**In-Proc Session State Management**

**What is ASP.NET Session?**

A session is defined as the period of time a unique user interacts with a particular web application.

HTTP is a stateless protocol, in the sense that a Web server is concerned only with the current HTTP request for any given Web page. The server retains no knowledge of previous requests.

The stateless nature of HTTP requests presents unique challenges when writing Web applications.

ASP and ASP.NET applications maintain a user’s session using a feature called session state.

In this series we are going to talk in detail about how to troubleshoot ASP.NET session state related issues.

**ASP.NET SESSION MANAGEMENT METHODOLOGIES**

There are three modes available for ASP.NET session state, within the worker process, SQL server or in a dedicated process which can run either on the same machine or on a different machine. We can select where to store the session data and this can be configured using the mode attribute of the <sessionState> element in the web.config file. Below is an example of the <sessionState> element:

<sessionState mode="InProc"

stateConnectionString="tcpip=127.0.0.1:42424"

sqlConnectionString="data source=127.0.0.1;Trusted\_Connection=yes"

cookieless="false" timeout="20" />

The following table describes the various options and a brief description of each mode

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Option** | **Description** |
| mode |  | Specifies where to store the session state. |
|  | Off | Disables session state management. |
|  | InProc | Session state is stored locally in memory of ASP.NET worker process. |
|  | StateServer | Session state is stored outside of the ASP.NET worker process and is managed by a Windows service. The location of this service is specified by stateConnectionString attribute. |
|  | SQLServer | Session state is stored outside of the ASP.NET worker process in a SQL Server database. The location of this database is specified using the sqlConnectionString attribute. |

The table lists the optional attributes that can be used along with the mode attribute within the <sessionState> element

|  |  |  |
| --- | --- | --- |
| Attribute | Option | Description |
| cookieless |  | Specifies whether sessions without cookies should be used to identify client sessions. |
|  | true | Indicates that sessions without cookies should be used. |
|  | false | Indicates that sessions without cookies should not be used. This is the default setting. |
| timeout |  | Specifies the number of **minutes** a session can be idle before it is abandoned. The default is 20. |
| stateConnectionString |  | Specifies the server name and port where session state is stored remotely. For example, "tcpip=127.0.0.1:42424". This attribute is required when mode is StateServer. |
| sqlConnectionString |  | Specifies the connection string for a SQL Server. For example, "data source=localhost;Integrated Security=SSPI;Initial Catalog=ASPState". This attribute is required when mode is SQLServer. |
| stateNetworkTimeout |  | When using StateServer mode to store session state, specifies the number of seconds the TCP/IP network connection between the Web server and the state server can be idle before the session is abandoned. The default is 10. |

**InProc Session State**

InProc state is considerably faster than either the state server or SQL Server session storage techniques. SQL Server and state server attribute their slowness to serialization/deserialization that happens while reading in/storing out the Session data from the SQL/State server.

. However storing the session state InProc has its own share of limitations. Some of these are

- With each app domain restart Session state is lost

- Process restart will result in loss of Session state data

- If considerable amount of data are stored in Session state, memory consumption for the process may increase to the point of experiencing issues due to high memory. Therefore, the amount of data being stored in Session state and the time frame during which these data need to be stored should be limited as much as possible.

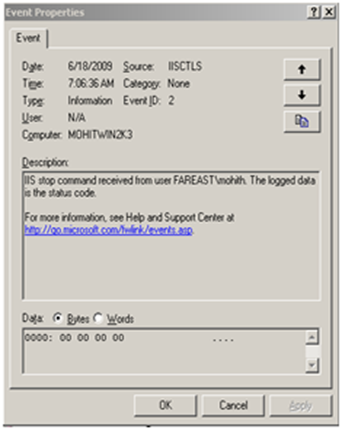
- In case one wants to implement InProc Session state in a web farm scenario ensure we have sticky sessions enabled for the web farm as it involves storing of the session data among processes hosted on multiple servers.

**Troubleshooting InProc Session Loss**

If you have established that you are running into a session loss issue you should determine whether all users face the issue at the same time or the issue happens for a specific user.

