to 4 to set the VisualStyle property for the other controls:
   - Set the VisualStyle for the control in the upper-right corner (**C1HeaderContentControl2**) to **Office2007Silver**.
   - Set the VisualStyle for the control in the bottom-left corner (**C1HeaderContentControl3**) to **Vista**.

   Note that you'll leave the control in the bottom-right corner (**C1HeaderContentControl4**) set to the default style of **ArcticFox**.

   Your page will now look similar to the following:



In this step you customized the appearance of the **C1HeaderControl** controls. In the next step you'll set the content of each control.

Step 3 of 3: Setting the C1HeaderContentControl's Content

In this step you'll change the content of the **C1HeaderContentControl**'s header and body. You can change the content of each control by deleting the default text and typing in your own text or adding controls. You can use the ContentURL property to display external content in the **C1HeaderContentControl** control. This means that you can have the content of another Web page in your project or even the content of a Web site outside of your project appear in a **C1HeaderContentControl**. Note that the following steps assume you've completed the Step 2 of 3: Changing the C1HeaderContentControl's Appearance (page 20) topic.

Complete the following steps:

1. Select **C1HeaderContentControl1** and in the Properties window set its ContentURL property to "http://www.google.com/".
2. Set the ContentURL of the remaining **C1HeaderContentControl** controls to the following:
   - Set **C1HeaderContentControl2.ContentURL** to "http://www.yahoo.com/".
   - Set **C1HeaderContentControl3.ContentURL** to "http://www.wikipedia.com/".
   - Set **C1HeaderContentControl4.ContentURL** to "http://www.worldcat.org/".

3. Click once in **C1HeaderContentControl1**'s header area to select it and delete the default "Header" text that appears by selecting the text and pressing the BACKSPACE key.
4. Set **C1HeaderContentControl1**'s header to "Google" by typing the text into the header area.
5. Select the header of each remaining **C1HeaderContentControl** control and replace the default "Header" text with the following for each control:
   - Set **C1HeaderContentControl2**'s header to "Yahoo".
   - Set **C1HeaderContentControl3**'s header to "Wikipedia".
   - Set **C1HeaderContentControl4**'s header to "WorldCat".
6. Delete the "Content" text that by default appears in each **C1HeaderContentControl** control by selecting the text and pressing the BACKSPACE key.
7. Run your application. Note that each **C1HeaderContentControl** now displays the content of each respective Web site:

You have successfully created a simple Web site that uses the **C1HeaderContentControl** control to organize content!

HeaderContent For ASP.NET AJAX Top Tips

This section details some tips and best practices that may be helpful when using **HeaderContent for ASP.NET AJAX**. The following tips were compiled from frequently asked user questions posted in the C1HeaderContentControl forum.

Tip 1: Use the ScriptManager Control

ComponentOne HeaderContent for ASP.NET AJAX requires that Microsoft ASP.NET AJAX Extensions and a **ScriptManager** control are included in your project before the C1HeaderContentControl control is placed on the page. This allows you to take advantage of ASP.NET AJAX and certain features such as partial-page rendering and client-script functionality of the Microsoft AJAX Library.

Tip 2: Set Visual Styles to Change the Appearance of the Control

HeaderContent for ASP.NET AJAX includes several built-in visual styles, including Vista and Office 2007 styles, to quickly style your application. See Visual Styles for more information. You can also create your own custom visual style based on an existing visual style. See Adding a Custom Visual Style (page 38) for more information and an example.

Tip 3: Use Header and Content Templates

You can easily reuse content by using Header and Content templates. Templates make it simple to copy and use content across multiple C1HeaderContentControl controls. For more information, see Header and Content Templates (page 31).

Tip 4: User Client-Side Script to Save Server Load

HeaderContent for ASP.NET AJAX had a robust client-side object model, where a majority of server-side properties can be set on the client-side and client-side events are described in the properties window. When a C1HeaderContentControl control is rendered, an instance of the client side control will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the C1HeaderContentControl control without having to postback to the server. See Client-Side Functionality (page 31) for more information.

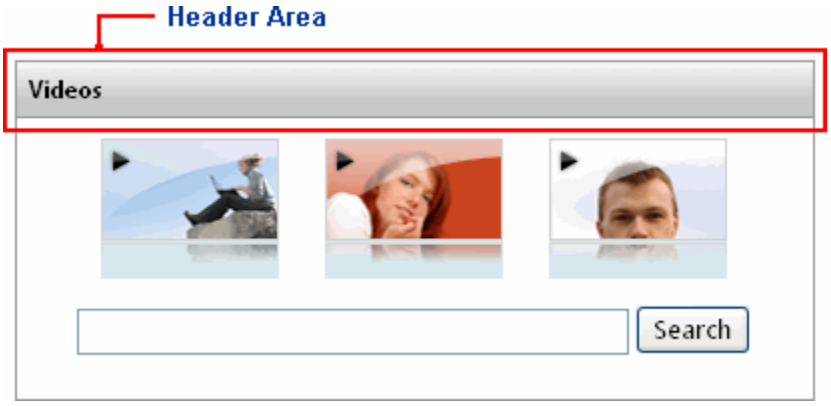
C1HeaderContentControl Elements

This section provides a visual and descriptive overview of the elements that comprise the C1HeaderContentControl control. The topics are categorized into the two distinct elements, the header element and the content element, that represent different aspects of the control.

Header Element

The **C1HeaderContentControl**'s header area appears at the top of the control and initially includes a title with the text "Header". You can add content, including text, HTML content, images, and other controls, to the header area of the **C1HeaderContentControl** using the header template. Elements in the header area of the control can be added and moved on the control through a simple drag-and-drop operation.

The following image labels the header area of the **C1HeaderContentControl** control:



You can use the Header element to customize the header area of the **C1HeaderContentControl** control.

Header

You can add content to the control at design time. To add content, simply click in the header area of the control and begin typing, or add images or controls from the Toolbox to the header area as you would normally.

