**Page and Application Context in ASP.NET Web Applications**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/swe97x0b(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/swe97x0b(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/swe97x0b(d=printer,v=vs.80).aspx)
* [Visual Studio .NET 2003](http://msdn.microsoft.com/en-us/library/swe97x0b(d=printer,v=vs.71).aspx)

When a Web application runs, ASP.NET maintains information about the current application, each user session, the current HTTP request, the requested page, and so on. ASP.NET contains a series of classes to encapsulate this context information.

ASP.NET makes instances of these classes available as intrinsic objects that you can access from your code. The following table lists these intrinsic objects and the classes they are instances of.

|  |  |  |
| --- | --- | --- |
| **Object Name** | **Description** | **ASP.NET Class** |
| Response | Provides access to the output stream for the current page. You can use this class to inject text into the page, to write cookies, and more. For details, see [Page.Response](http://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) property. | [HttpResponse](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx) |
| Request | Provides access to the current page request, including the request headers, cookies, client certificate, query string, and so on. You can use this class to read what the browser has sent. For details, see [Page.Request](http://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) property. | [HttpRequest](http://msdn.microsoft.com/en-us/library/system.web.httprequest.aspx) |
| Context | Provides access to the entire current context (including the request object). You can use this class to share information between pages. For details, see [Page.Context](http://msdn.microsoft.com/en-us/library/system.web.ui.page.context.aspx) property. | [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx) |
| Server | Exposes utility methods that you can use to transfer control between pages, get information about the most recent error, encode and decode HTML text, and more. For details, see [Page.Server](http://msdn.microsoft.com/en-us/library/system.web.ui.page.server.aspx) property. | [HttpServerUtility](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.aspx) |
| Application | Provides access to application-wide methods and events for all sessions. Also provides access to an application-wide cache you can use to store information. For details, see [ASP.NET Application State Overview](http://msdn.microsoft.com/en-us/library/ms178594.aspx). | [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) |
| Session | Provides information to the current user session. Also provides access to a session-wide cache you can use to store information, along with the means to control how the session is managed. For details, see [ASP.NET Session State Overview](http://msdn.microsoft.com/en-us/library/ms178581.aspx). | [HttpSessionState](http://msdn.microsoft.com/en-us/library/system.web.sessionstate.httpsessionstate.aspx) |
| Trace | Provides a way to display both system and custom trace diagnostic messages in the HTTP page output. For details, see [ASP.NET Tracing Overview](http://msdn.microsoft.com/en-us/library/bb386420.aspx). | [TraceContext](http://msdn.microsoft.com/en-us/library/system.web.tracecontext.aspx) |

The following topics show examples of how the intrinsic objects can be used.

|  |  |
| --- | --- |
| **Object** | **Example topics** |
| Application | * [How to: Read Values from Application State](http://msdn.microsoft.com/en-us/library/y8hhek39.aspx) * [How to: Save Values in Application State](http://msdn.microsoft.com/en-us/library/94xkskdf.aspx) |
| Request | * [How to: Read a Cookie](http://msdn.microsoft.com/en-us/library/bd70eh18.aspx) * [How to: Write a Cookie](http://msdn.microsoft.com/en-us/library/78c837bd.aspx) |
| Server | * [How to: Handle Application-Level Errors](http://msdn.microsoft.com/en-us/library/24395wz3.aspx) * [How to: Handle Page-Level Errors](http://msdn.microsoft.com/en-us/library/ed577840.aspx) |
| Session | * [How to: Save Values in Session State](http://msdn.microsoft.com/en-us/library/6ad7zeeb.aspx) * [How to: Read Values from Session State](http://msdn.microsoft.com/en-us/library/03sekbw5.aspx) |

# How to: Read Values from Application State

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/y8hhek39(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/y8hhek39(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/y8hhek39(d=printer,v=vs.80).aspx)

Application state is a data repository that is available to all classes within an ASP.NET application. Application state is stored in memory on the server and is faster than storing and retrieving data in a database. Unlike session state, which is specific to a single user session, application state applies to all users and sessions. Therefore, application state is a useful place to store small amounts of often-used data that does not change from one user to another.

