ComponentOne

WebDataObjects for ASP.NET

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 WebDataObjects for ASP.NET Overview

ComponentOne WebDataObjects for ASP.NET provides automatic server-side data caching and multiple caching options to increase speed and performance in your database Web applications.

WebDataObjects for ASP.NET includes two editions to provide you with a choice of tools to best suit your needs: Enterprise and Express. Enterprise edition includes the C1WebDataSet and C1WebSchemaDef components, allowing you to create functionally rich Web pages and independent, reusable business logic for WinForms as well as WebForms. If you are looking for ease-of-use, Express edition, which includes the C1WebExpressConnection and C1WebExpressTable components, offers the same multiple caching options as the Enterprise edition, but without the flexibility of reusable data libraries.

What's New in WebDataObjects for ASP.NET

This documentation was last revised for 2009 v3 on September 21, 2009.

There are no new features in **WebDataObjects for ASP .NET**.

Installing WebDataObjects for ASP.NET

The following sections provide helpful information on installing **WebDataObjects for ASP.NET**.

WebDataObjects for ASP.NET Setup Files

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

Bin	Contains copies of all ComponentOne binaries (DLLs, EXEs).
H2Help	Contains 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 Vista machines:

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

Windows 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.

C1WebDataObjects	Contains samples for **WebDataObjects for ASP.NET**.

System Requirements

System requirements 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) 6.x
Environments:	.NET Framework 2.0 or later
	C# .NET
	Visual Basic .NET
	ASP.NET
	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

Installing Demonstration Versions

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

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

Uninstalling WebDataObjects for ASP.NET

To uninstall **ComponentOne WebDataObjects for ASP.NET**:

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

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.

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 app directory to the target folder (Debug or Release).

3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)

4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:
   ```
         c1lc /target:MyApp.exe /complist:licenses.licx
   /i:C1.Win.C1FlexGrid.dll
   ```

5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:
   ```
   /ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
   ```

6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

Automated testing products that load assemblies dynamically may cause them to display license 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/Support.

Some methods for obtaining technical support include:

- **Online Support via HelpCentral**
  ComponentOne HelpCentral provides customers with a comprehensive set of technical resources in the form of FAQs, samples, Version Release History, Articles, 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.

- **Peer-to-Peer Product Forums and Newsgroups**
  ComponentOne peer-to-peer product forums and newsgroups are available to exchange information, tips, and techniques regarding ComponentOne products. ComponentOne sponsors these areas as a forum for users to share information. While ComponentOne does not provide direct support in the forums and newsgroups, we periodically monitor them to ensure accuracy of information and provide comments when appropriate. 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**
  ComponentOne documentation is installed with each of our products and is also available online at HelpCentral. 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 WebDataObjects for ASP.NET 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.Data.2.dll
- C1.Web.Data.Express.2.dll
- C1.Data.2.dll
- C1.Data.Express.2.dll

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

About This Documentation

Acknowledgements

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

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

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**. The namespaces for the **WebDataObjects for ASP.NET** components are **C1.Web.Data** and **C1.Web.Data.Express**. The following code fragment shows how to declare a C1WebDataSet component using the fully qualified name for this class:

- Visual Basic
```
Dim webdataset As C1.Web.C1WebDataSet.C1WebDataSet
```
- C#
```
C1.Web.C1WebDataSet.C1WebDataSet webdataset;
```

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

For example, if you create a new class named **GlobalCache**, you can use it inside your project without qualification. However, the **C1.Web.Data.2.dll** assembly also implements a class called **GlobalCache**. So, if you want to use C1WebDataObjects' **GlobalCache** class in the same project, you must use a fully qualified reference to make the reference unique. If the reference is not unique, Visual Studio .NET produces an error stating that the name is ambiguous. The following code snippet demonstrates how to declare these objects:

- Visual Basic
```
'   Define a new object based on your GlobalCache class.
Dim MyGlobalCache as GlobalCache

'   Define a new C1.Web.Data.GlobalCache object.
Dim NewGlobalCache as C1.Web.Data.GlobalCache
```
- C#
```
//   Define a new object based on your GlobalCache class.
GlobalCache MyGlobalCache;

//   Define a new C1.Web.Data.GlobalCache object.
C1.Web.Data.GlobalCache newGlobalCache;
```

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 GlobalCache = C1.Web.Data.GlobalCache
Imports MyGlobalCache = MyProject.Objects.GlobalCache

Dim g1 As GlobalCache
Dim g2 As MyGlobalCache
```

- C#

```
using GlobalCache = C1.Web.Data.GlobalCache;
using MyGlobalCache = MyProject.Objects.GlobalCache;

GlobalCache g1;
MyGlobalCache g2;
```

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 ASP.NET 2.0 Project

When creating ASP.NET 2.0 projects, Visual Studio gives you the option of creating a Web site project or a Web application project; the latter is similar to creating a Web project in Visual Studio 2003. The Web application project option was provided to help developers converting Web projects from Visual Studio 2003 to Visual Studio 2005.

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/. See Microsoft's Web site for more detailed information and comparisons on Web site and Web application projects.

The steps for creating both types of projects have been provided for your convenience in the Creating a Web Site Project (page 13) and Creating a Web Application Project (page 14) topics.

Creating a Web Site Project

To create a Web site project, complete the following steps.

1. From the **File** menu in Microsoft Visual Studio, select **New Web Site**. The **New Web Site** dialog box opens.
2. Select **ASP.NET Web Site** from the list of Templates.
3. Enter a URL for your application in the **Location** field and click **OK**.

Note: The Web server must have IIS version 5 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 in the Web Forms Designer in Design view.

4. Double-click a **C1WebDataObjects** component in the **Toolbox** to add it to WebForm1.aspx. For information on adding a component to the Toolbox, see Adding the WebDataObjects for ASP.NET Components to a Project (page 14).

Creating a Web Application Project

To create a new ASP.NET Web application project, complete the following steps.

1. From the **File** menu in Microsoft Visual Studio, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic** or **Visual C#**. Note that one of these options may be located under **Other Languages**.
3. Select **ASP.NET Web Application** from the list of **Templates** in the right pane.
4. Enter a URL for your application in the **Location** field and click **OK**.

    Note: The Web server must have IIS version 5 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 in the Web Forms Designer in Design view.

5. Double-click a **C1WebDataObjects** component in the **Toolbox** to add it to WebForm1.aspx. For information on adding a component to the Toolbox, see Adding the WebDataObjects for ASP.NET Components to a Project (page 14).

Adding the WebDataObjects for ASP.NET Components to a Project

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

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

ComponentOne **WebDataObjects for ASP.NET** provides the following controls:

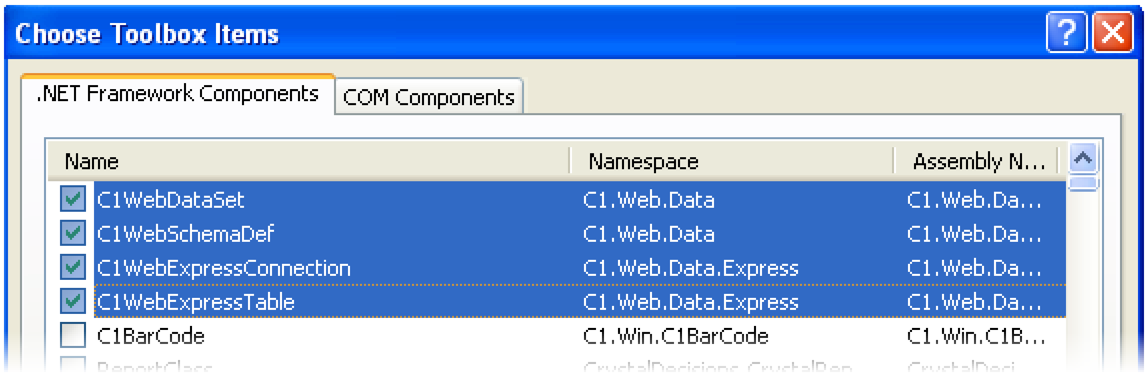
- C1WebDataSet
- C1WebSchemaDef
- C1WebExpressConnection
- C1WebExpressTable

To use **C1WebDataObjects**, add these controls to the form or add a reference to the C1.Web.Data assembly to your project.

Manually Adding C1WebDataObjects Components to the Toolbox

To add the 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. If you want one or more of the **WebDataObjects for ASP.NET** components to appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1WebDataObjects**, for example.
3. Right-click the tab where you want the components 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 components available in the C1.Web.Data or C1.Web.Data.Express namespace. Note that there may be more than one component for a namespace.



Name	Namespace	Assembly N...
✓ C1WebDataSet	C1.Web.Data	C1.Web.Da...
✓ C1WebSchemaDef	C1.Web.Data	C1.Web.Da...
✓ C1WebExpressConnection	C1.Web.Data.Express	C1.Web.Da...
✓ C1WebExpressTable	C1.Web.Data.Express	C1.Web.Da...
C1BarCode	C1.Win.C1BarCode	C1.Win.C1B...

Note: If you do not have **C1DataObjects** components installed in the **Toolbox**, you also need to select components in the C1.Data or C1.Data.Express namespace.

Adding WebDataObjects for ASP.NET Components to the Form

To add one of the components to a form:

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

Adding a Reference to the Assembly

To add a reference to the **C1.Web.Data.2** assembly:

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