In a scenario where all users face the issue at the same time, an application domain or a worker process recycle might be the cause of the issue. Process recycle can happen due to multiple reasons including an IISRESET or in case recycling options are enabled in the Application pool properties. If the process was recycled due to an IISRESET you will see the following in the System event logs

“IIS stop command received from user <domain\username>. The logged data is the status code”

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/image_4.png)

However if the process recycle is due to the application pool recycling options, then we may see different events in the system log. One such example is:

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image004_2.jpg)

You can configure the Application Pool recycling options by going to the properties of the application pool

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image006_2.jpg)

You can also configure IIS to log application pool recycle events that might cause session loss. This can be done by using the LogEventOnRecycle metabase key. By default the key is configured to capture the following events

1. AppPoolRecycleTime

2. AppPoolRecycleMemory

3. AppPoolRecyclePrivateMemory

But we can configure it to log events for other events as well. Below is the list of the recycling events that can be logged using this key

|  |  |
| --- | --- |
| Value (Decimal) | Description |
| 1 | AppPoolRecycleTime |
| 2 | AppPoolRecycleRequests |
| 4 | AppPoolRecycleSchedule |
| 8 | AppPoolRecycleMemory |
| 16 | AppPoolRecycleIsapiUnhealthy |
| 32 | AppPoolRecycleOnDemand |
| 64 | AppPoolRecycleConfigChange |
| 128 | AppPoolRecyclePrivateMemory |

In order to enable logging for all recycle events we need to run the following command

C:\Inetpub\AdminScripts>cscript adsutil.vbs SET W3SVC/AppPools/LogEventOnRecycle 255

Here is a screenshot of the event that will be logged in the System event log when one of the above events occur.

For e.g. if we have the value for AppPoolRecycleSchedule set in the IIS metabase you will see something similar as shown below.

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image008_2.jpg)

Reference: <http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/6f43da84-38b0-422b-aa2b-195643d05d22.mspx?mfr=true>

In case of a process crash, you may see the following events generated in the system event logs

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image010_2.jpg)

In this case you need to configure a crash rule with either Debug Diagnostic 1.1 (32 bit server) or ADPLUS (works for both 32 bit and a 64 bit server however does not have a GUI).

To Download Debug Diagnostic 1.1 - <http://www.iis.net/downloads/default.aspx?tabid=34&g=6&i=1286>

To Download ADPLUS - ADPlus is included with the latest Microsoft Debugging Tools for Windows. To obtain the latest Microsoft Debugging Tools for Windows - <http://www.microsoft.com/whdc/devtools/debugging/default.mspx>

How to use ADPlus to troubleshoot "hangs" and "crashes" - <http://support.microsoft.com/kb/286350/>

It is possible that you will not see any entries in the event logs. In that case the problem may be due to an app domain recycle.You can figure out what is causing app domain recycles by implementing one of the methods below :-

Enable health monitoring by adding the following code in the web.config file under <System.web> element

<healthMonitoring>

<rules>

<add name="Application Events"

eventName="Application Lifetime Events"

provider="EventLogProvider"

profile="Default"

minInterval="00:01:00" />

</rules>

</healthMonitoring>

Save your web.config file and run your application. View a couple of pages and change the web.config (just add a space and delete the space anywhere in your web.config and save it). Now have a look at your Event Viewer. It should have the following logs -

Event Type: Information   
Event Source: ASP.NET 2.0.50727.0   
Event Category: Web Event   
Event ID: 1305   
Date: 4/13/2006   
Time: 6:19:30 PM   
User: N/A   
Computer: <computer name>   
Description:   
Event code: 1001 Event message: Application is starting. Event time: 4/13/2006 6:19:29 PM Event time (UTC): 4/13/2006 12:49:29 PM Event ID: a01d7c9ff12a43a9baed46f7823f11cd Event sequence: 1 Event occurrence: 1 Event detail code: 0 Application information: Application domain

