ARTICLE

READER LEVEL:

State Management in ASP.Net

By Abhishek Jaiswal :) on Aug 17, 2014

This article is all about how to maintain, clear or hold the state of your pages in ASP.NET applications or in web-based applications. In this article I tried to briefly summarize the concept of State Management (but am including only Client-Side State Management).

271.9 k 25 9

This article is all about how to maintain, clear or hold the state of your pages in ASP.NET applications or in web-based applications. In this article I tried to briefly summarize the concept of State Management (but am including only Client-Side State Management).

Agenda

The agenda of this article will be as follows:

- State | Overview
- State | Introduction
- State | Outline
- State Management Types
    - Client-side Management
    - Server-side Management
- State Management | Scenario
- State Management | Techniques
    - Client-side Techniques
        - View
        - Hidden
        - Cookies
        - Control State
        - Query Strings

- Server-side Techniques
  - Session State
  - Application State

State | Overview

As we all know, **browsers are generally stateless**.

Now question arises here, **what does stateless actually mean**?

Stateless means, whenever we visit a website, our browser communicates with the respective server depending on our requested functionality or the request. The browser communicates with the respective server using the HTTP or HTTPs protocol.

But after that response, **what's next or what will happen when we visit that website again** after closing our web browser?

In this case HTTP/HTTPs doesn't remember what website or URL we visited or in other words we can say it doesn't hold the state of a previous website that we visited before closing our browser, that is called stateless.

Now I guess you have at least an idea of what state and stateless actually means.

So our browsers are stateless.

State | Introduction

In this article I'll try to give you a feel of state and actually why we need states and State Management in ASP.NET. I'll take you through several State Management techniques along with their respective general case examples.

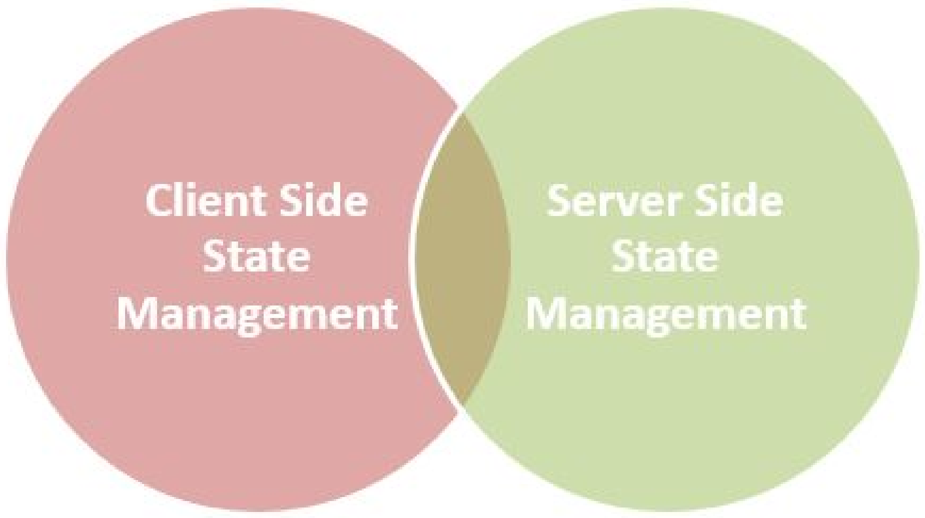
State | Outline

As I said in the beginning that HTTP is a stateless protocol. It just cleans up or we can say removes all the resources/references that were serving a specific request in the past. These resources can be:

- Objects
- Allocated Memory
- Sessions ID's
- Some URL info
  and so on.