```
Imports C1.Web.Data
```

Note: This makes the objects defined in the **C1WebDataObjects** assembly visible to the project. See Namespaces (page 12) for more information.

Migrating a WebDataObjects for ASP.NET Project to Visual Studio 2005

Currently, there is no way to migrate existing projects which use **C1WebDataObjects** components to Visual Studio 2005. You should rewrite your application from scratch using new Web controls, such as **C1WebSchemaDef**, **C1WebDataSet**, **C1WebExpressConnection**, and **C1WebExpressTable**. This is necessary because of the major changes in ASP.NET 2.0 regarding the use of data source controls.

Key Features

ComponentOne WebDataObjects for ASP.NET enhances the **ComponentOne DataObjects for .NET** comprehensive data framework for creating database applications with Web Forms (ASP.NET) functionality.

Note: For full description of **C1DataObjects** functionality, see the ComponentOne DataObjects Help also included with ComponentOne Studio Enterprise.

You can consider your data set objects with all the business logic you associated with them as always existing and functional on the WebForms server in the same way as they exist on a WinForms client. Your data objects in WebForms no longer have to be read-only or otherwise deprived of the rich functionality available in WinForms.

The **WebDataObjects for ASP.NET** key features include:

- **Reusable Business Logic**

  Add complex business logic to one data library assembly to be used in your WinForms and WebForms applications.

- **Server-side Data Caching**

  C1WebDataObjects provides automatic server-side caching so you can store data on the server for quick access.

- **Supports Global Caching**

  Stores information from all user requests while preserving individual user changes, offering quick data access and scalability.

- **Supports Session Caching**

  Provides the ability to store entire data sets, individual user data, and modified data for increased speed and performance.

- **Supports Memory-resident Caching**

  Pre-fetched table views allow immediate access to large amounts of read-only data.

- **Automatic Object Pooling**

  Eliminating the need for continuous internal object creation, an internal pool of worker objects is automatically maintained by **C1WebDataObjects** to optimize performance.

Using WebDataObjects for ASP.NET Components

WebDataObjects for ASP.NET component classes are derived from **ComponentOne DataObjects for WinForms** classes. Although you can use **DataObjects for WinForms** components in WebForms, you will not benefit from WebForm-specific functionality such as automatic caching unless you use **C1WebDataObjects**.

WebDataObjects for ASP.NET contains separate components for use with Enterprise edition and Express edition of **DataObjects for WinForms**. If you use Enterprise edition only, you need one assembly, C1.Web.Data.2.dll, in addition to C1.Data.2.dll. If you use Express edition, you need two assemblies, C1.Web.Data.2.dll and C1.Web.Data.Express.2.dll in addition to C1.Data.2.dll and C1.Data.Express.2.dll.

Enterprise Edition Components

There are two Web components for Enterprise edition: C1WebDataSet and C1WebSchemaDef. Using C1WebDataSet on a Web page enables the Web functionality enhancement such as automatic caching. C1WebSchemaDef hosts a Schema object that is the basis of a direct client configuration in ASP.NET 2.0 projects.

In direct client configuration, you connect C1WebDataSet to a **C1SchemaDef** component representing a data schema residing on the same Web page with your data set.

In a data library configuration, which is the standard way to use **C1DataObjects**, you build your data library regardless of whether it will be used in WinForms or in WebForms. The data library contains data description and business logic code. Both are used in WinForms and WebForms, usually without any change. The ability to specify business logic independently and then use it in WebForms and in WinForms, using the same data library assembly, is a powerful feature of **C1WebDataObjects**.

Since a data library is the same for WinForms and WebForms, you can use the standard **C1SchemaDef** component in your data library. However, **C1DataObjects** includes special property settings in the data schema, accessible in the **Schema Designer**, for tuning caching behavior. These settings are only active when the data library is used with **C1WebDataObjects**. You can see the CacheProperties object with tuning properties on the **Properties** tab of a **DataSet Editor** in the **Schema Designer**. These are the settings on data set level. There is also a Boolean property on table view level, MemoryResident that allows you to make individual read-only table views pre-fetched and resident in memory. Finally, there is a SessionCacheProperties object accessible as a property of **C1SchemaDef**, that you can use to set default argument values for session cache method calls.

In direct client configuration, you also use **C1SchemaDef** with the same cache settings in the schema and in the **C1SchemaDef** component. Use this configuration only if you don't need to reuse the schema and logic and decide to put it on a Web page together with C1WebDataSet. Consider creating a data library if there is a chance your business logic will be reused on more than one page or with different clients, such as WinForms.

To make caching work properly, you need to include some code, certain method calls, usually very simple, in your Web page. These methods are static members of two classes, GlobalCache and SessionCache. These two classes have no instantiated objects, they are just collections of static methods (functions) that you use to control the cache.

Express Edition Components

As in regular **C1DataObjects**, Express edition components in **WebDataObjects for ASP.NET** can be used in cases where you don't need independent and reusable business logic and ease of use is more important to you than functional richness. Express components reside directly on your Web forms (Web pages) and cannot be reused across Web pages.

Working with Express edition, you use C1WebExpressTable derived from **C1ExpressTable**, and C1WebExpressConnection derived from **C1ExpressConnection**. Put a **C1WebExpressConnection** and one or more **C1WebExpressTable** components on your Web form and connect the **C1WebExpressTable** components to the **C1WebExpressConnection**. Unlike C1ExpressTable, **C1WebExpressTable** must always be connected to a **C1WebExpressConnection**.

You can also use the **C1ExpressView** component that belongs to the regular **C1DataExpress**. **C1ExpressView** does not have Web-specific functionality, so there was no need to create a special counterpart for it in **C1WebDataObjects**, the regular **C1ExpressView** is enough.

C1WebExpressConnection has special object properties called CacheProperties and SessionCacheProperties containing settings for tuning caching behavior. These are the settings on data set level. You can tune CacheProperties if you use global cache and SessionCacheProperties if you use session cache. There is also a Boolean property on table level, MemoryResident that allows you to make individual read-only tables pre-fetched and resident in memory.

To make caching work properly, you need to include some code, certain method calls, usually very simple, in your Web page. These methods are static members of two classes, GlobalCache and SessionCache. These two classes have no instantiated objects, they are just collections of static methods (functions) that you use to control the cache.

Global Cache

The global cache is the default caching mode, it is used by default for every **WebDataObjects for ASP.NET** data set if you don't change cache settings and don't call SessionCache methods. To disable caching for a data set completely, set CacheMode = **NotCached**.

The global cache is a common storage on the server for your Web application serving all requests from all users while preserving the changes made by an individual user. When a data set is filled with data from database, the fetched data is stored in the cache for later use, together with information about user request, such as the filter condition used to fetch data. When another request for the same data set with the same filter condition arrives from the same or another user later on, cached data is used to fill the data set instead of fetching data from the database.

If the user modifies data, the global cache stores user modifications and then applies them automatically to the cached or fetched data next time a request arrives from the same user, to preserve the consistency of this user's view of modified data. The users' responsibility is to call SaveChanges after they are done with modifying data; this enables global cache to save this user's changes for future use.

Comparing with session cache, the global cache has the advantage of scalability, because data is stored without duplication. Same data fetched by multiple users is stored only once. It can also improve performance, because when one user fetches some data and then the same request (with the same filter condition if one is specified) arrives from a different user, the second user's request will likely be served by the cache, without database fetch. The global cache is particularly effective when you have relatively large data sets (so it is not practical to store them in the session cache) with modifiable tables and filter conditions used in requests are likely to repeat themselves. Read-only (non-modifiable) tables are best represented as memory-resident, see Memory-resident Table Views (page 20), but you can make one or more table views memory-resident and store the rest in the global cache (or in the session cache). You can also store some data sets in the global cache, some in the session cache, and make others not cacheable.

On the other hand, the global cache has a disadvantage (or rather a feature that can be detrimental in some cases, though neutral or even beneficial in others) with respect to the session cache in that the data received by the same user can change between requests due to changes in the database made by other users. In the session cache, each user's data set is stored separately, so they don't feel any changes made by others until the whole data set is re-fetched from the database. In the global cache, a user is guaranteed to receive the same data for same requests until another user updates the database. At the time of database update, all global cache data pertaining to this data set is cleared, so it is fetched from the database next time it is requested by any user. However, it is important to understand that this only affects rows that have not been changed by the user. The global cache guarantees that users always see modified rows intact, cannot lose their modifications under any circumstances. The fact that unmodified rows can change between requests simply means that the user sees the most recent rows from the database, excluding rows modified by the same user, which appear locked, owned by the user until they are committed to the database.

Session Cache

The session cache stores every user's data set separately on the server. There are many advantages to using the session cache, including:

- **Ability to increase speed and gain performance**