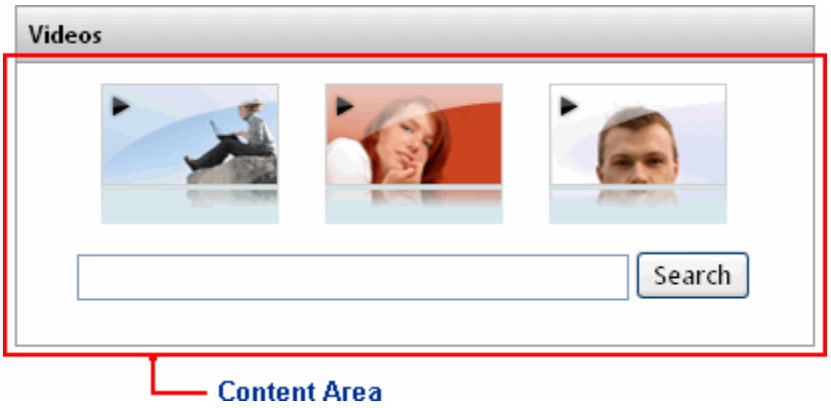
When you add content to the **C1HeaderContentControl**'s header and switch to Source view you will notice that your content appears inside `<Header>` tags inside the `<cc1:C1HeaderContentControl>` tags:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="200px" Width="100px">
    <Content>
        Content
    </Content>
    <Header>
        Header
    </Header>
</cc1:C1HeaderContentControl>
```

Content Element

The **C1HeaderContentControl**'s content area initially consists of an empty area with the text "Content". In the content area you can add rich text through custom HTML content, URL links through the ContentUrl property, and add arbitrary controls through the content template. Elements in the content area of the control can be added and moved on the control through a simple drag-and-drop operation.

The following image labels the content area of the **C1HeaderContentControl** control:



You can use the following elements to customize the content area of the **C1HeaderContentControl** control:

- Content
- ContentUrl

Content

You can add content to the control at design time. To add content, simply click in the content area of the control and begin typing, or add images or controls from the Toolbox to the content area as you would normally.

When you add content to the **C1HeaderContentControl** control and switch to Source view you will notice that your content appears inside `<Content>` tags inside the `<cc1:C1HeaderContentControl>` tags:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="200px" Width="100px">
    <Content>
        Content
    </Content>
    <Header>
        Header
    </Header>
</cc1:C1HeaderContentControl>
```

ContentUrl

You can use the ContentUrl property to set external content to appear within the content area of the **C1HeaderContentControl** control.

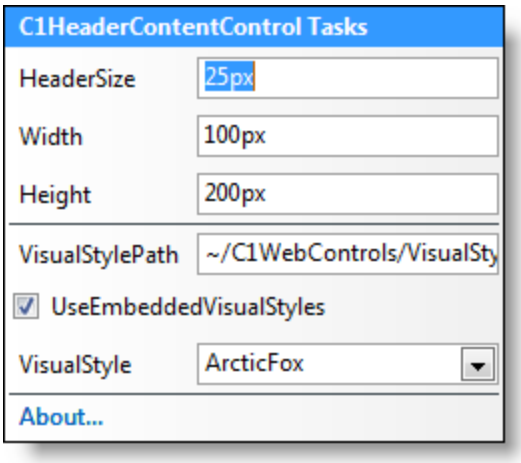
Design-Time Support

The following sections describe how to use **C1HeaderContentControl** 's design-time environment to configure the C1HeaderContentControl control.

C1HeaderContentControl Smart Tag

In Visual Studio, the **C1HeaderContentControl** control includes a smart tag. A smart tag represents a shortcut tasks menu that provides the most commonly used properties in **C1HeaderContentControl**.

To access the **C1HeaderContentControl Tasks** menu, click on the smart tag in the upper-right corner of the **C1HeaderContentControl** control. This will open the **C1HeaderContentControl Tasks** menu:



The **C1HeaderContentControl Tasks** menu operates as follows:

- **HeaderSize**

  Changing the **HeaderSize** value changes the height of the header using the HeaderSize property. By default the HeaderSize property and the height of the header is set to 25 pixels.

- **Width**

  Changing the **Width** value changes the width of the **C1HeaderContentControl** control using the Width property. By default the Width property and the width of the control is set to 100 pixels.

- **Height**

  Changing the **Height** value changes the height of the **C1HeaderContentControl** control using the Height property. By default the Height property and the height of the control is set to 200 pixels.

- **VisualStylePath**

  The **VisualStylePath** property specifies the location of the visual styles used for the control. By default, embedded visual styles are located in **~/C1WebControls/VisualStyles**. If you create a custom style, add it to this location **~/VisualStyles/StyleName/C1HeaderContentControl/styles.css**, set the **VisualStylePath** property to **~/VisualStyles**, and set the **VisualStyle** property to **StyleName** (assuming that **StyleName** is the name used to define the style in the style.css file). Uncheck the **UseEmbeddedVisualStyles** property.

- **UseEmbeddedVisualStyles**

  This check box is checked by default so that the internal visual styles, such as **ArcticFox** and **Vista** can be used. If you want to use your own custom styles, uncheck this check box and specify the location of your visual styles using the **VisualStylePath** property.

- **Visual Style**

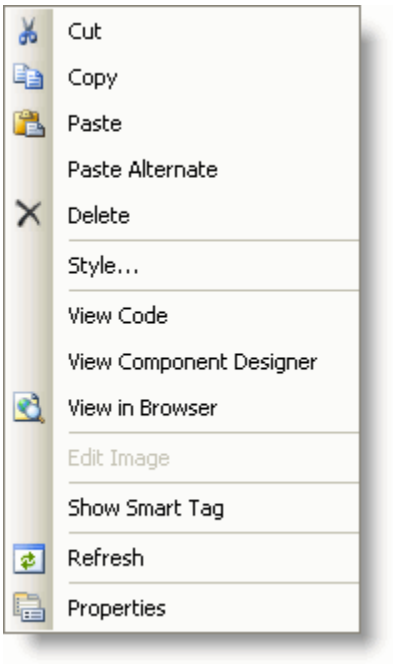
  Clicking the **Visual Style** drop-down box allows you to select from various visual schemes. For more information about available visual styles, see Visual Styles (page 29).

- **About**

  Clicking on the **About** item displays the **About** dialog box, which is helpful in finding the version number of **HeaderContent for ASP.NET AJAX** and online resources.

C1HeaderContentControl Context Menu

Right-click anywhere on the list to display the C1HeaderContentControl context menu, which is a context menu that Visual Studio provides for all .NET controls, although the **C1HeaderContentControl** context menu has a few extra features.



The context menu commands operate as follows:

- **Show Smart Tag**

  Shows the Tasks menu for the C1HeaderContentControl control. For more information on how to use the smart tag and available features in the Tasks menu, see C1HeaderContentControl Smart Tag (page 27).

HeaderContent for ASP.NET AJAX Appearance

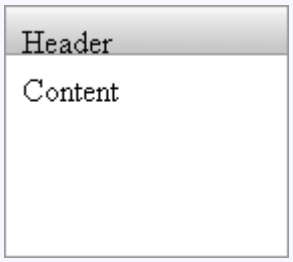
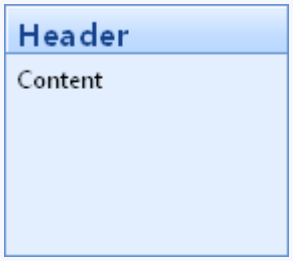
There are several options for customizing the appearance of the C1HeaderContentControl control. The following sections describe how to change the appearance of the control through built-in Visual Styles as well as how to customize other elements of the C1HeaderContentControl control.