Event Type: Information   
Event Source: ASP.NET 2.0.50727.0   
Event Category: Web Event   
Event ID: 1305   
Date: 4/13/2006   
Time: 6:19:30 PM   
User: N/A   
Computer: <computer name>   
Description:   
Event code: 1002 Event message: Application is shutting down. Event time: 4/13/2006 6:19:29 PM Event time (UTC): 4/13/2006 12:49:29 PM Event ID: a01d7c9ff12a43a9baed46f7823f11cd Event sequence: 1 Event occurrence: 1 Event detail code: 0 Application information: Application domain

Reference – <http://msdn.microsoft.com/en-us/library/2fwh2ss9.aspx>

You can also add the following code snippet to the Application\_End event in the global.asax of your application.

public void Application\_End()

{

HttpRuntime runtime = (HttpRuntime) typeof(System.Web.HttpRuntime).InvokeMember("\_theRuntime", BindingFlags.NonPublic |

BindingFlags.Static |BindingFlags.GetField, null, null, null);

if (runtime == null) return;

string shutDownMessage = (string) runtime.GetType().InvokeMember("\_shutDownMessage", BindingFlags.NonPublic | BindingFlags.Instance |

BindingFlags.GetField, null, runtime, null);

string shutDownStack = (string) runtime.GetType().InvokeMember("\_shutDownStack", BindingFlags.NonPublic | BindingFlags.Instance | BindingFlags.GetField, null, runtime, null);

if (!EventLog.SourceExists(".NET Runtime"))

{

EventLog.CreateEventSource(".NET Runtime", "Application");

}

EventLog log = new EventLog();

log.Source = ".NET Runtime";

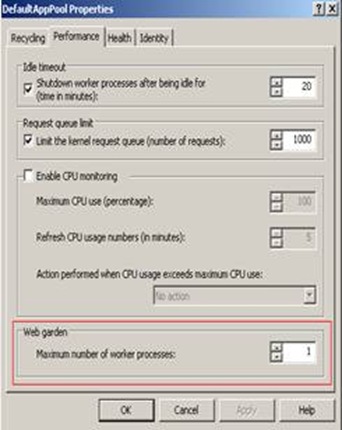
log.WriteEntry(String.Format("\r\n\r\n\_shutDownMessage={0}\r\n\r\n\_shutDownStack={1}", shutDownMessage, shutDownStack),

EventLogEntryType.Error);

}

Reference - <http://weblogs.asp.net/scottgu/archive/2005/12/14/433194.aspx>

If it’s happening for only some users there is a possibility that the Session state is set to "InProc” and Web garden is enabled in IIS. If this is indeed true then you should disable Web Gardening.

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image012_2.jpg)

If you are running Web farm and using InProc Session, then this could be a possible cause of session loss as all the Session information is stored within the Worker process. Ensure that you have NLB with sticky sessions enabled. In order to verify this we need to take a network trace and ensure that all the requests are reaching the same server they originated from.

Another common issue seen with Web farm scenarios is when the IIS Metabase is not same for all the Web servers in the farm. The main issue comes with Web site identifier mismatch. ( <http://support.microsoft.com/?id=325056> )

Another thing to check is whether the <domain name> \ <server name> has special characters such as “\_” in it? Check this <http://support.microsoft.com/?id=316112>

There can be scenarios where users get Session loss from a specific machine but not from other machine. This could be due to the browser dropping cookie. The best way to approach this is to try browsing the site with an alternate browser or use tools such as Fiddler or Network Monitor to capture the traffic and see whether we are getting the cookie.

Some of the known issues with IE are described here

**Session variables are lost if you use FRAMESET in Internet Explorer 6** à <http://support.microsoft.com/kb/323752>

Before we troubleshoot Session state, we need an application to test it with. Below is the code for a simple ASP.NET page that writes to and reads from session state.

<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="\_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>Untitled Page</title>

</head>

<body>

<script runat="server">

Sub Session\_Add(ByVal sender As Object, ByVal e As EventArgs)

Session("MySession") = text1.Value

span1.InnerHtml = "Session data updated! <P>" + "Your session contains: <font color=red>" + Session("MySession").ToString()

End Sub

Sub CheckSession(ByVal sender As Object, ByVal e As EventArgs)

If (IsNothing(Session("MySession"))) Then

span1.InnerHtml = "NOTHING, SESSION DATA LOST!"

