ASP.Net State Management Techniques

By Santhosh Babu January 02, 2007

This article gives brief introduction to various state management techniques in ASP.NET.

State Management Techniques in ASP.NET

This article discusses various options for state management for web applications developed using ASP.NET. Generally, web applications are based on stateless HTTP protocol which does not retain any information about user requests. In typical client and server communication using HTTP protocol, page is created each time the page is requested.

Developer is forced to implement various state management techniques when developing applications which provide customized content and which "remembers" the user.

Here we are here with various options for ASP.NET developer to implement state management techniques in their applications. Broadly, we can classify state management techniques as client side state management or server side state management. Each technique has its own pros and cons. Let's start with exploring client side state management options.

Client side State management Options:

ASP.NET provides various client side state management options like Cookies, QueryStrings (URL), Hidden fields, View State and Control state (ASP.NET 2.0). Let's discuss each of client side state management options.

Bandwidth should be considered while implementing client side state management options because they involve in each roundtrip to server. Example: Cookies are exchanged between client and server for each page request.

Cookie:

A cookie is a small piece of text stored on user's computer. Usually, information is stored as name-value pairs. Cookies are used by websites to keep track of visitors. Every time a user visits a website, cookies are retrieved from user machine and help identify the user.

Let's see an example which makes use of cookies to customize web page.

```
if (Request.Cookies["UserId"] != null)
    lbMessage.text = "Dear" + Request.Cookies["UserId"].Value + ", Welcome to our website!";
else
    lbMessage.text = "Guest,welcome to our website!";
```

If you want to store client's information use the below code

```
Response.Cookies["UserId"].Value=username;
```

Advantages:

- Simplicity

Disadvantages:

- Cookies can be disabled on user browsers
- Cookies are transmitted for each HTTP request/response causing overhead on bandwidth
- Inappropriate for sensitive data

Hidden fields:

Hidden fields are used to store data at the page level. As its name says, these fields are not rendered by the browser. It's just like a standard control for which you can set its properties. Whenever a page is submitted to server, hidden fields values are also posted to server along with other controls on the page. Now that all the asp.net web controls have built in state management in the form of view state and new feature in asp.net 2.0 control state, hidden fields functionality seems to be redundant. We can still use it to store insignificant data. We can use hidden fields in ASP.NET pages using following syntax

```
protected System.Web.UI.HtmlControls.HtmlInputHidden Hidden1;

//to assign a value to Hidden field
Hidden1.Value="Create hidden fields";
//to retrieve a value
string str=Hidden1.Value;
```

Advantages:

- Simple to implement for a page specific data
- Can store small amount of data so they take less size.

Disadvantages:

- Inappropriate for sensitive data
- Hidden field values can be intercepted(clearly visible) when passed over a network

View State:

View State can be used to store state information for a single user. View State is a built in feature in web controls to persist data between page post backs. You can set View State on/off for each control using **EnableViewState** property. By default, **EnableViewState** property will be set to true. View state mechanism poses performance overhead. View state information of all the controls on the page will be submitted to server on each post back. To reduce performance penalty, disable View State for all the controls for which you don't need state. (Data grid usually doesn't need to maintain state). You can also disable View State for the entire page by adding **EnableViewState=false** to @page directive. View state data is encoded as binary Base64 - encoded which add approximately 30% overhead. Care must be taken to ensure view state for a page is smaller in size. View State can be used using following syntax in an ASP.NET web page.

```
// Add item to ViewState
ViewState["myviewstate"]  = myValue;

//Reading items from ViewState
Response.Write(ViewState["myviewstate"]);
```

Advantages:

- Simple for page level data
- Encrypted
- Can be set at the control level

Disadvantages:

- Overhead in encoding View State values
- Makes a page heavy

Query strings:

Query strings are usually used to send information from one page to another page. They are passed along with URL in clear text. Now that cross page posting feature is back in asp.net 2.0, Query strings

seem to be redundant. Most browsers impose a limit of 255 characters on URL length. We can only pass smaller amounts of data using query strings. Since Query strings are sent in clear text, we can also encrypt query values. Also, keep in mind that characters that are not valid in a URL must be encoded using **Server.UrlEncode**.

Let's assume that we have a Data Grid with a list of products, and a hyperlink in the grid that goes to a product detail page, it would be an ideal use of the Query String to include the product ID in the Query String of the link to the product details page (for example, productdetails.aspx?productid=4).

When product details page is being requested, the product information can be obtained by using the following codes:

```
string productid;
productid=Request.Params["productid"];
```

Advantages:

- Simple to Implement

Disadvantages:

- Human Readable
- Client browser limit on URL length
- Cross paging functionality makes it redundant
- Easily modified by end user

Control State:

Control State is new mechanism in ASP.NET 2.0 which addresses some of the shortcomings of View State. Control state can be used to store critical, private information across post backs. Control state is another type of state container reserved for controls to maintain their core behavioral functionality whereas View State only contains state to maintain control's contents (UI). Control State shares same memory data structures with View State. Control State can be propagated even though the View State for the control is disabled. For example, new control Grid View in ASP.NET 2.0 makes effective use of control state to maintain the state needed for its core behavior across post backs. Grid View is in no way affected when we disable View State for the Grid View or entire page

Server Side State management:

As name implies, state information will be maintained on the server. Application, Session, Cache and Database are different mechanisms for storing state on the server.