State Management | Types

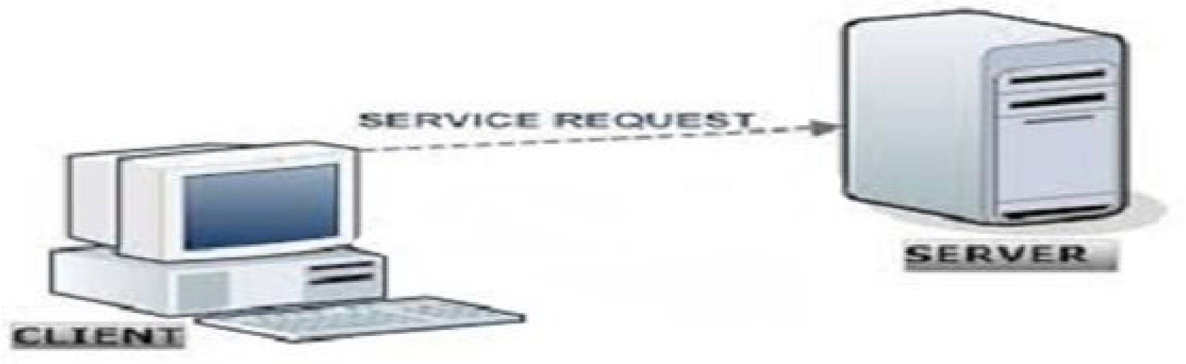
In ASP.NET there are the following 2 State Management methodologies:



Client-Side State Management

Whenever we use Client-Side State Management, the state related information will directly get stored on the client-side. That specific information will travel back and communicate with every request generated by the user then afterwards provides responses after server-side communication.

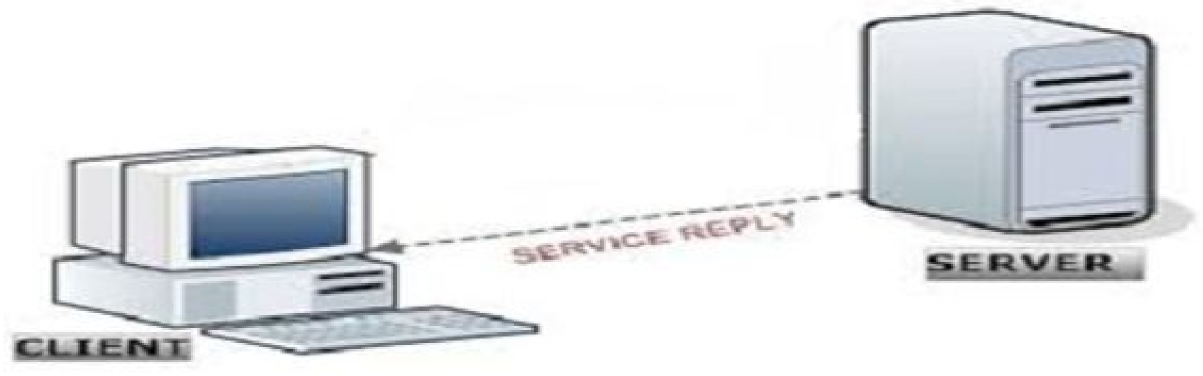
This architecture is something like the following:



Server-Side State Management

Server-Side State Management is different from Client-Side State Management but the operations and working is somewhat the same in functionality. In Server-Side State Management all the information is stored in the user memory. Due to this functionality there is more secure domains at the server side in comparison to Client-Side State Management.

The structure is something like the following:

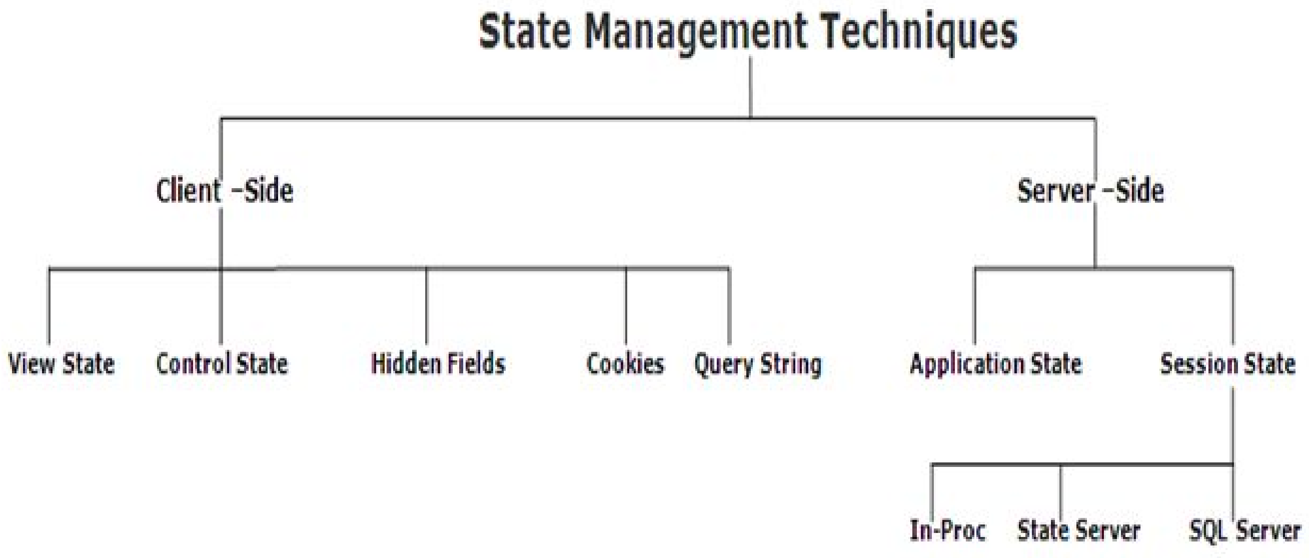


State Management | Scenario

It will be a little difficult to directly evaluate what will be better for our application. We cannot directly say that we will use client-side or server-side architecture of State Management.

State Management | Techniques

State Management techniques are based on client side and server side. Their functionality differs depending on the change in state, so here is the hierarchy:



Client-side | Techniques

Client-Side State Management techniques are:

- View State
- Hidden field
- Cookies
- Control State
- Query Strings

Server-side | Technique

Server-Side State Management techniques are:

- Session State
- Application State

Now I am defining each and every technique in detail with their reference example.

View State

In general we can say it is used for storing user data in ASP.NET, sometimes in ASP.NET applications the user wants to maintain or store their data temporarily after a post-back.. In this case VIEW STATE is the most used and preferred way of doing that.

This property is enabled by default but we can make changes depending on our functionality, what we need to do is just change the EnableViewState value to either TRUE for enabling it or FALSE for the opposite operation.

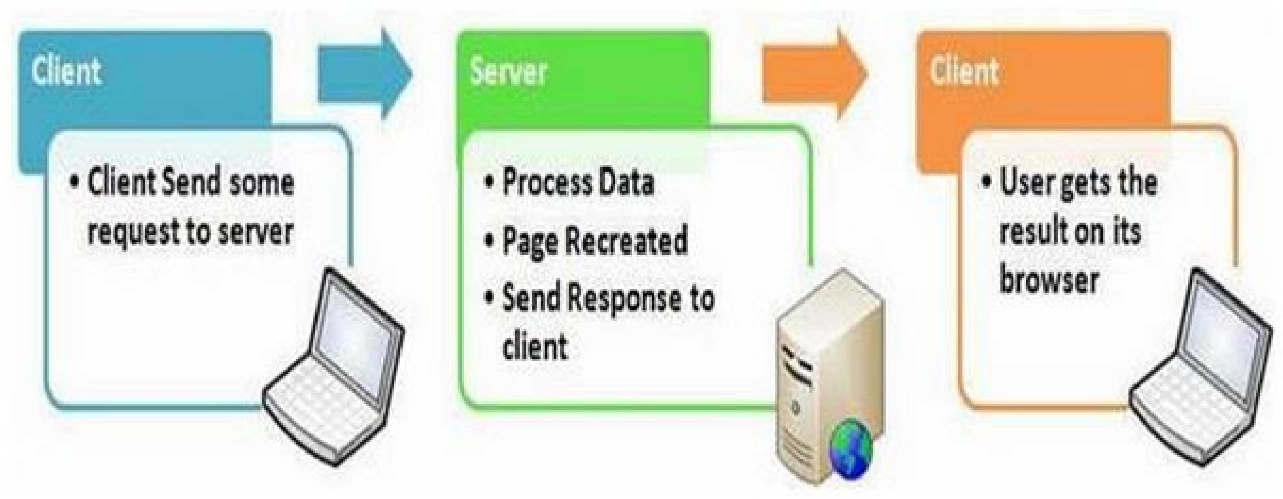

Client
• Client Send some request to server
Server
• Process Data
• Page Recreated
• Send Response to client
Client
• User gets the result on its browser