  The session cache is most beneficial when it is used in Memory storage mode. In this mode, the whole data set object is stored in memory in a ready filled state, so it is immediately reused on postbacks and does not need to be filled with data. The performance gain from using this mode can be dramatic, since no data is copied, even in memory, on saving the data set, and there is virtually no delay for data retrieval. However, Memory storage mode has some drawbacks. It consumes server memory. Also, the immediate, without copying data, storage and retrieval in Memory mode take effect only if you configure your application for in-process mode (use **SessionStateMode = InProc**).

In other session state modes, SqlServer and StateServer, the data is first copied to a persistent object in memory, which is also fast, depending on the data size. If you consider using the session cache to improve performance, weigh it against the possibility of users' data taking up too much server memory. To conserve server memory, you can use File storage mode with session cache. This mode is not as fast as Memory, but it is still about two times faster than the global cache. If you want greater scalability and do not want to allocate server resources for individual users, consider using the global cache or combine session or global cache with memory-resident table views.

- **Ability to store data separately**

  Another possible reason for using the session cache is when it is important that each user's data are stored separately on the server so they are completely insulated from other users' changes. If you make a decision to use session cache for that reason, make sure you really need complete isolation from other users' changes, which in effect means creating a snapshot of data for every user. It is often not necessary and even undesirable to work with such snapshots. Using the global cache might be a better option because it ensures that once you modify a row, it won't change, and unmodified rows changing in accordance with database changes is often acceptable and indeed desirable. Still, if you really need snapshots, use the session cache.

- **Ability to effectively store modified data**

  Also, the session cache can be recommended if a considerable part of the data in the data set, say 50%, is expected to be modified by the user. In this case, the session cache is preferable because it stores modified data as efficiently as unmodified, whereas the global cache stores data modifications separately for each user (thus consuming server resources for this purpose no less than session cache does) and furthermore, the global cache performs special actions on each modified row restoring the data set on postback. This is not a performance drain if modified rows are relatively few, but it becomes essential if they constitute a considerable part of the overall row count.

Unlike the global cache, the session cache is never used automatically. You call SaveData to store data in session cache and Fill to retrieve data from the session cache.

As with global cache, you can make one or more table views memory-resident and store the rest in the session cache. You can also store some data sets in the global cache, some in the session cache, and make others not cacheable.

Cache Storage

Cache storage mode can be specified in CacheProperties (for global cache) and in **SessionCacheProperties** (for session cache, default that can be overridden in session cache method calls). It is one of the two options in the CacheStorageModeEnum enumeration: Memory or File.

Cached data can be stored in memory (option: Memory). This option provides slightly better performance, but takes up server memory resources. It is less a concern with the global cache because the global cache holds only limited number of cached data set instances (see the **MaxCount** property). With the session cache, Memory is a less frequently used option because session cache instances remain alive as long as the user session is alive, so using the session cache for large data sets (not counting memory-resident table views) with large number of users can use up server memory.

The default option for the global cache is File. File storage is available for the session cache as well. In this case data is stored in binary files on the server, a file per data set instance, in a directory that can be specified in cache tuning properties.

The default option for session cache is Memory. In this case the whole data set object is stored in memory in ready filled state, so it is immediately reused on postbacks, does not need to be filled with data at all. Performance gain can be dramatic, since there is virtually no delay for data retrieval. However, Memory storage mode consumes server memory. Also, the immediate, without copying data, storage and retrieval in Memory mode take effect only if you configure your application for in-process mode (use SessionStateMode = InProc). In other session state modes, SqlServer and StateServer, the data is first copied to a persistent object in memory, which is also fast, but depends on the data size.

Memory-Resident Table Views

In Enterprise edition, you can specify a table view as memory-resident setting MemoryResident = True. In Express edition you can make a C1WebExpressTable memory-resident setting MemoryResident = **True**. In any data set, you can make one or more table views memory-resident while storing the rest in the global cache or in the session cache. Memory-resident table views are automatically excluded from global and session cache operations.

A memory-resident table view is fetched from the database only once, when it is first requested, and then it will be just reused on every user request, for any user, at no performance cost.

This is, of course, a powerful technique to optimize performance, but it can only be used for read-only tables. A memory-resident table view must be read-only (TableView.**ReadOnly = True**) and it can be based only on read-only tables (Table.**ReadOnly = True**).

And another restriction applies to memory-resident table views: you can't specify different filter conditions for them calling **C1DataSet.Fill**. Memory-resident table views are always in memory, they are skipped when data is fetched in **C1DataSet.Fill**, except when it is the first fetch of the application. Therefore, you can't change their filter conditions, they need to be constant. The filter condition will take effect only once for the application, in the first **Fill**.

More precisely, a memory-resident table can be fetched more than once, due to concurrency issues. This occurs when a user requests the table view while it is being used by another user request. In this case, if the object pool is not full, a new worker object is created (see Object Pooling (page 20)), added to the pool, and the memory-resident table view is fetched to that worker object. So, a memory-resident table views belongs to a worker object, and maximum number of times it is fetched is the maximum worker object count in the pool, which is specified by the PoolSize property.

Object Pooling

Another optimization technique used by **C1WebDataObjects** is object pooling. Object pooling is automatic, you don't need to write code or set properties to benefit from it. Thereare, however, a few properties that you might want to set to fine-tune object pooling.

Object pooling eliminates the need for **C1WebDataObjects** to create a new internal object each time to service a new request. Creating an internal object (called *worker object*) has a performance cost, although not particularly high This performance cost is usually a small fraction of a second, but is perceptible nevertheless.

C1WebDataObjects maintains an internal pool of worker objects, which once created is never destroyed until the application process is shut down. The number of worker objects in the pool is limited by the PoolSize property. The first worker object is created when the first user makes a request for data. Then new worker objects are created and added to the pool on a need-to-serve basis. When a new request arrives, **C1WebDataObjects** first looks for a free worker object in the pool to service the request. If all worker objects are busy and the PoolSize limit has not yet been reached, a new worker object is created and added to the pool. If the pool is full, reached its maximum size, and all worker objects are busy, then the new request will wait until one of the busy worker objects finishes its job and becomes free to service the new request. This waiting time is limited by the PoolTimeout property. After the timeout expires, **C1WebDataObjects** throws an exception saying that it is unable to service the request. By default, this timeout is 0, which means that no timeout limit is imposed, the request will wait indefinitely until one of the worker processes becomes free, and the exception is never thrown. Setting non-zero PoolTimeout can make sense for cases where it is possible for a request (through user code) to hang indefinitely and you don't want this to affect other requests by making them wait indefinitely while all pooled worker processes are hanging (a rather hypothetical situation, but theoretically possible).

Object pooling can be disabled by setting PoolSize = 0. Then a new worker object is created for each new request and destroyed when it finishes serving the request.

Using WebDataObjects Enterprise Edition

The following topics explain how to use global cache, session cache, and memory-resident table views in the Enterprise version of **C1WebDataObjects**.

Using the Global Cache in Enterprise Edition

Using the global cache is almost completely automatic. By default, each data set in the schema has CacheMode = **InWebForms**, which means that it uses the global cache.

To disable caching, set CacheMode = **NotCached**.

You may need to specify cache properties for global cache tuning. This is done for each data set on the Properties tab of the data set editor in the schema designer. If you select **CacheStorage = File** (default), especially in production settings, you may want to increase the value of the **MaxCount**, 50 by default, limiting the number of data set instances in the cache.

With **CacheStorage = File**, you will usually set the CacheStoragePath property to point to a directory where you will store cache files. If you use a multi-server configuration (Web farm), you must include a server name in the path to ensure that all cache files are stored on the same server (usually called state server). If you fail to provide a server name, data will be cached on the server that is serving current user request. That can cause data inconsistency when requests are assigned to different servers. By default, if you leave CacheStoragePath empty, the temporary directory of the current server is used.

If you use CacheStoragePath = File, you will also usually set the ChangesCacheStoragePath property to specify a directory where you store files containing saved user data modifications. This is a part of the global cache necessary if users are allowed to modify data. The same considerations regarding server name in the path apply to ChangesCacheStoragePath as to CacheStoragePath.

Other than tuning cache properties, which is optional, the global cache works automatically. It is used every time you call **C1WebDataSet.Fill**. Filling the data set, **C1WebDataObjects** first looks for data in the cache and performs database fetch only if not found in the cache.

The only additional code you must include in your application to support global is one line in your Global.aspx file:

- Visual Basic