Visual Styles

C1HeaderContentControl includes Visual Styles allowing you to easily change the control's appearance. The control includes several built-in visual styles, including Vista and Office 2007 styles, to quickly style your application. You can easily change Visual Styles from the C1HeaderContentControl Tasks menu (page 27), from the Properties window, and in code. For more information on changing Visual Styles see the Setting the Visual Style (page 36) topic.

Note: Additional styles are installed with the product, but they must be added as custom visual styles at this time. These styles include: **BureauBlack**, **Evergreen**, **ExpressionDark**, **RainierOrange**, **RainierPurple**, **ShinyBlue**, and **Windows7**. See Customizing the Control's Appearance (page 31) for more information.

HeaderContent for ASP.NET AJAX includes the following built-in styles:

Visual Style	Preview
ArcticFox (default)	
Office2007Black	
Office2007Blue	
Office2007Silver	
Vista	

Customizing the Control's Appearance

If you choose to completely customize the appearance of the C1HeaderContentControl you may not wish to use any of the available built-in Visual Styles. In that case, to override any visual styles with your own custom appearance, you will need to set the C1HeaderContentControl.**UseEmbeddedVisualStyles** property to **False**.

By default C1HeaderContentControl.**UseEmbeddedVisualStyles** property is **True** and Visual Styles are used. Any customizations you make while using Visual Styles will simply set specific elements in the control's appearance on top of the current Visual Style. To start customizing the control's appearance from scratch set C1HeaderContentControl.**UseEmbeddedVisualStyles** to **False** and set your own styles.

Header and Content Templates

The content of the header and content areas of the C1HeaderContentControl control can be controlled by using templates. C1HeaderContentControl has two special properties, Header, and Content, that can be used to apply templates to the header and content areas of the C1HeaderContentControl control. Header and content templates are useful for additionally customizing your C1HeaderContentControl control to your application and for adding content to the header and content areas of the control.

Client-Side Functionality

C1HeaderContentControl includes a robust client-side object model, where a majority of server-side properties can be set on the client-side and client-side events are described in the properties window.

When a C1HeaderContentControl control is rendered, an instance of the client-side control will be created automatically. This means that you can enjoy the convenience of accessing properties and methods of the C1HeaderContentControl control without having to postback to the server.

For example, suppose a C1HeaderContentControl control with name **C1HeaderContentControl1** is hosted on a Web page; when the page is rendered, a corresponding client-side object will be created. Use the following syntax to get the client object:

```
$get("C1HeaderContentControl1").control
```

OR

```
$find("C1HeaderContentControl1")
```

By using **C1HeaderContentControl**'s client-side functionality, you can implement many features in your Web page without the need to take up time by sending information to the Web server. Thus, using client-side methods and events can increase the efficiency of your Web site.

The following topics will describe the available client-side properties.

Client-Side Properties

The following conventions are used when accessing the client object properties:

- Server properties on the client are implemented as a pair of Get- and Set- methods.
- Method names must start with "get_" (Get-method) and "set_" (Set-method) followed with the server property name. The first letter of the server property name must be lowercase (camel case).

For example in the code below the C#, Visual Basic, and JavaScript examples are equivalent:

- Visual Basic

```
Dim s As String = C1HeaderConentControl1.ContentUrl
C1HeaderContentControl1.ContentUrl = s
```

- C#

```
string s = C1HeaderConentControl1.ContentUrl;
C1HeaderConentControl1.ContentUrl = s;
```

- JavaScript

```
var oHeader = $get("C1HeaderContentControl1").control;
var s = oHeader.get_contentUrl();
oHeader.set_contentUrl(s);
```

For an example of setting the Height and Width properties at run time, see Setting the Height and Width at Run Time (page 51).

C1HeaderContentControl includes a rich client-side object model in which several properties can be set on the client side. For information about these client-side methods and what properties can be set on the client side, see the C1HeaderContentControl Reference.

HeaderContent for ASP.NET AJAX 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 your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for ASP.NET | Samples | Palomino Samples**.

C# Samples

The following pages within the **ControlExplorer** sample installed with **ComponentOne Studio for ASP.NET AJAX** detail the C1HeaderContentControl control's functionality:

Sample	Description
ContentUrl	Details the ContentUrl property, allowing you to display external content in the C1HeaderContentControl control.
Template	Showcases how Header and Content templates can be used to display controls and customized content in the header and body areas of the C1HeaderContentControl control.
VisualStyles	Displays the built-in C1HeaderContentControl control Visual Styles including ArcticFox, Office2007Black, Office2007Blue, Office2007Silver, and Vista.

HeaderContent for ASP.NET AJAX Task-Based Help

The task-based help assumes that you are familiar with programming in ASP.NET and know how to use controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of C1HeaderContentControl's features, and get a good sense of what the C1HeaderContentControl control can do.

Each topic provides a solution for specific tasks using the C1HeaderContentControl control. Each task-based help topic also assumes that you have created a new ASP.NET AJAX-Enabled project. For additional information on this topic, see Creating an AJAX-Enabled ASP.NET Project (page 13).

Note: HeaderContent for ASP.NET AJAX requires Microsoft ASP.NET AJAX Extensions installed and a **ScriptManager** on the page before the C1HeaderContentControl control is placed on the page. You must create an ASP.NET AJAX-Enabled Project so that the **ScriptManager** and Microsoft AJAX Extensions are included on the page. For more information, see Creating an AJAX-Enabled ASP.NET Project (page 13). For more information about Microsoft ASP.NET AJAX Extensions, see http://ajax.asp.net/. For information about the **ScriptManager**, see MSDN.

Creating a C1HeaderContentControl in Code

Creating a C1HeaderContentControl control in code is fairly simple. In the following steps you'll add a **PlaceHolder** control to the page, add an import statement, add and customize the C1HeaderContentControl, and add the control to the **PlaceHolder**.

Complete the following steps:

1. In Design view, navigate to the Visual Studio Toolbox and add a **PlaceHolder** control to your page.
2. Double-click the page to create the **Page_Load** event and switch to Code view.
3. Add the following statement to the top of the Code Editor to import the appropriate namespace:
   - Visual Basic
     ```
     Imports C1.Web.UI.Controls.C1HeaderContentControl
     ```
   - C#
     ```
     using C1.Web.UI.Controls.C1HeaderContentControl;
     ```
4. Add the following code to the **Page_Load** event to create and customize the **C1HeaderContentControl** control.
   - Visual Basic
     ```
     ' Create a new C1HeaderContentControl.
     Dim C1HC As New C1HeaderContentControl
     ' Set the control's size, appearance, and content.
     C1HC.VisualStyle = "Office2007Blue"
     C1HC.Height = 200
     C1HC.Width = 200
     C1HC.ContentUrl = "http://www.wikipedia.com/"
     ' Add the HeaderContentControl to the PlaceHolder control.
     PlaceHolder1.Controls.Add(C1HC)
     ```
   - C#
     ```
     // Create a new C1HeaderContentControl.
     C1HeaderContentControl C1HC = new C1HeaderContentControl();
     // Set the control's size, appearance, and content.
     C1HC.VisualStyle = "Office2007Blue";
     C1HC.Height = 200;
     C1HC.Width = 200;
     C1HC.ContentUrl = "http://www.wikipedia.com/";
     // Add the HeaderContentControl to the PlaceHolder control.
     PlaceHolder1.Controls.Add(C1HC);
     ```

