**ASP.NET Application State Overview**

Application state is a data repository 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. For information on saving data on a per-user basis see [ASP.NET Session State Overview](http://msdn.microsoft.com/en-us/library/ms178581.aspx) and [ASP.NET Profile Properties Overview](http://msdn.microsoft.com/en-us/library/2y3fs9xs.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Using Application State

Application state is stored in an instance of the [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class. This class exposes a key-value dictionary of objects.

The [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) instance 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 most often accessed through the [Application](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.application.aspx) property of the [HttpContext](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.aspx) class.

You can use application state in two ways. You can add, access, or remove values from the [Contents](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.contents.aspx) collection directly through code. The [HttpApplicationState](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.aspx) class can be accessed at any time during the life of an application. However, it is often useful to load application state data when the application starts. To do so, you can put code to load application state into the **Application\_Start** method in the Global.asax file. For more information see [ASP.NET Application Life Cycle Overview for IIS 5.0 and 6.0](http://msdn.microsoft.com/en-us/library/ms178473.aspx).

Alternatively, you can add objects to the [StaticObjects](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.staticobjects.aspx) collection via an <object runat="server"> declaration in your Web application's Global.asax file. Application state defined in this way can then be accessed from code anywhere in your application. The following example shows an object declaration for an application state value:

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl30other');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl30other');)

<object runat="server" scope="application" ID="MyInfo"

PROGID="MSWC.MYINFO">

</object>

You can add objects to the [StaticObjects](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.staticobjects.aspx) collection only in the Global.asax file. The collection throws a [NotSupportedException](http://msdn.microsoft.com/en-us/library/system.notsupportedexception.aspx) if you attempt to add objects directly through code.

You can access members of objects stored in application state without having to reference the [Application](http://msdn.microsoft.com/en-us/library/system.web.httpcontext.application.aspx) collection. The following code example shows how to reference a member of an object defined in the [StaticObjects](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.staticobjects.aspx) collection of application state:

Visual Basic

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl35VisualBasic');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl35VisualBasic');)

Protected Sub Page\_Load(ByVal sender As Object, ByVal e As EventArgs)

Label1.Text = MyInfo.Title

End Sub

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl36CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl36CSharp');)

protected void Page\_Load(Object sender, EventArgs e)

Label1.Text = MyInfo.Title;

End Sub

http://i.msdn.microsoft.com/Global/Images/clear.gif Application State Considerations

When using application state, you must be aware of the following important considerations:

* **Resources**   Because it is stored in memory, application state is very fast compared to saving data to disk or a database. However, storing large blocks of data in application state can fill up server memory, causing the server to page memory to disk. As an alternative to using application state, you can use the ASP.NET cache mechanism for storing large amounts of application data. The ASP.NET cache also stores data in memory and is therefore very fast; however, ASP.NET actively manages the cache and will remove items when memory becomes scarce. For more information see [ASP.NET Caching Overview](http://msdn.microsoft.com/en-us/library/ms178597.aspx).
* **Volatility**   Because application state is stored in server memory, it is lost whenever the application is stopped or restarted. For example, if the Web.config file is changed, the application is restarted and all application state is lost unless application state values have been written to a non-volatile storage medium such as a database.
* **Scalability**   Application state is not shared among multiple servers serving the same application, as in a Web farm, or among multiple worker processes serving the same application on the same server, as in a Web garden. Your application therefore cannot rely on application state containing the same data for application state across different servers or processes. If your application will run in multi-processor or multi-server environments, consider using a more scalable option, such as a database, for data that must preserve fidelity across the application.
* **Concurrency**   Application state is free-threaded, which means that application state data can be accessed simultaneously by many threads. Therefore, it is important to ensure that when you update application state data, you do so in a thread-safe manner by including built-in synchronization support. You can use the [Lock](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.lock.aspx) and [UnLock](http://msdn.microsoft.com/en-us/library/system.web.httpapplicationstate.unlock.aspx) methods to ensure data integrity by locking the data for writing by only one source at a time. You can also reduce the likelihood of concurrency problems by initializing application state values in the **Application\_Start** method in the Global.asax file. For more information see [ASP.NET Application Life Cycle Overview for IIS 5.0 and 6.0](http://msdn.microsoft.com/en-us/library/ms178473.aspx).

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

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.

Visual Basic

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08VisualBasic');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl08VisualBasic');)

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

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl09CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl09CSharp');)

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.

Visual Basic

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl12VisualBasic');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl12VisualBasic');)

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

Application("PageRequestCount") = 0

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl13CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl13CSharp');)

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

Application["PageRequestCount"] = 0;

http://i.msdn.microsoft.com/Global/Images/clear.gif Writing a Value to Application State with Locking

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.

|  |
| --- |
| **94xkskdf.alert_note(en-us,VS.90).gifNote:** |
| 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.

Visual Basic

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl30VisualBasic');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl30VisualBasic');)

Application.Lock()

Application("PageRequestCount") = \_

CInt(Application("PageRequestCount")) + 1

Application.UnLock()

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl31CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl31CSharp');)

Application.Lock();

Application["PageRequestCount"] =

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

Application.UnLock();

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

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.

http://i.msdn.microsoft.com/Global/Images/clear.gif Procedure

**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).

Visual Basic

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl11VisualBasic');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl11VisualBasic');)

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

Dim myAppStartTime As DateTime = \_

CDate(Application("AppStartTime"))

End If

C#

[[http://i.msdn.microsoft.com/Global/Images/clear.gif](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl12CSharp');)Copy Code](javascript:CopyCode('ctl00_rs1_mainContentContainer_ctl12CSharp');)

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

{

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

}