Else

End If

span1.InnerHtml = "Your session contains: <font color=red>" + Session("MySession").ToString()

End Sub

</script>

<form id="form1" runat="server">

<input id="text1" type="text" runat="server"/>

<input type="submit" runat="server" onserverclick="Session\_Add" value="Add to Session State"/>

<input type="submit" runat="server" onserverclick="CheckSession" value="View Session State" />

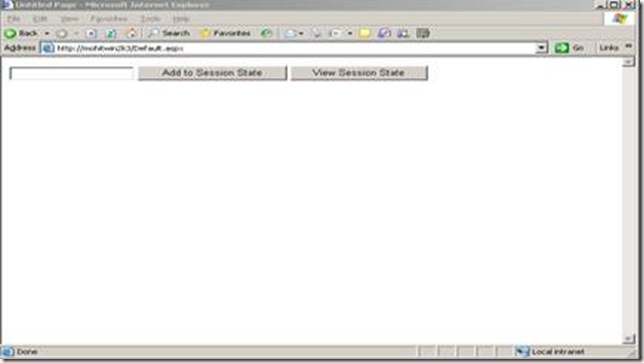
<span id="span1" runat="server" />

</form>

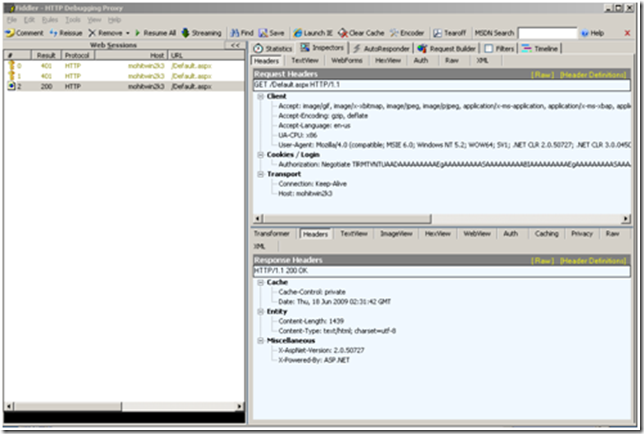
</body>

</html>

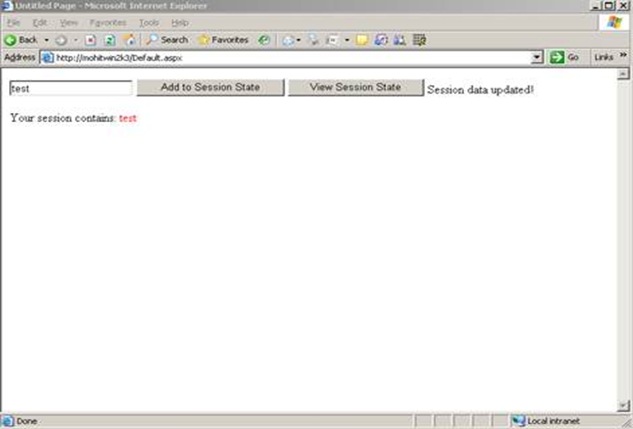
The above page has two buttons one for adding a Session state value, and the other for retrieving the value. When you browse the application you initially get the following page

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image014_2.jpg)

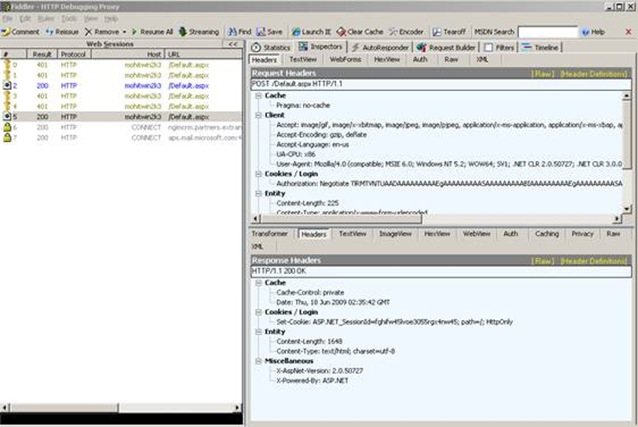
The corresponding Fiddler trace gives you the following result