Application state is stored in the [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class, a new instance of which is created the first time a user accesses a URL resource within an application. For more information, see [ASP.NET Application State Overview](http://msdn.microsoft.com/en-us/library/ms178594.aspx).

Application state stores data typed as [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx). Therefore, even though you do not have to serialize the data when storing it in application state, you must cast the data to the appropriate type when retrieving it. Although you can cast a null (Nothing in Visual Basic) object, if you attempt to use a non-existent application-state entry in some other way (for example, to examine its type), a [NullReferenceException](http://msdn.microsoft.com/en-us/library/system.nullreferenceexception.aspx) exception is thrown.

[Procedure](javascript:void(0))

### To read a value from application state

* Determine whether the application variable exists, and then convert the variable to the appropriate type when you access it.

The following code example retrieves the application state AppStartTime value and converts it to a variable named appStateTime of type [DateTime](http://msdn.microsoft.com/en-us/library/system.datetime.aspx).

VB

If (Not Application("AppStartTime") Is Nothing) Then

Dim myAppStartTime As DateTime = \_

CDate(Application("AppStartTime"))

End If

C#

if (Application["AppStartTime"] != null)

{

DateTime myAppStartTime = (DateTime)Application["AppStartTime"];

}

# How to: Save Values in Application State

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/94xkskdf(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/94xkskdf(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/94xkskdf(d=printer,v=vs.80).aspx)

Application state is a data repository that is available to all classes in an ASP.NET application. Application state is stored in memory on the server and is faster than storing and retrieving information in a database. Unlike session state, which is specific to a single user session, application state applies to all users and sessions. Therefore, application state is a useful place to store small amounts of often-used data that does not change from one user to another.

Application state is stored in the [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class, a new instance of which is created the first time a user accesses any URL resource in an application. The [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class is exposed through the [Application](http://msdn.microsoft.com/en-us/library/system.web.ui.page.application.aspx) property.

Application state stores data as [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx) data types. Therefore, you must convert the data back to the appropriate type when retrieving it.

Application state is stored in memory on the server, so a large amount of data in application state can fill up server memory quickly. If the application is restarted, application state data is lost. Application state is not shared between multiple servers within a Web farm or between worker processes in a Web garden. Finally, application state is free-threaded, so any data that is stored in application state must have built-in synchronization support. For more information about these considerations, see [ASP.NET Application State Overview](http://msdn.microsoft.com/en-us/library/ms178594.aspx) and [ASP.NET State Management Recommendations](http://msdn.microsoft.com/en-us/library/z1hkazw7.aspx).

### To write a value to application state

* In your application, set the value of the variable in the [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class.

The following code example shows how you can set the application variable Message to a string.

VB

Application("Message") = "Welcome to the Contoso site."

C#

Application["Message"] = "Welcome to the Contoso site.";

### To write a value to application state when the application starts

* In Application\_Start handler of your application's Global.asax file, set the value of the application state variable. Just as in a regular .aspx page, the [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class is exposed through the [Application](http://msdn.microsoft.com/en-us/library/system.web.ui.page.application.aspx) object.

The following code example shows how you can set the application variable Message to a string and initialize the variable PageRequestCount to 0.

VB

Application("Message") = "Welcome to the Contoso site."

Application("PageRequestCount") = 0

C#

Application["Message"] = "Welcome to the Contoso site.";

Application["PageRequestCount"] = 0;

[Writing a Value to Application State with Locking](javascript:void(0))

Application state variables can be accessed by multiple threads at the same time. Therefore, to prevent invalid data, you must lock application state for writing by only one thread before setting values.

|  |
| --- |
| **Note** |
| You should always modify application state data within a lock statement unless you have set some other type of lock. For more information, see [Synchronizing Data for Multithreading](http://msdn.microsoft.com/en-us/library/z8chs7ft.aspx). |

### To write a value to application state with locking

* In the code where you set the application variable, call the [HttpApplicationState.Lock](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.lock.aspx) method, set the application state value, and then call the [HttpApplicationState.UnLock](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.unlock.aspx) method to unlock the application state, freeing it for other write requests.