Run your application and observe:

The C1HeaderContentControl control was created and displays external content:

Suppressing Header Post Back

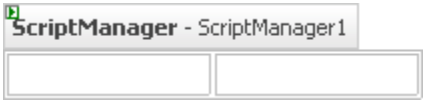
You can easily set whether or not the header of the C1HeaderContentControl automatically posts back to the server by using the SuppressHeaderPostbacks property. By default the SuppressHeaderPostbacks property is set to **False** and any client-side click handlers in the header are not suppressed; to suppress client-side click handlers in the header of a **C1HeaderContentControl**, set SuppressHeaderPostbacks to **True**.

In this topic you'll create two **C1HeaderContentControl**s with hyperlinks in the header area, one **C1HeaderContentControl** with SuppressHeaderPostbacks set to **True** and one by default with SuppressHeaderPostbacks set to **False**.

Complete the following steps:

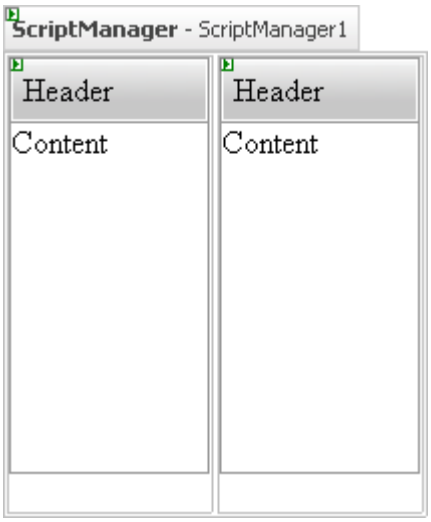
1. Select **Layout | Inset Table** and in the **Insert Table** dialog box create a table with one row and two columns.

   Your page should now look similar to the following:



2. Click within the left cell of the table and in the Visual Studio Toolbox double-click the **C1HeaderContentControl** to add the control to the left cell.
3. Click within the right cell of the table and in the Visual Studio Toolbox double-click the **C1HeaderContentControl** to add the control to the right cell.

   The page should now look similar to the following:



4. Delete the default "Header" text within each **C1HeaderContentControl**'s header and from the Toolbox add a **HyperLink** control to each header.
5. Select each **HyperLink** control and set the following properties in the Properties window:
   - Set both the **HyperLink** controls' **NavigateUrl** properties to "http://www.componentone.com/".
   - Set both the **HyperLink** controls' **Target** properties to "_blank".

- Set the left side **HyperLink** control's **Text** property to "Suppressed".
- Set the right side **HyperLink** control's **Text** property to "Unsuppressed".

The page will now look similar to the following:

6. Select the left **C1HeaderContentControl** and in the Properties window set its SuppressHeaderPostbacks property to **True**.

   By default the right-side **C1HeaderContentControl**'s SuppressHeaderPostbacks property will remain set to **False**.

Run your application and observe:

1. Click the hyperlink in the left **C1HeaderContentControl**'s header. The hyperlink does not open.

   Note that you can manually right-click the link to open the link, but because the SuppressHeaderPostbacks property was set to **True**, client-side click handlers in the header of the **C1HeaderContentControl** were suppressed.

2. Click the hyperlink in the right **C1HeaderContentControl**'s header. The hyperlink opens the ComponentOne Web site in a new page as expected.

Setting the Visual Style

The control includes several built-in visual styles, including Vista and Office 2007 styles, to style your application. For more information about available styles, see Visual Styles (page 29). You can easily change Visual Styles in Source view, from the **C1HeaderContentControl Tasks** menu, from the Properties window, and in code. In the examples below, the Vista style was applied to the **C1HeaderContentControl**.

In Source View

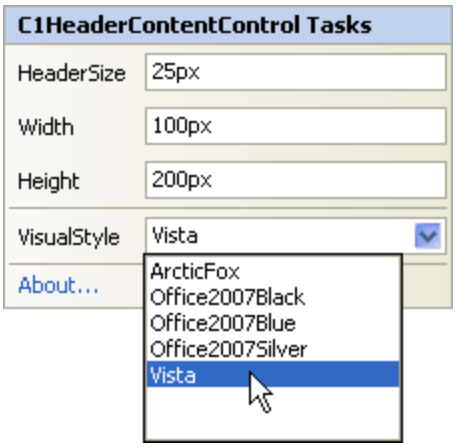
In Source view add `VisualStyle="Vista"` to the `<cc1:C1HeaderContentControl>` tag so it appears similar to the following:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="200px" Width="100px" VisualStyle="Vista">
```

From the Tasks Menu

You can access the **VisualStyle** property from the **C1HeaderContentControl Tasks** menu:

1. Click on the **C1HeaderContentControl**'s smart tag to open the **C1HeaderContentControl Tasks** menu.
2. Click the **Visual Style** drop-down arrow and select a Visual Style, for example **Vista**.

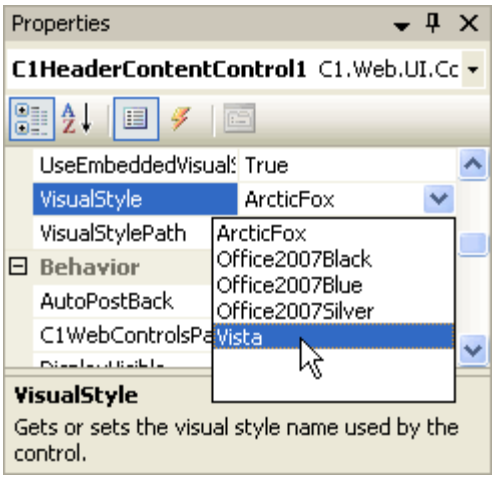


The Visual Style you choose will be applied to the **C1HeaderContentControl**.

From the Properties Window

You can select a Visual Style to apply from the Properties window:

1. Click on the **C1HeaderContentControl** to select it.
2. Navigate to the Properties window and select the drop-down arrow next to the **C1HeaderContentControl.VisualStyle** property.
3. Select a style to apply, for example **Vista**.



The Visual Style you chose will be applied to the **C1HeaderContentControl**.

In Code

Add the following code to the **Page_Load** event to set the VisualStyle property to **Vista**:

- Visual Basic