```
Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
      GlobalCache.DeleteFiles(Me)
End Sub
```

- C#

```
protected void Application_End(Object sender, EventArgs e)
{
  GlobalCache.DeleteFiles(this);
}
```

This is necessary to clean up the cache files when your application is shut down. It is necessary only if you use CacheStorage = File, but it is a good idea to always call this method in the **Application_End** event.

For additional information on using the global cache and CacheProperties, see Using CacheProperties to Specify Global Cache Settings (page 32) in the task-based help.

Using the Session Cache in Enterprise Edition

You can use the global cache for some data sets and the session cache for others. You don't need to change any settings in order to use the session cache. The session cache is used by calling SessionCache methods.

The usual pattern is to replace **C1DataSet.Fill** calls with the following:

- Visual Basic

```
If Not SessionCache.Fill(c1DataSet1) Then
        c1DataSet1.Fill()
    SessionCache.SaveData(c1DataSet1)
End If
```

- C#

```
if (!SessionCache.Fill(c1DataSet1))
{
      c1DataSet1.Fill();
      SessionCache.SaveData(c1DataSet1);
}
```

This code first tries to fill the data set with data from the session cache. If this is executed for the first time in the user session, then the session cache does not yet contain data c1DataSet1, and the data is fetched from the database with **C1WebDataSet1.Fill()**. After fetch, we save the data in the session cache for future use.

Note: Make sure that **C1WebDataSet1.FillOnRequest** is set to **False**, to prevent C1WebDataSet from filling the data set automatically when you bind controls to it.

If you modify data in a data set that uses the session cache, you are responsible for saving the modified data set in the session cache if you need (and you usually do) the modifications to be persistent, to be restored next time you restore the data set from session cache. Usually, you update the session cache after you are done modifying the data set by calling the **SaveData** method:

- Visual Basic

```
SessionCache.SaveData(c1DataSet1)
```

- C#

```
SessionCache.SaveData(c1DataSet1);
```

SessionCache.SaveData has a cacheStorage argument that is one of two possible values: **File** or **Memory**. If the argument is omitted, the value specified in C1SchemaDef.**SessionCacheProperties.DefaultStorage** is used. The default is **Memory**.

If you use **cacheStorage = File**, you will usually need to set the **SessionCacheProperties.StoragePath** property of the C1SessionDef component to point to a directory where your session cache files are stored. If you use a multi-server configuration (Web farm), you must include a server name in the path to ensure that all cache files are stored on the same server (usually called state server). If you fail to provide a server name, data will be cached on the server that is serving current user requests. That can cause data inconsistency when requests are assigned to different servers. By default, if you leave **SessionCacheProperties.StoragePath** empty, the temporary directory of the current server is used.

Finally, if you use **cacheStorage = File**, you must include a clean-up line in your application Global.aspx file:

- Visual Basic