The following code example shows how you can lock and unlock application state. The code increases the PageRequestCount variable by 1 and then unlocks application state.

VB

Application.Lock()

Application("PageRequestCount") = \_

CInt(Application("PageRequestCount")) + 1

Application.UnLock()

C#

Application.Lock();

Application["PageRequestCount"] =

((int)Application["PageRequestCount"])+1;

Application.UnLock();

# How to: Read a Cookie

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/bd70eh18(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/bd70eh18(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/bd70eh18(d=printer,v=vs.80).aspx)

Cookies provide a means in Web applications to store user-specific information, such as history or user preferences. A cookie is a small bit of text that accompanies requests and responses as they go between the Web server and client. The cookie contains information that the Web application can read whenever the user visits the site.

The browser is responsible for managing cookies on a user system. Cookies are sent to the server with a page request and are accessible as part of the [HttpRequest](http://msdn.microsoft.com/en-us/library/system.web.httprequest.aspx) object, which exposes a [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httprequest.cookies.aspx) collection. You can read only cookies that have been created by pages in the current domain or path.

[Procedure](javascript:void(0))

### To read a cookie

* Read a string from the [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httprequest.cookies.aspx) collection using the cookie's name as the key.

The following example reads a cookie named UserSettings and then reads the value of the subkey named Font.

VB

If (Request.Cookies("UserSettings") IsNot Nothing) Then

Dim userSettings As String

If (Request.Cookies("UserSettings")("Font") IsNot Nothing) Then

userSettings = Request.Cookies("UserSettings")("Font")

End If

End If

C#

if (Request.Cookies["UserSettings"] != null)

{

string userSettings;

if (Request.Cookies["UserSettings"]["Font"] != null)

{ userSettings = Request.Cookies["UserSettings"]["Font"]; }

}

# How to: Write a Cookie

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/78c837bd(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/78c837bd(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/78c837bd(d=printer,v=vs.80).aspx)

Cookies provide a means in Web applications to store user-specific information, such as history or user preferences. A cookie is a small bit of text that accompanies requests and responses as they go between the Web server and client. The cookie contains information that the Web application can read whenever the user visits the site.

The browser manages the cookies on client computers. Cookies are sent to the client using the [HttpResponse](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx) object, which exposes a property called [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cookies.aspx). Any cookies that you want your Web application to send to the browser must be added to this collection. When you write a new cookie, you must specify the [Name](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.name.aspx) and [Value](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.value.aspx). Each cookie must have a unique name so that your Web application can identify it when the browser sends it with future requests.

There are two ways to write a cookie to a user's computer. You can either directly set cookie properties on the [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cookies.aspx) collection or you can create an instance of the [HttpCookie](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.aspx) object and add it to the [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cookies.aspx) collection. You must create cookies before the ASP.NET page is rendered to the client. For example, you can write a cookie in a Page\_Load event handler but not in a Page\_Unload event handler. For more information on the page life cycle see [ASP.NET Page Life Cycle Overview](http://msdn.microsoft.com/en-us/library/ms178472.aspx).

For more information, see [ASP.NET Cookies Overview](http://msdn.microsoft.com/en-us/library/ms178194.aspx).

### To write a cookie by setting cookie properties on the Cookies collection

* In the ASP.NET page you want to write a cookie, assign properties to a cookie in the [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cookies.aspx) collection.

The following code example shows a cookie named UserSettings with the values of the subkeys Font and Color set. It also sets the expiration time to be tomorrow.