```
Me.C1HeaderContentControl1.VisualStyle = "Vista"
```

- C#

```
this.C1HeaderContentControl1.VisualStyle = "Vista";
```

Adding a Custom Visual Style

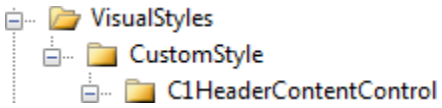
If you choose to completely customize the appearance of the C1HeaderContentControl control you may not wish to use any of the available built-in Visual Styles. In that case, you can create a custom visual style based on an existing **ComponentOne HeaderContent for ASP.NET AJAX** visual style. In the following steps you'll use the **VisualStyle**, **VisualStylePath**, and **UseEmbeddedVisualStyles** properties to create a custom visual style for your C1HeaderContentControl control. For more information on custom visual styles, see Customizing the Control's Appearance (page 31).

Step 1: Setting up Folders

In order to create a custom visual style, you must first add new folders to your project to create the appropriate directory structure for visual style files. You'll add a style sheet and image files to the style folder you create.

To add set up the application, complete the following steps:

1. Navigate to the Solution Explorer, right-click on your project's name, and select **New Folder**.
2. Rename the new folder "VisualStyles".
3. Right-click on the **VisualStyles** folder in the Solution Explorer and select **New Folder**.
4. Name this folder "CustomStyle".
5. Right-click on the **CustomStyle** folder in the Solution Explorer and select **New Folder**.
6. Name this folder "C1HeaderContentControl". The folder hierarchy will now appear similar to the following image:



In general, **HeaderContent for ASP.NET AJAX** visual styles should be placed in a **C1HeaderContentControl** folder within a folder named for the style, here **CustomStyle**, which is in a **VisualStyles** folder in the project.

Step 2: Adding Files to the Folder

Once you've created the folder hierarchy, you will need to add a style sheet and image folder to create your custom style. In this example, you'll use the existing **ArcticFox** style

To add a custom visual style, complete the following steps:

1. In the Solution Explorer, right-click the **C1HeaderContentControl** folder and select **New Folder**. Rename the new folder "Images". In this example you will not need to add images to the folder, but you can use this folder as a placeholder for image files.
2. Right-click the **C1HeaderContentControl** folder and select **Add New Item**.
3. In the **Add New Item** dialog box choose **Style Sheet** from the list of installed templates, name the style sheet, "styles.css", and click the **Add** button to close the dialog box and add the file to your project.
4. If it did not automatically open, open the **styles.css** file by double-clicking the item in the Solution Explorer.
5. Replace the existing body style in the styles.css file with the following CSS styles:

Custom CSS Style

```
/* =========== CustomStyle C1HeaderContentControl
========================================== */
.C1HeaderContentControl_CustomStyle
{
border: solid 1px #999;
}
.C1HeaderContentControl_CustomStyle .C1eContentPanel .C1eOuter,
.C1HeaderContentControl_CustomStyle .C1eContentPanel .C1eInner,
.C1HeaderContentControl_CustomStyle .C1eContentPanel .C1eContent
{
  height: 100%;
  position: relative;
  float: none;
  display: block;
}
/* Common definitions */
.C1HeaderContentControl_CustomStyle .C1eContentPanel {
      background: #fff;
      display:block;
      margin:0;
      padding:0;
}
.C1HeaderContentControl_CustomStyle .C1eContentPanel .C1eOuter .C1eInner
{
  padding:0;
  margin:0;
}
/* C1Top, C1Bottom */
.C1HeaderContentControl_CustomStyle-C1Bottom .C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Top .C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Right .C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Left .C1eHeaderPanel
{
border: solid 1px #fff;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1eHeaderPanel
{
      background: #C6C6C6;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
      min-height:20px;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Opened-C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1eHeaderPanel
{
      background: #ffffff;
      color: #000;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Opened-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1Hover-
C1eHeaderPanel
```

```
{
      background: #ffffff;
      color: #000;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Opened-C1Active-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1Active-C1Hover-
C1eHeaderPanel
{
      background: #666;
      color: #fff;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Closed-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Closed-C1Hover-
C1eHeaderPanel
{
      background: #ffffff;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Closed-C1Active-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Closed-C1Active-C1Hover-
C1eHeaderPanel
{
      background: #999;
}
/* C1Right, C1Left */
.C1HeaderContentControl_CustomStyle-C1Right .C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Left .C1eHeaderPanel
{
      background: #C6C6C6;
      display:block;
      float:left;
      clear:none;
      margin:0;
      padding:0;
      min-width:20px;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1Opened-C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Left .C1Opened-C1eHeaderPanel
{
      background: #ffffff;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1Opened-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Left .C1Opened-C1Hover-
C1eHeaderPanel
{
      background: #ffffff;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1Opened-C1Active-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Left .C1Opened-C1Active-C1Hover-
C1eHeaderPanel
{
      background: #666;
}
```

```
.C1HeaderContentControl_CustomStyle-C1Right .C1eHeaderPanel .C1eOuter,
.C1HeaderContentControl_CustomStyle-C1Left .C1eHeaderPanel .C1eOuter
{
      background: none;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1Closed-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Right .C1Closed-C1Hover-
C1eHeaderPanel
{
      background: #ffffff;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1Closed-C1Active-C1Hover-
C1eHeaderPanel,
.C1HeaderContentControl_CustomStyle-C1Right .C1Closed-C1Active-C1Hover-
C1eHeaderPanel
{
      background: #666;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1eHeaderPanel .C1eOuter,
.C1HeaderContentControl_CustomStyle-C1Right .C1eHeaderPanel .C1eOuter
{
      background: none;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
}
.C1HeaderContentControl_CustomStyle .C1eHeaderPanel .C1eInner
{
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
}
.C1HeaderContentControl_CustomStyle-C1Bottom .C1eHeaderPanel .C1eContent
{
      background: #66FF33;
      color: #000000;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:5px;
      vertical-align:middle;
}
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1Active-C1Hover-
C1eHeaderPanel .C1eContent,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Closed-C1Active-C1Hover-
C1eHeaderPanel .C1eContent
```

```
{
color: #fff;
}
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1eHeaderPanel
.C1eContent,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1Hover-
C1eHeaderPanel .C1eContent,
.C1HeaderContentControl_CustomStyle-C1Bottom .C1Opened-C1Active-C1Hover-
C1eHeaderPanel .C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1eHeaderPanel .C1eContent
{
      background: #66FF33;
      color: #000000;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:5px;
      vertical-align:middle;
}
.C1HeaderContentControl_CustomStyle-C1Top .C1Opened-C1eHeaderPanel
.C1eContent,
.C1HeaderContentControl_CustomStyle-C1Top .C1Opened-C1Hover-C1eHeaderPanel
.C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1eHeaderPanel .C1eContent
{
      background: #66FF33;
      color: #000000;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0 5px;
      vertical-align:middle;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1Opened-C1eHeaderPanel
.C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Right .C1Opened-C1Hover-
C1eHeaderPanel .C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1eHeaderPanel .C1eContent
{
      background: #66FF33;
      color: #000000;
      display:block;
      float:none;
```

