**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

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_303293c9-097d-4197-a647-831a96b69abe');" \o "Copy to clipboard.)

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

Dim myAppStartTime As DateTime = \_

CDate(Application("AppStartTime"))

End If

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_0f556d64-77b3-48af-9ed0-23ebf5290bb4');)

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.

|  |
| --- |
| **Description: NoteNote** |
| 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();