VB

Response.Cookies("UserSettings")("Font") = "Arial"

Response.Cookies("UserSettings")("Color") = "Blue"

Response.Cookies("UserSettings").Expires = DateTime.Now.AddDays(1)

C#

Response.Cookies["UserSettings"]["Font"] = "Arial";

Response.Cookies["UserSettings"]["Color"] = "Blue";

Response.Cookies["UserSettings"].Expires = DateTime.Now.AddDays(1d);

### To write a cookie by creating an instance of the HttpCookie object

1. Create an object of type [HttpCookie](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.aspx) and assign it a name.
2. Assign values to cookie's subkeys and set any cookie properties.
3. Add the cookie to the [Cookies](http://msdn.microsoft.com/en-us/library/system.web.httpresponse.cookies.aspx) collection.

The following code example shows an instance of the [HttpCookie](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.aspx) object named myCookie, which represents a cookie named UserSettings.

VB

Dim myCookie As HttpCookie = New HttpCookie("UserSettings")

myCookie("Font") = "Arial"

myCookie("Color") = "Blue"

myCookie.Expires = Now.AddDays(1)

Response.Cookies.Add(myCookie)

C#

HttpCookie myCookie = new HttpCookie("UserSettings");

myCookie["Font"] = "Arial";

myCookie["Color"] = "Blue";

myCookie.Expires = DateTime.Now.AddDays(1d);

Response.Cookies.Add(myCookie);

[Robust Programming](javascript:void(0))

By default, cookies are shared by all pages that are in the same domain, but you can limit cookies to specific subfolders in a Web site by setting their [Path](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.path.aspx) property. To allow a cookie to be retrieved by all pages in all folders of your application, set it from a page that is in the root folder of your application and do not set the [Path](http://msdn.microsoft.com/en-us/library/system.web.httpcookie.path.aspx) property.

If you do not specify an expiration limit for the cookie, the cookie is not persisted to the client computer and it expires when the user session expires.

Cookies can store values only of type [String](http://msdn.microsoft.com/en-us/library/system.string.aspx). You must convert any non-string values to strings before you can store them in a cookie. For many data types, calling the [ToString](http://msdn.microsoft.com/en-us/library/system.object.tostring.aspx) method is sufficient. For more information, see the ToString method for the data type you wish to persist.

[Security](javascript:void(0))

Do not store sensitive information, such as a user name or a password, in a cookie. For more cookie security information see [ASP.NET Cookies Overview](http://msdn.microsoft.com/en-us/library/ms178194.aspx).

**How to: Handle Application-Level Errors**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/24395wz3(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/24395wz3(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/24395wz3(d=printer,v=vs.80).aspx)

This code example shows how to create an error handler in the Global.asax file that will catch all unhandled ASP.NET errors while processing a request — in other words, all the errors that are not caught with a Try/Catch block or in a page-level error handler. In the example, the handler transfers control to a generic error page named GenericErrorPage.aspx, which interprets the error and displays an appropriate message.

[Example](javascript:void(0))

The following example is from a complete code sample in [Complete Example for Error Handlers](http://msdn.microsoft.com/en-us/library/bb397417.aspx).

|  |
| --- |
| **Description: Security noteSecurity Note** |
| Never set customErrors to Off in your Web.config file if you do not have an Application\_Error handler in your Global.asax file. Potentially compromising information about your Web site can be exposed to anyone who can cause an error to occur on your site. |

C#

[VB](http://msdn.microsoft.com/en-us/library/24395wz3(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

void Application\_Error(object sender, EventArgs e)

{

// Code that runs when an unhandled error occurs

// Get the exception object.

Exception exc = Server.GetLastError();

// Handle HTTP errors

if (exc.GetType() == typeof(HttpException))

{

// The Complete Error Handling Example generates

// some errors using URLs with "NoCatch" in them;

// ignore these here to simulate what would happen

// if a global.asax handler were not implemented.

if (exc.Message.Contains("NoCatch") || exc.Message.Contains("maxUrlLength"))

return;

//Redirect HTTP errors to HttpError page

Server.Transfer("HttpErrorPage.aspx");

}

// For other kinds of errors give the user some information

// but stay on the default page

Response.Write("<h2>Global Page Error</h2>\n");

Response.Write(

"<p>" + exc.Message + "</p>\n");