```
      clear:both;
      margin:0;
      padding:0 5px;
      vertical-align:middle;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1Opened-C1eHeaderPanel
.C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Left .C1Opened-C1Hover-
C1eHeaderPanel .C1eContent
{
      background: #66FF33;
}
.C1HeaderContentControl_CustomStyle-C1Bottom .C1eContentPanel
{
}
.C1HeaderContentControl_CustomStyle-C1Top .C1eContentPanel
{
}
.C1HeaderContentControl_CustomStyle-C1Right .C1eContentPanel
{
}
.C1HeaderContentControl_CustomStyle-C1Left .C1eContentPanel
{
}
/*----DISABLED STATE----*/
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Top .C1eHeaderPanel
{
      border:solid 1px #999;
      background: #C6C6C6;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
      min-height:20px;
      border-top: none;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Bottom .C1eHeaderPanel
{
      border:solid 1px #999;
      background: #C6C6C6;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
      min-height:20px;
      border-bottom: none;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Right .C1eHeaderPanel
{
      border:solid 1px #999;
      background: #C6C6C6;
      display:block;
```

```
      float:none;
      clear:both;
      margin:0;
      padding:0;
      min-height:20px;
      border-bottom: none;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Left .C1eHeaderPanel
{
      border:solid 1px #999;
      background: #C6C6C6;
      display:block;
      float:none;
      clear:both;
      margin:0;
      padding:0;
      min-height:20px;
      border-top: none;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Bottom .C1eHeaderPanel
{
    border: solid 1px #999;
    border-bottom: none;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Bottom .C1eHeaderPanel
.C1eOuter .C1eInner .C1eContent
{
    padding: 5px;
}
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Top .C1eContentPanel
.C1eOuter .C1eInner .C1eContent,
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Bottom .C1eContentPanel
.C1eOuter .C1eInner .C1eContent,
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Right .C1eContentPanel
.C1eOuter .C1eInner .C1eContent,
.C1HeaderContentControl_CustomStyle-C1Disabled-C1Left .C1eContentPanel
.C1eOuter .C1eInner .C1eContent
{
    border:1px solid #6f7074;
}
/* end of CustomStyle C1HeaderContentControl
========================================== */
```

Step 3: Applying the Custom Style

Once you've created the custom style, you will need to apply the style to the **C1HeaderContentControl** control. In this step you'll learn how to apply a style in Design view, in code, and in Source view.

In Design View

To apply a custom visual style in Design view, complete the following steps:

1. Select the C1HeaderContentControl control on your form and click the smart tag to open the **C1HeaderContentControl Tasks** menu.
2. Deselect **UseEmbeddedVisualStyles** check box to set the **UseEmbeddedVisualStyles** property to **False**.
3. Set the **VisualStylePath** property is to **~/VisualStyles**.
4. Select **CustomVisualStyle (external)** from the **VisualStyle** property drop-down list.

5. Click on the page to close the Tasks menu and apply the styles.

In Source View

To add a custom visual style in Source view, complete the following steps:

1. Switch to Source view by right-clicking the **Default.aspx** page in the Solution Explorer and selecting **View Markup**.
2. Enter Source view and enter `VisualStyle="CustomStyle"`, `VisualStylePath="~/VisualStyles"`, and `UseEmbeddedVisualStyles="False"` into the `<cc1:C1HeaderContentControl>` tag.

   The control's markup will appear similar to the following:

```
cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
Expanded="False" HeaderSize="25px" Height="200px"
VisualStyle="CustomStyle" VisualStylePath="~/VisualStyles"
UseEmbeddedVisualStyles="False" Width="100px">
```

In Code

To add a custom visual style, complete the following steps:

1. Double-click the page in Design view to switch to Code view and create the **Page_Load** event handler.
2. Set the **UseEmbeddedVisualStyles** property to **False** by adding the following code to the **Page_Load** event:
   - Visual Basic

```
C1HeaderContentControl1.UseEmbeddedVisualStyles = False
```

   - C#

```
C1HeaderContentControl1.UseEmbeddedVisualStyles = false;
```

3. Set the **VisualStylePath** property to the folder you created:
   - Visual Basic

```
C1HeaderContentControl1.VisualStylePath = "~/VisualStyles"
```

   - C#

```
C1HeaderContentControl1.VisualStylePath = "~/VisualStyles";
```

4. Apply the custom visual style to the control:
   - Visual Basic

```
C1HeaderContentControl1.VisualStyle = "CustomStyle"
```

   - C#

```
C1HeaderContentControl1.VisualStyle = "CustomStyle";
```

Run the program and observe that the C1HeaderContentControl control has adopted your custom visual style.

Setting the Content Background Color

You can customize the appearance of the C1HeaderContentControl by using styles. You can create a new style from scratch or customize the built-in styles. In this topic you'll customize the **ArcticFox** theme by adding a style to change the background color of the control. The following topic assumes you've added a C1HeaderContentControl to the page.

Complete the following steps:

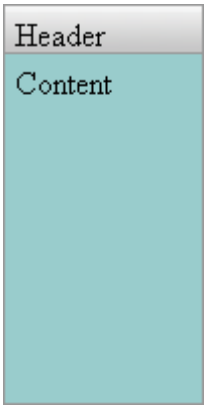
1. Click on the **C1HeaderContentContro**l's smart tag to open the **C1HeaderContentControl Tasks** menu.
2. From the Tasks menu, select the **VisualStyle** drop-down arrow and select the **ArcticFox** scheme. The style you'll add in the next steps will be added on top of this theme.

3. Switch to Source view.
4. Add the following style markup between the `<head>` and `</head>` tags at the top of the document:

```
<style>
.C1HeaderContentControl_ArcticFox .C1hContentPanel {
   background: #99CCCC;
   display:block;
   margin:0;
   padding:0;
}
</style>
```

   This will set the background color of the control's content area.
5. Save and run your project.

   The **C1HeaderContentControl** will appear in the **ArcticFox** theme and the control's content area will appear with a green background:



Resizing the Header

You can easily change the height of the header by setting the HeaderSize property. By default the height of the header is set to **25px**. You can easily change the header's height in Source view, from the **C1HeaderContentControl Tasks** menu, from the Properties window, or in code.

In Source View

In Source view change `HeaderSize="25px"` in the `<cc1:C1HeaderContentControl>` tag to the size you wish to set the header's height to, for example:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="100px" Height="200px" Width="100px">
```

The above will resize the header's height to 100 pixels tall.

From the Tasks Menu

You can set the header's height from the **C1HeaderContentControl Tasks** menu:

1. Click on the **C1HeaderContentControl**'s smart tag to open the **C1HeaderContentControl Tasks** menu.
2. In the **HeaderSize** text box, replace "25px" with the size you want the header to be, for example "100px".
3. Click outside of the **C1HeaderContentControl Tasks** menu to close the menu and set the height of the header.