```
Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
  SessionCache.DeleteFiles(Me)
End Sub
```

- C#

```
protected void Session_End(Object sender, EventArgs e)
{
  SessionCache.DeleteFiles(this);
}
```

This is necessary to clean up the cache files when a session is closed. It is necessary only if you use cacheStorage = File, but it is a good idea to always call this method in the Session_End event.

For additional information on using the session cache, see Using the Session Cache (page 31) in the WebDataObjects for ASP.NET Task-Based Help (page 28).

Using Memory-Resident Table Views in Enterprise Edition

Making large table views memory-resident can dramatically improve performance. To make a table view memory-resident, ensure the following:

- It is read-only.

  AND

- You always specify the same filter condition for this table view or no condition at all (filter condition will take effect only once for the application, in the first **Fill**).

In the **Schema Designer**, open the data set containing the table view, select the table view on the Diagram tab, select MemoryResident property in the property grid below, set **MemoryResident = True**. This is all that is needed to make a table view memory-resident.

Using WebDataObjects for ASP.NET Express Edition

The following topics explain how to use the global cache, the session cache, and memory-resident table views in the Express edition of **ComponentOne WebDataObjects for ASP.NET**.

Using the Global Cache in Express Edition

Using the global cache is almost completely automatic. By default, a C1WebExpressConnection component has CacheMode = **InWebForms**, which means that it uses the global cache.

To disable caching, set CacheMode = **NotCached**.

You may need to specify cache properties for global cache tuning. This is done in the CacheProperties property. If you select CacheStorage = **File** (default), especially in production settings, you may want to increase the value of the MaxCount, 50 by default, limiting the number of data set instances in the cache.

With CacheStorage = **File**, you will usually set the CacheStoragePath property to point to a directory where you will store cache files. If you use a multi-server configuration (Web farm), you must include a server name in the path to ensure that all cache files are stored on the same server (usually called state server). If you fail to provide a server name, data will be cached on the server that is serving the current user request. That can cause data inconsistency when requests are assigned to different servers. By default, if you leave CacheStoragePath empty, the temporary directory of the current server is used.

If you use ChangesCacheStorage = **File**, you will also usually set the ChangesCacheStoragePath property to specify a directory where you store files containing saved user data modifications. This is a part of the global cache necessary if users are allowed to modify data. The same considerations regarding server name in the path apply to ChangesCacheStoragePath as to CacheStoragePath.

Other than tuning cache properties, which is optional, the global cache works automatically. It is used every time you call C1WebExpressConnection.Fill. Filling the data set, **C1WebDataObjects** first looks for data in the cache and performs database fetch only if it is not found in the cache.

The only additional code you must include in your application to support global is one line in your Global.aspx file:

- Visual Basic

```
Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
      GlobalCache.DeleteFiles(Me)
End Sub
```

- C#

```
protected void Application_End(Object sender, EventArgs e)
{
GlobalCache.DeleteFiles(this);
}
```

This is necessary to clean up the cache files when your application is shut down. It is necessary only if you use CacheStorage = File, but it is a good idea to always call this method in the Application_End event.

Using the Session Cache in Express Edition

You can use the global cache for some C1WebExpressConnection components and session cache for others. You do not need to change any settings in order to use the session cache. The session cache is used by calling SessionCache methods.

The usual pattern is to replace C1WebExpressConnection.Fill calls with the following:

- Visual Basic

```
If Not SessionCache.Fill(c1WebExpressConnection1) Then
  c1WebExpressConnection1.Fill()
  SessionCache.SaveData(c1WebExpressConnection1)
End If
```

- C#

```
if (!SessionCache.Fill(c1WebExpressConnection1))
{
      c1WebExpressConnection1.Fill();
      SessionCache.SaveData(c1WebExpressConnection1);
}
```

This code first tries to fill the data set with data from the session cache. If this is executed for the first time in the user session, then the session cache does not yet contain data c1WebExpressConnection1, and the data is fetched from the database with c1WebExpressConnection1.Fill(). After fetch, we save the data in the session cache for future use.

Note: Make sure that c1WebExpressConnection1.FillOnRequest is set to **False**, to prevent **C1WebDataObjects** from filling the data set automatically when you bind controls to it.

If you modify data in a data set that uses session cache, you are responsible for saving the modified data set in the session cache if you need (and you usually do) the modifications to be persistent, to be restored the next time you restore the data set from session cache. Usually, you update the session cache after you are done with modifying the data set, calling:

- Visual Basic

```
SessionCache.SaveData(c1WebExpressConnection1)
```

- C#

```
SessionCache.SaveData(c1WebExpressConnection1);
```

SaveData has a cacheStorage argument that is one of two possible values: **File** or **Memory**. If the argument is omitted, the value specified in **SessionCacheProperties.DefaultStorage** is used. The default is **Memory**.

If you use **cacheStorage = File**, you will usually need to set the **SessionCacheProperties.StoragePath** property of the c1WebExpressConnection1 component to point to a directory where your session cache files are stored. If you use a multi-server configuration (Web farm), you must include a server name in the path to ensure that all cache files are stored on the same server (usually called state server). If you fail to provide a server name, data will be cached on the server that is serving the current user request. That can cause data inconsistency when requests are assigned to different servers. By default, if you leave **SessionCacheProperties.StoragePath** empty, the temporary directory of the current server is used.

Finally, if you use **cacheStorage = File**, you must include a clean-up line in your application Global.aspx file:

- Visual Basic

```
Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
      SessionCache.DeleteFiles(Me)
End Sub
```

- C#