Figure: [View State Management]

Snippet

```
// Page Load Event
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        if (ViewState["count"] != null)
        {
            int ViewstateVal = Convert.ToInt32(ViewState["count"]) + 1;
            View.Text = ViewstateVal.ToString();
            ViewState["count"]=ViewstateVal.ToString();
        }
        else
        {
            ViewState["count"] = "1";
        }
    }
}
```

```

// Click Event
protected void Submit(object sender, EventArgs e)
{
       View.Text=ViewState["count"].ToString();
}
```

Points to Remember

Some of the features of view state are:

- It is page-level State Management
- Used for holding data temporarily
- Can store any type of data
- Property dependent

Hidden Field

A hidden field is used for storing small amounts of data on the client side. In most simple words it's just a container of some objects but their result is not rendered on our web browser. It is invisible in the browser.

It stores a value for the single variable and it is the preferable way when a variable's value is changed frequently but we don't need to keep track of that every time in our application or web program.

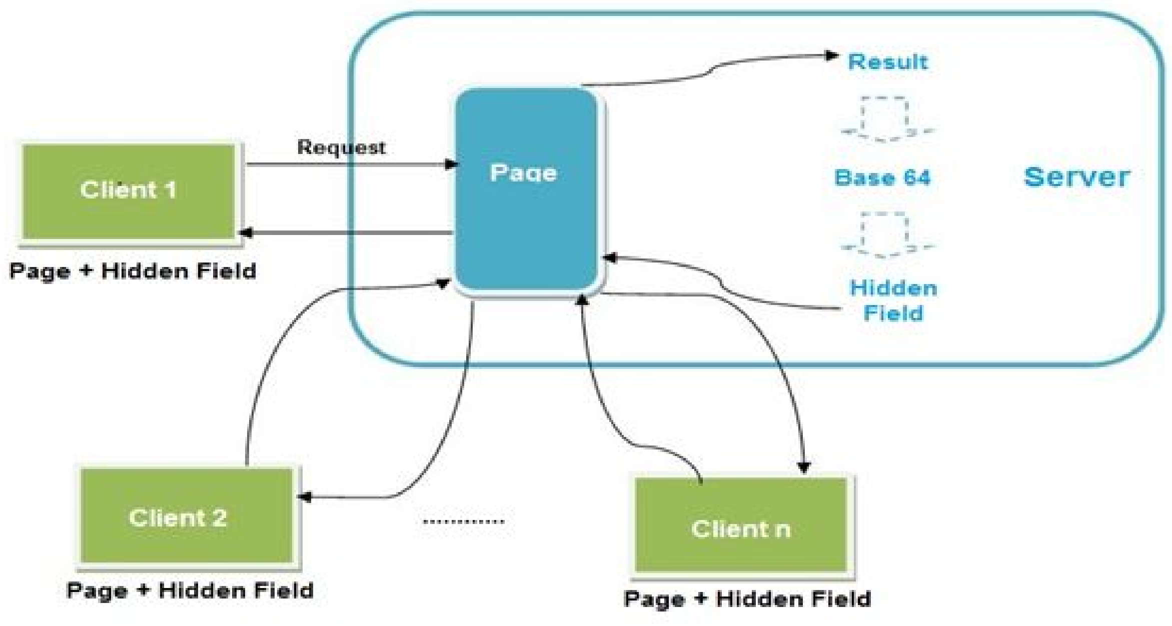


Figure: [Hidden Field Management]

Snippet

```
// Hidden Field

```

```
int newVal = Convert.ToInt32(HiddenField1.Value) + 1;
HiddenField1.Value = newVal.ToString();
Label2.Text = HiddenField1.Value;
```

Points to Remember

Some features of hidden fields are:

- Contains a small amount of memory
- Direct functionality access

Cookies

A set of Cookies is a small text file that is stored in the user's hard drive using the client's browser. Cookies are just used for the sake of the user's identity matching as it only stores information such as sessions id's, some frequent navigation or post-back request objects.