Response.Write("Return to the <a href='Default.aspx'>" +

"Default Page</a>\n");

// Log the exception and notify system operators

ExceptionUtility.LogException(exc, "DefaultPage");

ExceptionUtility.NotifySystemOps(exc);

// Clear the error from the server

Server.ClearError();

}

[Robust Programming](javascript:void(0))

An error handler that is defined in the Global.asax file will only catch errors that occur during processing of requests by the ASP.NET runtime. For example, it will catch the error if a user requests an .aspx file that does not occur in your application. However, it does not catch the error if a user requests a nonexistent .htm file. For non-ASP.NET errors, you can create a custom handler in Internet Information Services (IIS). The custom handler will also not be called for server-level errors.

You cannot directly output error information for requests from the Global.asax file; you must transfer control to another page, typically a Web Forms page. When transferring control to another page, use [Transfer](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.transfer.aspx) method. This preserves the current context so that you can get error information from the [GetLastError](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.getlasterror.aspx) method.

After handling an error, you must clear it by calling the [ClearError](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.clearerror.aspx) method of the Server object ([HttpServerUtility](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.aspx) class).

[Security](javascript:void(0))

Be sure that you do not display error information that might help malicious users compromise your application. For details, see [How to: Display Safe Error Messages](http://msdn.microsoft.com/en-us/library/994a1482.aspx).

**How to: Handle Page-Level Errors**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/ed577840(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/ed577840(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/ed577840(d=printer,v=vs.80).aspx)

If possible, you should handle errors in Try/Catch blocks within your code, because a problem is more easily corrected where it occurs. If the user can help correct a problem, the page needs to return to the same place so the user has a context for understanding what to do.

A page-level handler returns you to the page, but there is no longer anything on the page because instances of controls are not created. To provide the user any information, you must specifically write it to the page.

You would probably use a page-level error handler to log unhandled errors or to take the user to a page that can display helpful information.

This code example shows a handler for the [Error](http://msdn.microsoft.com/en-us/library/system.web.ui.templatecontrol.error.aspx) event in an ASP.NET Web page. This handler catches all exceptions that are not already handled within Try/Catch blocks in the page.

After you handle an error, you must clear it by calling the [ClearError](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.clearerror.aspx) method of the Server object ([HttpServerUtility](http://msdn.microsoft.com/en-us/library/system.web.httpserverutility.aspx) class).

[Example](javascript:void(0))

This handler filters for specific kinds of exceptions. For an [ArgumentOutOfRangeException](http://msdn.microsoft.com/en-us/library/system.argumentoutofrangeexception.aspx) exception, the handler writes some text on the page, provides a link back to the page, logs the error, and notifies system administrators. For an [InvalidOperationException](http://msdn.microsoft.com/en-us/library/system.invalidoperationexception.aspx) exception, the handler simply transfers the exception to the Generic Error Page. For any other kind of exception, the handler does nothing, which allows your site to automatically redirect to the generic page specified in the Web.config file. Your own code would filter for exceptions that are important to your application.

The following example is part of a complete code sample in [Complete Example for Error Handlers](http://msdn.microsoft.com/en-us/library/bb397417.aspx)

C#

[VB](http://msdn.microsoft.com/en-us/library/ed577840(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

private void Page\_Error(object sender, EventArgs e)

{

// Get last error from the server

Exception exc = Server.GetLastError();

// Handle exceptions generated by Button 1

if (exc is InvalidOperationException)

{

// Pass the error on to the Generic Error page

Server.Transfer("GenericErrorPage.aspx", true);

}

// Handle exceptions generated by Button 2

else if (exc is ArgumentOutOfRangeException)

{

// Give the user some information, but

// stay on the default page

Response.Write("<h2>Default Page Error</h2>\n");

Response.Write("<p>Provide as much information here as is " +

"appropriate to show to the client.</p>\n");

Response.Write("Return to the <a href='Default.aspx'>" +

"Default Page</a>\n");

// Log the exception and notify system operators

ExceptionUtility.LogException(exc, "DefaultPage");

ExceptionUtility.NotifySystemOps(exc);

// Clear the error from the server

Server.ClearError();

}

else

{

// Pass the error on to the default global handler

}

}

[Security](javascript:void(0))

Use the [customErrors](http://msdn.microsoft.com/en-us/library/h0hfz6fc.aspx) element to restrict display of detailed error messages to local users only.