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/image_6.png)

At this moment no cookie has been created and only the Authorization Header is being passed to the server in the request. Now enter a random value in the textbox and click Add to Session state. In my example, I am adding “test” as the session state value. Observe the following

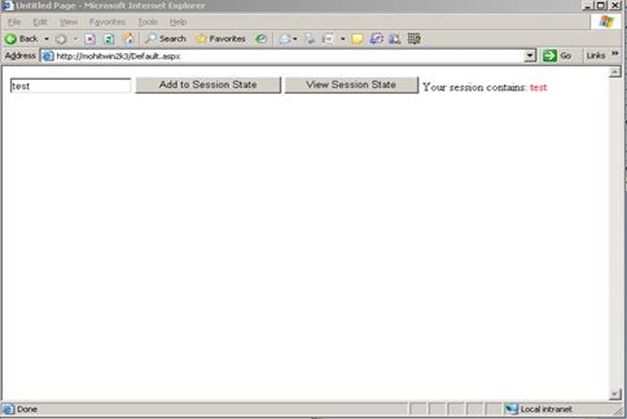
[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image018_2.jpg)

The corresponding Fiddler trace shows the following result

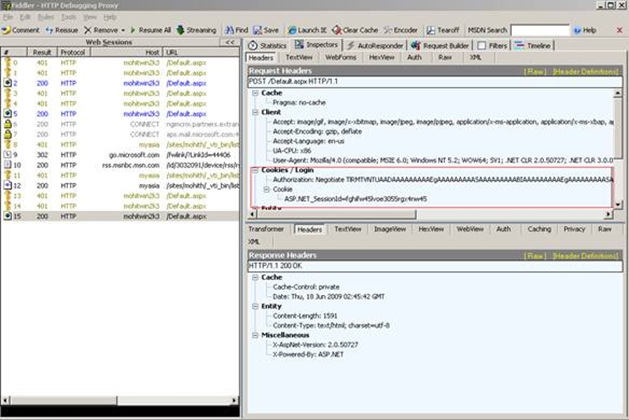
[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image020_2.jpg)

Notice in the response headers that the server is sending a Session cookie.

When the Session value is retrieved, I get the following results:

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image022_2.jpg)

The corresponding Fiddler trace gives the following result

[](http://blogs.msdn.com/blogfiles/webtopics/WindowsLiveWriter/SessionManagement_670E/clip_image024_2.jpg)

- The client has sent the session cookie. If the cookie is lost in transit or is not sent by the client, the server will be unable to locate the Session which will lead to a Session loss scenario.

Tools to use

Download Network Monitor 3.3:- <http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=983b941d-06cb-4658-b7f6-3088333d062f>

Download Fiddler: - <http://www.fiddlertool.com/fiddler/>

Note: - Fiddler doesn’t capture data over localhost. So use machine name or FQDN while trying to access the Web site.

Some of the other common issues that can contribute to a session loss are

1. Deleting ASP.NET 2.0 Application Sub-Directories Shuts Down the AppDomain

Reference - <http://blogs.msdn.com/toddca/archive/2005/12/01/499144.aspx>

2. Unhandled exceptions cause ASP.NET-based applications to unexpectedly quit in the .NET Framework 2.0

Reference - <http://support.microsoft.com/?id=911816>

3. Overwhelming Change Notifications

Reference - <http://support.microsoft.com/?id=920970>

4. Application domain recycle due to file change notifications

Reference - <http://support.microsoft.com/kb/911272/en-us>

5. In case of other queries regarding Session loss issues refer to - <http://forums.asp.net/p/7504/7504.aspx#7504>