```
protected void Session_End(Object sender, EventArgs e)
{
SessionCache.DeleteFiles(this);
}
```

This is necessary to clean up the cache files when a session is closed. It is necessary only if you use cacheStorage = File, but it is a good idea to always call this method in the **Session_End** event.

Using Memory-Resident Table Views in Express Edition

Making large tables memory-resident can dramatically improve performance. To make a table memory-resident, make sure its settings are as follows:

- Verify that it is read-only (C1WebExpressTable.**ReadOnly = True**).
- Do not change the **FillFilter** property for this table during user session (FillFilter condition will take effect only once for the application, in the first Fill).
- Set MemoryResident property to **True**.

Design-Time Support

WebDataObjects for ASP.NET provides customized context menus, smart tags, and a designer that offers rich design-time support and simplifies working with the object model. The following sections describe how to use C1WebDataObjects' design-time environment to configure the **WebDataObjects for ASP.NET** controls.

Tasks Menus

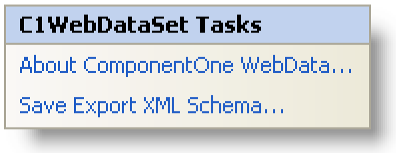
A smart tag represents a short-cut tasks menu that provides the most commonly used properties in each control. You can invoke each coontrol's tasks menu by clicking on the smart tag (▶) in the upper-right corner of the control.

Properties Window

You can also easily configure C1WebDataObjects at design time using the Properties window in Visual Studio. You can access the Properties window by right-clicking the control and selecting **Properties**.

C1WebDataSet Component

You can access the **C1WebDataSet Tasks** menu by clicking the smart tag in the upper-right corner of the C1WebDataSet component.



You can access the C1WebDataSet context menu by right-clicking the C1WebDataSet component.

About ComponentOne WebDataObjects

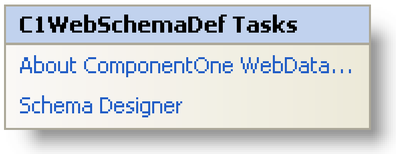
Clicking **About** displays the C1WebDataObjects' **About** dialog box, which is helpful in finding the build number of the control.

Save Export XML Schema

Clicking **Save Export XML Schema** opens the **Save schema to a file** dialog box which you can use to export data from a C1WebDataSet to XML.

C1WebSchemaDef Component

You can access the **C1WebSchemaDef Tasks** menu by clicking the smart tag in the upper-right corner of the C1WebSchemaDef component.



You can access the C1WebSchemaDef context menu by right-clicking the C1WebSchemaDef component.

About ComponentOne WebDataObjects

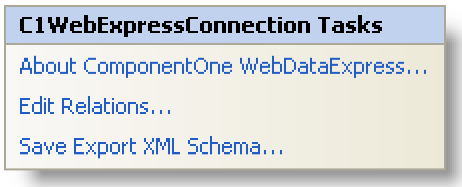
Clicking **About** displays the C1WebDataObjects' **About** dialog box, which is helpful in finding the build number of the control.

Schema Designer

Clicking **Schema Designer** opens the **ComponentOne WebDataObjects Schema Designer**. A Schema is the basis and starting point of C1WebDataObjects development. It contains data structure information, defining basic entities, such as tables and relations, with their properties.

C1WebExpressConnection Component

You can access the **C1WebExpressConnection Tasks** menu by clicking the smart tag in the upper-right corner of the C1WebExpressConnection component.


C1WebExpressConnection Tasks
About ComponentOne WebDataExpress...
Edit Relations...
Save Export XML Schema...

You can access the C1WebExpressConnection context menu by right-clicking the C1WebExpressConnection component.

About ComponentOne WebDataObjects
Clicking **About** displays the C1WebDataObjects' **About** dialog box, which is helpful in finding the build number of the control.

Edit Relations

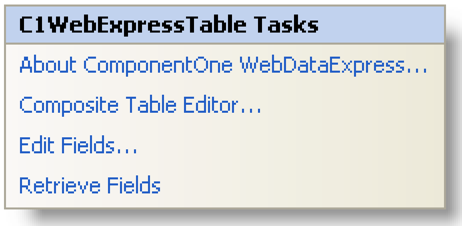
Clicking **Edit Relations** opens the Relations window where you can add, delete, and edit relations.

Save Export XML Schema

Clicking **Save Export XML Schema** opens the **Save schema to a file** dialog box which you can use to export data from a C1WebExpressConnection to XML.

C1WebExpressTable Component

You can access the **C1WebExpressTable Tasks** menu by clicking the smart tag in the upper-right corner of the C1WebExpressTable component.


C1WebExpressTable Tasks
About ComponentOne WebDataExpress...
Composite Table Editor...
Edit Fields...
Retrieve Fields

You can access the C1WebExpressTable context menu by right-clicking the C1WebExpressTable component.

About ComponentOne WebDataObjects
Clicking **About** displays the C1WebDataObjects' **About** dialog box, which is helpful in finding the build number of the control.

Composite Table Editor

Clicking **Composite Table Editor** opens the **Composite Table Editor** where you can add, delete, and edit composite tables and joins.

Edit Fields

Clicking **Edit Fields** opens the **Fields** window where you can add, delete, and edit DB fields and Calculated fields.

Retrieve Fields

Clicking **Retrieve Fields** retrieves table view fields from a table specified in the **Table** property.

WebDataObjects for ASP.NET 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 ComponentOne Studios.

Note: The ComponentOne Samples are also available at http://helpcentral.componentone.com/Samples.aspx.

Visual Basic and C# Samples

Sample	Description
DataOnTheWeb	Shows how to use C1WebDataObjects for building Web applications allowing showing and updating data from SQL server. This sample uses the C1WebDataSet component.
DataOnTheWebExpress	Shows how to use C1WebDataExpress for building Web applications allowing showing and updating data from SQL server. This sample uses the C1WebExpressConnection and C1WebExpressTable components.