Be sure that you do not display error information that might help malicious users compromise your application. For details, see [How to: Display Safe Error Messages](http://msdn.microsoft.com/en-us/library/994a1482.aspx).

**How to: Save Values in Session State**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/6ad7zeeb(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/6ad7zeeb(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/6ad7zeeb(d=printer,v=vs.80).aspx)

This example uses the [HttpSessionState](http://msdn.microsoft.com/en-us/library/system.web.sessionstate.httpsessionstate.aspx) object to persist values within an individual session.

[Example](javascript:void(0))

C#

[VB](http://msdn.microsoft.com/en-us/library/6ad7zeeb(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

string firstName = "Jeff";

string lastName = "Smith";

string city = "Seattle";

Session["FirstName"] = firstName;

Session["LastName"] = lastName;

Session["City"] = city;

[Compiling the Code](javascript:void(0))

This example requires:

* A Web Forms page or class that has access to the current request context using the [Current](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.current.aspx) property in an ASP.NET application that has session state enabled.

[Robust Programming](javascript:void(0))

Session state can expire (by default, after 20 minutes of inactivity), and the information that you store there can be lost. You can control session-state lifetime using the timeout attribute of the [sessionState](http://msdn.microsoft.com/en-us/library/h6bb9cz9.aspx) configuration section.

Depending on your application requirements, you may want to consider an alternative to session state for storing information for each user. ASP.NET provides several other options for persisting data within an application. For a comparison of each, see [ASP.NET State Management Recommendations](http://msdn.microsoft.com/en-us/library/z1hkazw7.aspx).

**How to: Read Values from Session State**

**.NET Framework 4**

[Other Versions](javascript:;)

Description: http://i.msdn.microsoft.com/Areas/Epx/Content/Images/ImageSprite.png

* [Visual Studio 2008](http://msdn.microsoft.com/en-us/library/03sekbw5(d=printer,v=vs.90).aspx)
* [.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/03sekbw5(d=printer,v=vs.85).aspx)
* [Visual Studio 2005](http://msdn.microsoft.com/en-us/library/03sekbw5(d=printer,v=vs.80).aspx)

This example accesses the [Item](http://msdn.microsoft.com/en-us/library/system.web.sessionstate.httpsessionstate.item.aspx) property to retrieve the values in session state.

[Example](javascript:void(0))

C#

[VB](http://msdn.microsoft.com/en-us/library/03sekbw5(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-1)

string firstName = (string)(Session["First"]);

string lastName = (string)(Session["Last"]);

string city = (string)(Session["City"]);

[Compiling the Code](javascript:void(0))

This example requires:

* A Web Forms page or class that has access to the current request context using the [Current](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.current.aspx) property in an ASP.NET application that has session state enabled.

[Robust Programming](javascript:void(0))

No exception is thrown if you attempt to get a value out of session state that does not exist. To be sure that the value you want is in session state, check first for the existence of the object with a test such as the following:

C#

[VB](http://msdn.microsoft.com/en-us/library/03sekbw5(d=printer).aspx?cs-save-lang=1&cs-lang=vb#code-snippet-2)

if (Session["City"] == null)

// No such value in session state; take appropriate action.

If you attempt to use a nonexistent session state entry in some other way (for example, to examine its type), a [NullReferenceException](http://msdn.microsoft.com/en-us/library/system.nullreferenceexception.aspx) exception is thrown.

Session values are of type [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx). In Visual Basic, if you set Option Strict On, you must cast from type [Object](http://msdn.microsoft.com/en-us/library/system.object.aspx) to the appropriate type when getting values out of session state, as shown in the example. In C#, you should always cast to the appropriate type when reading session values.