From the Properties Window

You can change the HeaderSize property to set the header's height in the Properties window:

1. Click on the **C1HeaderContentControl** to select it.
2. Navigate to the Properties window and if needed expand the **Layout** node to locate the HeaderSize property.
3. Next to **HeaderSize**, replace "25px" with the size you want the header to be, for example "100px".
4. Press the ENTER key or click outside of the Properties window for the header height you set to be applied to the **C1HeaderContentControl**.

In Code

Add the following code to the **Page_Load** event to set the HeaderSize property to 100 pixels:

- Visual Basic
```
Me.C1HeaderContentControl1.HeaderSize = 100
```
- C#
```
this.C1HeaderContentControl1.HeaderSize = 100;
```

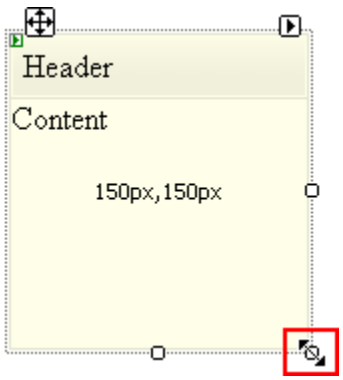
Resizing the Control

You can easily change the height and width on the **C1HeaderContentControl** by setting the Height and Width properties. By default the height of the control is set to **200px** and the width of the control is set to **100px**. This topic details how to change the control's height and width in Design view, Source view, from the **C1HeaderContentControl Tasks** menu, from the Properties window, or in code. For information on resizing the control on the client side instead, see the Setting the Height and Width at Run Time topic.

In Design View

To resize the **C1HeaderContentControl** control in Design view, click the bottom-right corner of the control and perform a drag-and-drop operation to set the control's size.

For example, in the following image the control has been resized to 150 pixels wide and 150 pixels tall:



In Source View

In Source view change `Height="200px" Width="100px"` in the `<cc1:C1HeaderContentControl>` tag to the size you wish to set the control's height and width to, for example:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="150px" Width="150px">
```

The above will resize the control to 150 pixels tall and 150 pixels wide.

From the Tasks Menu

You can set the control's height and width from the **C1HeaderContentControl Tasks** menu:

1. Click on the **C1HeaderContentControl**'s smart tag to open the **C1HeaderContentControl Tasks** menu.
2. In the **Width** text box, replace "100px" with the size you want the control's width to be, for example "150px".
3. In the **Height** text box, replace "200px" with the size you want the control's height to be, for example "150px".
4. Click outside of the **C1HeaderContentControl Tasks** menu to close the menu and set the width and height of the control.

From the Properties Window

You can change the Height and Width properties to set the control's height and width in the Properties window:

1. Click on the **C1HeaderContentControl** to select it.
2. Navigate to the Properties window and if needed expand the **Layout** node to locate the Height and Width properties.
3. Next to **Height**, replace "200px" with the size you want the control's height to be, for example "150px".
4. Next to **Width**, replace "100px" with the size you want the control's width to be, for example "150px".
5. Press the ENTER key or click outside of the Properties window for the height and width you set to be applied to the **C1HeaderContentControl** control.

In Code

Add the following code to the **Page_Load** event to set the Height and Width properties to 150 pixels:

- Visual Basic
```
Me.C1HeaderContentControl1.Height = 150
Me.C1HeaderContentControl1.Width = 150
```
- C#
```
this.C1HeaderContentControl1.Height = 150;
this.C1HeaderContentControl1.Width = 150;
```

Displaying External Content

You can use the ContentUrl property to display external content in the **C1HeaderContentControl** control. This means that you can have the content of another Web page in your project or even the content of a Web site outside of your project appear in a **C1HeaderContentControl**. You can easily set the ContentUrl property in Source view, from the Properties window, or in code.

In Source View

In Source view change add `ContentUrl` in the `<cc1:C1HeaderContentControl>` tag to set the external URL to display in the **C1HeaderContentControl** control, for example:

```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="200px" Width="100px"
ContentUrl="http://www.google.com/">
```

The above will set the Google Web site to appear in the **C1HeaderContentControl**.

From Design View

You can change the ContentUrl property to the URL to be displayed in the **C1HeaderContentControl** via the Properties window:

1. Click on the **C1HeaderContentControl** to select it.
2. Navigate to the Properties window and if needed expand the **Misc** node to locate the ContentUrl property.

3. Next to **ContentUrl**, type the URL for the page you wish to appear in the control, for example "http://www.google.com/".
4. Press the ENTER key or click outside of the Properties window for ContentUrl you set to be applied to the **C1HeaderContentControl** control.

In Code

Add the following code to the **Page_Load** event to set the ContentUrl property to display external content in the **C1HeaderContentControl** control:

- Visual Basic
```
Me.C1HeaderContentControl1.ContentUrl = "http://www.google.com/"
```
- C#
```
this.C1HeaderContentControl1.ContentUrl = "http://www.google.com/";
```

Setting the ContentUrl Property and Header at Run Time

The ContentUrl property allows you display external content, such as internal or external Web pages, in the **C1HeaderContentControl** control. In this topic you'll set the **C1HeaderContentControl** control's ContentUrl property and header content to change on the client side. If, instead, you'd like to set the ContentUrl property in Source view, from the Properties window, or in code, see the Displaying External Content (page 48) topic.

To set the control's header content and ContentUrl property at run time, create a new AJAX-Enabled Web site project and complete the following steps:

1. Navigate to the Visual Studio Toolbox and double-click the **C1HeaderContentControl** icon to add the control to your page.
2. Click on the **Source** tab to switch to Source view, and note that the body of the page looks similar to the following:
```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <br />
        <cc1:C1HeaderContentControl ID="C1HeaderContentControl1"
runat="server" HeaderSize="25px"
            Height="200px" Width="100px">
            <Header>
                Header</Header>
            <Content>
                Content</Content>
        </cc1:C1HeaderContentControl>
    </div>
    </form>
</body>
```
3. In the `<cc1:C1HeaderContentControl>` tag, delete the default "Content" text between the `<Content></Content>` tags.
4. In the `<cc1:C1HeaderContentControl>` tag, change `Height="200px" Width="100px"` to `Height="250px" Width="350px"` so the tag appears similar to the following:
```
<cc1:C1HeaderContentControl ID="C1HeaderContentControl1" runat="server"
HeaderSize="25px" Height="250px" Width="350px">
    <Header>
        Header</Header>
    <Content></Content>
```