UsingMemoryResident	Shows how to use memory-resident table views in C1WebDataObects, using a memory-resident table to validate an ID entered by the user. This sample uses the C1WebDataSet component.
UsingMemoryResidentExpress	Shows how to use memory-resident table views in C1WebDataExpress, using a memory-resident table to validate an ID entered by the user. This sample uses the C1WebExpressConnection and C1WebExpressTable components.
UsingSessionCache	Demonstrates usage of session cache to achieve isolated data representation for each user in C1WebDataObjects. This sample uses the C1WebDataSet component.
UsingSessionCacheExpress	Demonstrates usage of session cache to achieve isolated data representation for each user in C1WebDataExpress. This sample uses the C1WebExpressConnection and C1WebExpressTable components.

WebDataObjects for ASP.NET Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET, and know how to use bound controls in general. By following the steps outlined in the help, you will be able to create projects demonstrating a variety of **C1WebDataObjects** features and get a good sense of what **C1WebDataObjects** can do.

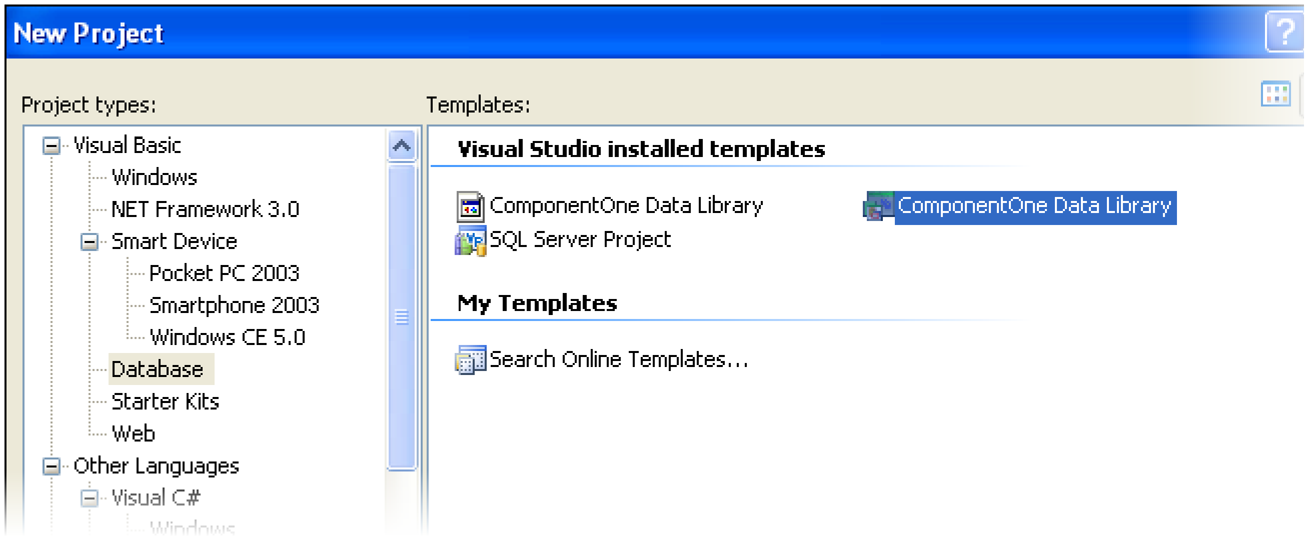
Each task-based help topic also assumes that you have created a new ASP.NET project. For additional information on this topic, see Creating an ASP.NET 2.0 Project (page 13).

Creating a Data Library

To create a ComponentOne data library in Visual Studio, complete the following steps:

1. Select **File | New Project**. The **New Project** dialog box appears.
2. Under **Project types**, choose either **Visual Basic** or **Visual C#**, according to your language preference. Note that one of these options may be located under **Other Languages**.

3. Expand the **Database** node under your selected programming language.
4. Select *ComponentOne Data Library* from the list of **Templates** in the right pane.

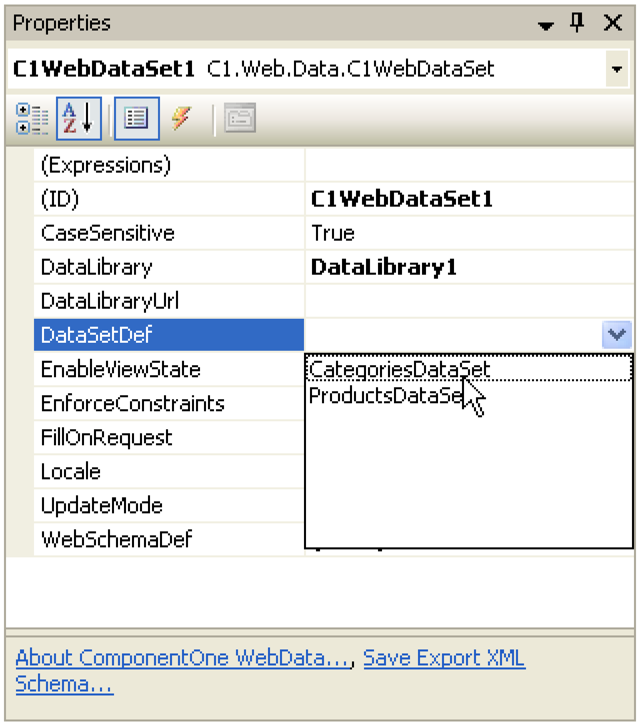


5. Enter a project name in the **Name** text box, specify a location and press click **OK**.
6. Use the C1DataObjects **Import Wizard** and **Schema Designer** to configure your schema.
7. Select **Build DataLibraryName** from the **Build** menu to produce the data library assembly.

Using C1WebDataSet with a Data Library

To assign a data library to a C1WebDataSet component which is bound to a grid, complete the following steps.

1. Create and compile a new Data Library project containing at least one data set definition.
2. Create a new ASP.NET project.
3. Add a reference to the new Data Library:
   a. In the **Website** menu, select **Add Reference**.
   b. Browse to the bin folder of the new Data Library project.
   c. Select the new library's .dll and click **OK**. This adds a reference to the DataLibrary.dll to your project.
4. Double-click C1WebDataSet from the Toolbox to add a C1WebDataSet component to the form.
   - Set the **DataLibrary** property of the C1WebDataSet to the new Data Library.
   - Select a data set definition from the dropdown box next to the **DataSetDef** property.


Properties

C1WebDataSet1 C1.Web.Data.C1WebDataSet

(Expressions)	
(ID)	C1WebDataSet1
CaseSensitive	True
DataLibrary	DataLibrary1
DataLibraryUrl	
DataSetDef	
EnableViewState	
EnforceConstraints	
FillOnRequest	
Locale	
UpdateMode	
WebSchemaDef	

CategoriesDataSet
ProductsDataSe

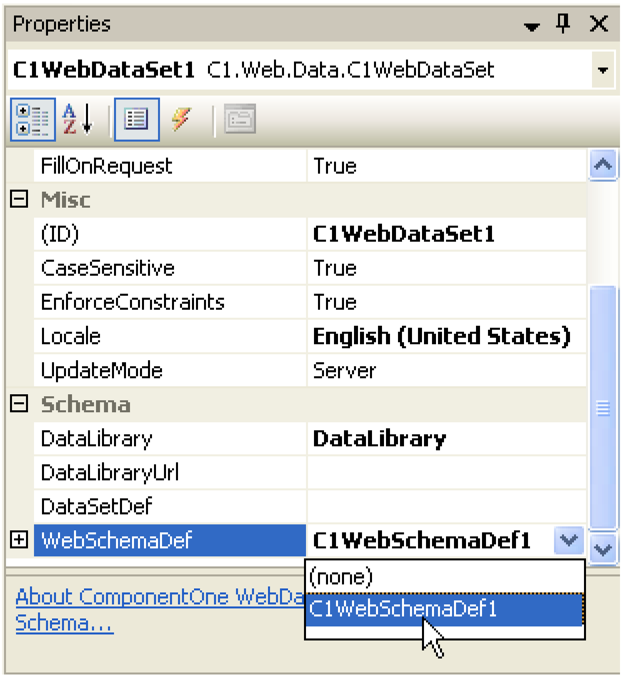
About ComponentOne WebData..., Save Export XML Schema...

5. Add a GridView component to the form.
   - Set the **DataSourceID** property to **C1WebDataSet**.
   - Set the **DataMember** property to the desired table.

Using C1WebDataSet in Direct Client Configuration

To connect C1WebDataSet to a C1WebSchemaDef for direct client configuration, complete the following steps.

1. Create a new ASP.NET project and a C1WebDataSet component to the Web form.
2. Double-click **C1WebSchemaDef** from the **Toolbox** to add a **C1WebSchemaDef** component to the form.
3. Use the C1DataObjects **Import Wizard** and **Schema Designer** to configure your schema.
4. Select **C1WebDataSet1**.
   - Set the **WebSchemaDef** property to **C1WebSchemaDef1**.


Properties

C1WebDataSet1 C1.Web.Data.C1WebDataSet

FillOnRequest	True
Misc	
(ID)	**C1WebDataSet1**
CaseSensitive	True
EnforceConstraints	True
Locale	**English (United States)**
UpdateMode	Server
Schema	
DataLibrary	**DataLibrary**
DataLibraryUrl	
DataSetDef	
WebSchemaDef	**C1WebSchemaDef1**

(none)
C1WebSchemaDef1

About ComponentOne WebDa
Schema...

- Select a data set definition from the dropdown box next to the **DataSetDef** property.

5. Add a DataGrid component to the form.

- Set the **DataSourceID** property to C1WebDataSet1.
- Set the **DataMember** property as desired.

Using the Session Cache

To find data in a session cache and save changes there when data is changed, complete the following steps. Note that in this example a DataList was created and bound to a C1WebDataSet.

1. Add a **DataList** and C1WebDataSet to your project.
2. Add the following code to your **Page_Load** event to find data in the session cache, if it is there:

- Visual Basic

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    ' try to find data in session cache first
    If Not SessionCache.Fill(C1WebDataSet1) Then
    ' not found in session cache, fetch from the database
      C1WebDataSet1.Fill()
    ' save changes in session cache every time you change data in the
data set
            SessionCache.SaveData(C1WebDataSet1)
    End If

End Sub
```