Care must be taken to conserve server resources. For a high traffic web site with large number of concurrent users, usage
of sessions object for state management can create load on server causing performance degradation

Application object:

Application object is used to store data which is visible across entire application and shared across multiple user sessions. Data which needs to be persisted for entire life of application should be stored in application object.

In classic ASP, application object is used to store connection strings. It's a great place to store data which changes infrequently. We should write to application variable only in application_Onstart event (global.asax) or application.lock event to avoid data conflicts. Below code sample gives idea

```
Application.Lock();
Application["mydata"]="mydata";
Application.UnLock();
```

Session object:

Session object is used to store state specific information per client basis. It is specific to particular user. Session data persists for the duration of user session you can store session's data on web server in different ways. Session state can be configured using the <session State> section in the application's web.config file.

Configuration information:

```
<sessionState mode = <"inproc" | "sqlserver" | "stateserver">
 cookieless = <"true" | "false">
 timeout = <positive integer indicating the session timeout in minutes>
 sqlconnectionstring = <SQL connection string that is only used in the SQLServer mode>
 server = <The server name that is only required when the mode is State Server>
 port = <The port number that is only required when the mode is State Server>
```

Mode:

This setting supports three options. They are InProc, SQLServer, and State Server

Cookie less:

This setting takes a Boolean value of either true or false to indicate whether the Session is a cookie less one.

Timeout:

This indicates the Session timeout vale in minutes. This is the duration for which a user's session is active. Note that the session timeout is a sliding value; Default session timeout value is 20 minutes

SqlConnectionString:

This identifies the database connection string that names the database used for mode SQLServer.

Server:

In the out-of-process mode State Server, it names the server that is running the required Windows NT service: aspnet_state.

Port:

This identifies the port number that corresponds to the server setting for mode State Server. Note that a port is an unsigned integer that uniquely identifies a process running over a network.

*You can disable session for a page using **EnableSessionState** attribute. You can set off session for entire application by setting mode=off in web.config file to reduce overhead for the entire application.*

Session state in ASP.NET can be configured in different ways based on various parameters including scalability, maintainability and availability

- In process mode (in-memory)- State information is stored in memory of web server
- Out-of-process mode- session state is held in a process called aspnet_state.exe that runs as a windows service.
- Database mode â€" session state is maintained on a SQL Server database.

In process mode:

This mode is useful for small applications which can be hosted on a single server. This model is most common and default method to store session specific information. Session data is stored in memory of local web server

Configuration information:

```
<sessionState mode="Inproc"
 sqlConnectionString="data source=server;user id=freelance;password=freelance"
 cookieless="false" timeout="20" />
```

Advantages:

- Fastest mode
- Simple configuration

Disadvantages:

- Session data will be lost if the worker process or application domain recycles
- Not ideal for web gardens and web farms

Out-of-process Session mode (state server mode):

This mode is ideal for scalable and highly available applications. Session state is held in a process called aspnet_state.exe that runs as a windows service which listens on TCP port 42424 by default. You can invoke state service using services MMC snap-in or by running following net command from command line.

Net start aspnet_state

Configuration information:

```
<sessionState mode="StateServer"
 StateConnectionString="tcpip=127.0.0.1:42424"
 sqlConnectionString="data source=127.0.0.1;user id=freelance; password=freelance"
 cookieless="false" timeout="20"/>
```

Advantages:

- Supports web farm and web garden configuration
- Session data is persisted across application domain recycles. This is achieved by using separate worker process for maintaining state

Disadvantages:

- Out-of-process mode provides slower access compared to In process
- Requires serializing data

SQL-Backed Session state:

ASP.NET sessions can also be stored in a SQL Server database. Storing sessions in SQL Server offers resilience that can serve sessions to a large web farm that persists across IIS restarts.

SQL based Session state is configured with aspnet_regsql.exe. This utility is located in .NET Framework's installed directory
 C:\<windows>\microsoft.net\framework\<version>. Running this utility will create a database which will manage the session state.

Configuration Information:

```
<sessionState mode="SQLServer"
  sqlConnectionString="data source=server;user id=freelance;password=freelance"
  cookieless="false" timeout="20" />
```

Advantages:

- Supports web farm and web garden configuration

- Session state is persisted across application domain recycles and even IIS restarts when session is maintained on different server.

Disadvantages:

- Requires serialization of objects

Choosing between client side and Server side management techniques is driven by various factors including available server resources, scalability and performance. We have to leverage both client side and server side state management options to build scalable applications.

When leveraging client side state options, ensure that little amount of insignificant information is exchanged between page requests.

Various parameters should be evaluated when leveraging server side state options including size of application, reliability and robustness. Smaller the application, In process is the better choice. We should account in the overheads involved in serializing and deserializing objects when using State Server and Database based session state. Application state should be used religiously.

Thank you for using Mindcracker Network