```
</cc1:C1HeaderContentControl>
```

This will resize the **C1HeaderContentControl**.

5. Add the following tags just below the `</cc1:C1HeaderContentControl>` tag to add a button and text box to set the ContentUrl property.

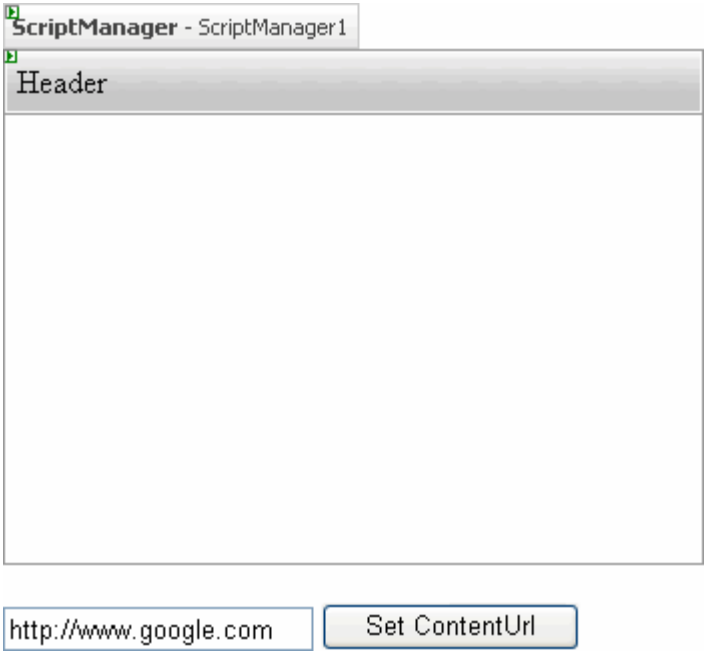
```
<br /><input id="TxtUrl" type="text" value="http://www.google.com" />
<input id="Button3" type="button" value="Set ContentUrl"
onclick="LoadContentUrl()"/>
```

6. At the top of the page, add the following JavaScript just above the opening `<html>` tag:

```
<script language="javascript" type="text/javascript">
function LoadContentUrl()
{
var oHeader =
Sys.Application.findComponent("<%=C1HeaderContentControl1.ClientID%>");
oHeader.set_contentUrl(document.getElementById('TxtUrl').value);
oHeader.get_headerPanel().get_contentElement().innerHTML =
document.getElementById('TxtUrl').value;
}
</script>
```

This function will change the **C1HeaderContentControl**'s ContentUrl property and the title in the header of the control when the button on the page is pressed.

7. Switch to Design view and note that the page now looks similar to the following:



Run the project and observe:

Click the **Set ContentUrl** button; the page will look similar to the following:

Enter a new URL in the text box and click the **Set ContentUrl** button again.

The Web site you have just entered will be loaded into the **C1HeaderContentControl** control.

Setting the Height and Width at Run Time

You can easily change the height and width on the **C1HeaderContentControl** by setting the Height and Width properties. By default the height of the control is set to **200px** and the width of the control is set to **100px**. After completing the steps in this topic users willl be able to resize the control at run time. For information about changing the control's height and width in Design view, Source view, In Design view, and in code, instead, see Resizing the Control.

To set the control's Height and Width properties at run time, create a new AJAX-Enabled Web site project and complete the following steps:

1. Navigate to the Visual Studio Toolbox and double-click the **C1HeaderContentControl** icon to add the control to your page.
2. Click on the **Source** tab to switch to Source view, and note that the body of the page looks similar to the following:

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <br />
        <cc1:C1HeaderContentControl ID="C1HeaderContentControl1"
runat="server" HeaderSize="25px" Height="200px" Width="100px">
            <Header>Header</Header><Content>Content</Content>
        </cc1:C1HeaderContentControl>
    </div>
    </form>
</body>
```

3. Add the following tags just below the `</cc1:C1HeaderContentControl>` tag to add a button and text boxes to set the Height and Width properties.

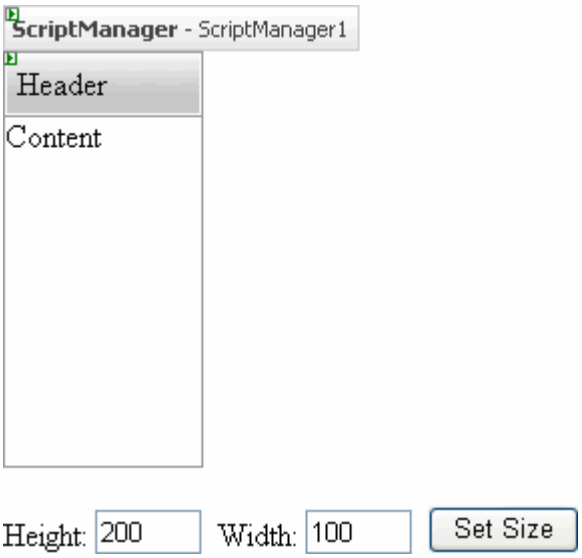
```
<br />Height: <input id="TxtHeight" type="text" size="3"
value="200"/>  Width: <input id="TxtWidth" type="text" size="3"
value="100"/>  <input id="Button3" type="button" value="Set Size"
onclick="SetSize()"/>
```

4. At the top of the page, add the following JavaScript just above the opening `<html>` tag:

```
<script language="javascript" type="text/javascript">
function SetSize()
{
var oHeader =
Sys.Application.findComponent("<%=C1HeaderContentControl1.ClientID%>");
oHeader.set_height(parseInt(document.getElementById('TxtHeight').value)
);
oHeader.set_width(parseInt(document.getElementById('TxtWidth').value));
}
</script>
```

   This function will change the **C1HeaderContentControl**'s Height and Width properties when the button on the page is pressed.

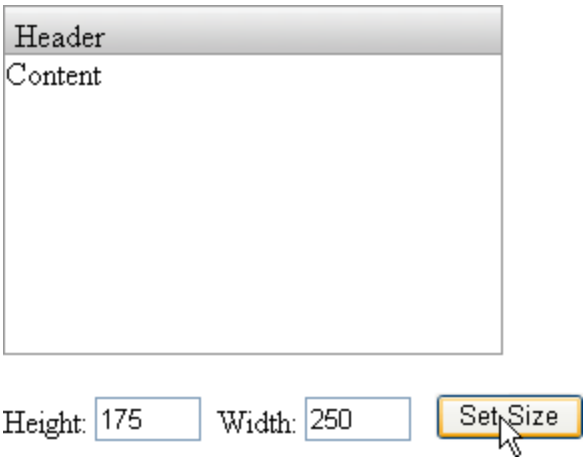
5. Switch to Design view and note that the page now looks similar to the following:



Run the project and observe:

1. Change the **Height** and **Width** values in the text boxes, for example set the **Height** to **175** and the **Width** to **250**.
2. Press the **Set Size** button.

   The control will resize and the page will appear similar to the following.



3. Try setting the control's **Height** and **Width** to different values to view how the control's size changes.