- C#

```
private void Page_Load(object sender, System.EventArgs e)
    {
        // try to find data in session cache first
        if (!SessionCache.Fill(c1WebEmployeesDataSet))
       {
            // not found in session cache, fetch from the
            // database
            c1WebEmployeesDataSet.Fill();
            // save changes in session cache every time
            // you change data in the data set
            SessionCache.SaveData(c1WebEmployeesDataSet);
        }
    }
```

3. To save changes in session cache every time you change data in the data set, add the following code:

   - Visual Basic

```
Public Sub DataList1_Update(ByVal sender As Object, ByVal e As
DataListCommandEventArgs)
    ' save changes in session cache every time you change
    ' data in the data set
    SessionCache.SaveData(C1WebDataSet1)
End Sub
```

   - C#

```
public void DataList1_Update(Object sender, DataListCommandEventArgs e)
    {
        // save changes in session cache every time you
        // change data in the data set
        SessionCache.SaveData(c1WebEmployeesDataSet);
        DataList1.EditItemIndex = -1;
        BindList();
    }
```

4. Clean up the cache files when a session is closed. Enter the following code in the Global.asax:

   - Visual Basic

```
Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
SessionCache.DeleteFiles(Me)
End Sub
```

   - C#

```
protected void Session_End(Object sender, EventArgs e)
{
SessionCache.DeleteFiles(this);
}
```

Sample Available

For additional information see the **UsingSessionCache** sample which is available for download from the ComponentOne HelpCentral Sample page.

Using CacheProperties to Specify Global Cache Settings

The CacheProperties class is used to specify global cache settings for a DataSetDef class using its CacheProperties property.

Global cache is the default caching mode. It is used by default for every WebDataObjects for ASP.NET data set if you do not change the cache settings and do not call SessionCache methods.

To disable caching for a data set completely, set CacheProperties.CacheMode = **NotCached**.

Here is some example code that can be used with global cache.

1. Enter the following in the **Page_Load** event handler:

   - Visual Basic

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    ' c1WebProductsDataSet (ProductsDataSet) uses global
    ' cache
    ' See in Schema Designer: SessionProperties.CacheMode =
    'InWebForms.
    c1WebProductsDataSet.Fill() 'global cache works
    'automatically when you call C1WebDataSet.Fill

End Sub
```

   - C#

```
private void Page_Load(object sender, System.EventArgs e)
    {
        // c1WebProductsDataSet (ProductsDataSet) uses global
        //cache
        // See in Schema Designer:
        //SessionProperties.CacheMode = InWebForms.
        c1WebProductsDataSet.Fill();
    // global cache works automatically when you call
    // C1WebDataSet.Fill
    }
```

2. Save changes in the global cache every time you change data in the data set:

   - Visual Basic

```
Public Sub DataGrid1_Update(ByVal source As Object, ByVal e As
System.Web.UI.WebControls.DataGridCommandEventArgs)

    GlobalCache.SaveChanges(c1WebProductsDataSet) ' save
    ' changes in global cache every time you change data in
    'the data set

End Sub
```

   - C#

```
public void DataGrid1_Update(object source,
System.Web.UI.WebControls.DataGridCommandEventArgs e)
    {
        GlobalCache.SaveChanges(c1WebProductsDataSet);
        // save changes in global cache every time you change
        // data in the data set
    }
```

3. If you need to refresh data from the database, the global cache should be cleared:

   - Visual Basic

```
Private Sub btnRefresh_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles btnRefresh.Click
    ' We need to clear global cache first to ensure the data is fetched
from the database, not from cache.
    GlobalCache.Remove(c1WebProductsDataSet)
    ' fetch data from the database
    c1WebProductsDataSet.Fill()
    ViewState.Remove("SortExpression")
    BindGrid()
End Sub
```

- C#

```
private void btnRefresh_Click(object sender, System.EventArgs e)
    {
        // We need to clear global cache first to ensure the
        // data is fetched from the database, not from cache.
        GlobalCache.Remove(c1WebProductsDataSet);
        // fetch data from the database
        c1WebProductsDataSet.Fill();
        ViewState.Remove("SortExpression");
        BindGrid();
    }
```

4. Clean up the cache files when your application is shut down. Enter the following code in the Global.asax:

- Visual Basic

```
Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
    GlobalCache.DeleteFiles(Me)
End Sub
```

- C#

```
protected void Application_End(Object sender, EventArgs e)
   {
        GlobalCache.DeleteFiles(this);
   }
```

Sample Available

For additional information see the **DataOnTheWeb** sample, which is available for download from the ComponentOne HelpCentral Sample page.

Adding References in the Web.Config file

This topic demonstrates how to add references to **WebDataObjects for ASP.NET** assemblies in a Web.config file.

If your assembly is in the GAC and not in the bin, ASP.NET will load your assembly at run time only if you include it in a Web.config file.

If you have a control, not just component, in your assembly, and you drop that component on a Web form in your project at design time, Visual Studio generates a reference to the assembly in the .aspx file. For example, adding a C1WebGrid control to a Web form generates the following reference:

```
<%@ Register Assembly="C1.Web.C1WebGrid.2" Namespace="C1.Web.C1WebGrid"
TagPrefix="C1WebGrid" %>
```

With such reference, ASP.NET will load your assembly from the GAC at run time without problems. Therefore, there is no need for additional references in Web.config if you use a control, not just a component. But if your assembly only has components, not controls, as is the case with C1WebDataObjects, or, if you do not have a control on the Web form at design time, then you need to place those references in Web.config (only if your assembly is NOT in the bin on the development machine). Otherwise, it will not work at run time. It will fail to load the assembly and say that your Component class belongs to an assembly whose reference is not included in the project.

Note: If your assembly is in the GAC and in the bin, it can be loaded twice in Visual Studio at design time. This can cause problems for objects that have typeconverters. Because of this, ComponentOne requires **WebDataObjects for ASP.NET** assemblies to be present in the GAC and NOT present in the bin at design time.

Add the following code to Web.config of your WebDataObjects for ASP.NET project, replacing the version number with the correct version number of your assembly:

```
<configuration>
  <system.web>

    <compilation
         defaultLanguage="c#"
         debug="true">
            <assemblies>
                <add assembly="C1.Web.Data.2, Version=2.0.200X3.0,
Culture=neutral, PublicKeyToken=545227e12fe8ee01"
    Namespace="C1.Web.Data" TagPrefix="cc1" %>
                <add assembly="C1.Web.Data.Express.2, Version=2.0.20053.0,
Culture=neutral, PublicKeyToken=e887d47c090832ef"
    Namespace="C1.Web.Data.Express" TagPrefix="cc2" %>

            </assemblies>
    </compilation>
 </system.web>

</configuration>
```

Remove any references to these assemblies from the application once they have been added to the Web.config file.

Note: You will only have to change this code if the assembly version of the .dll changes.