Whenever we get connected to the internet for accessing a specific service, the cookie file is accessed from our hard drive via our browser for identifying the user. The cookie access depends upon the life cycle or expiration of that specific cookie file.

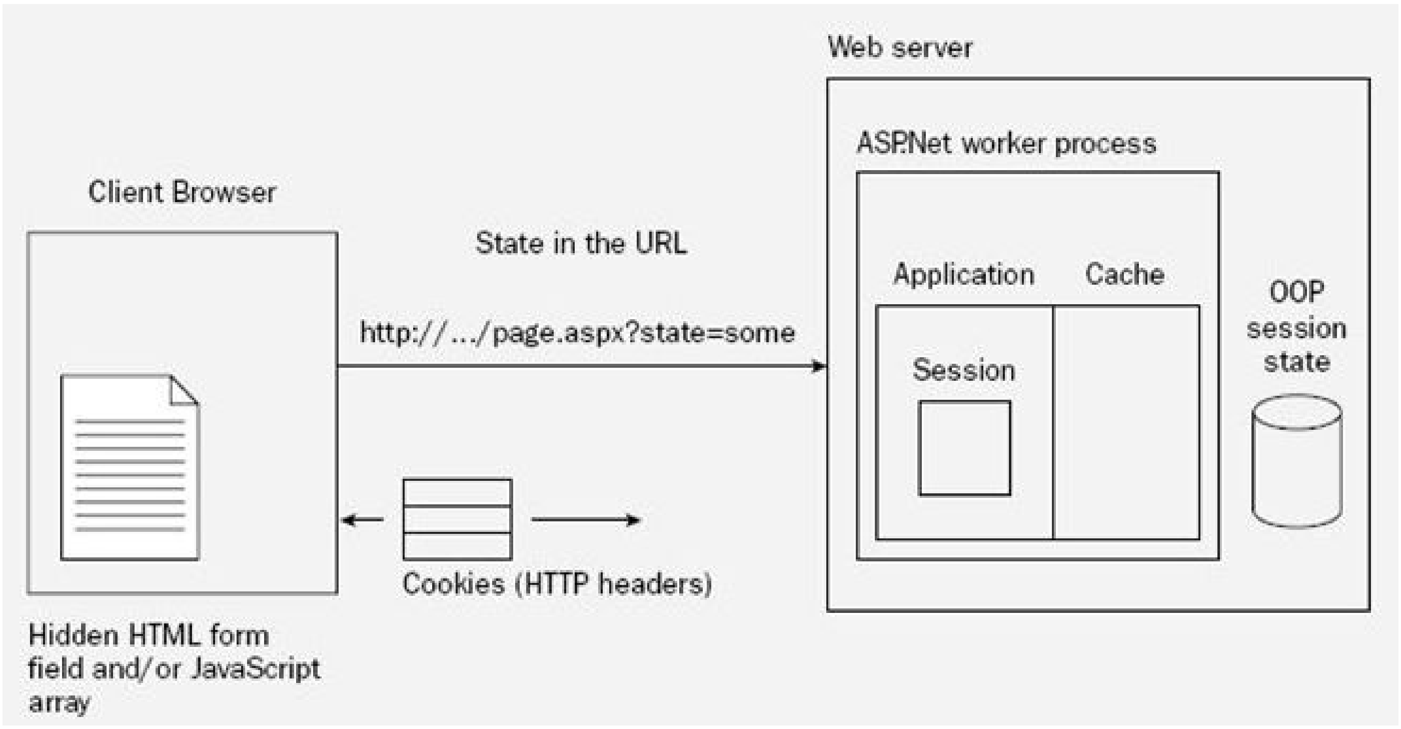
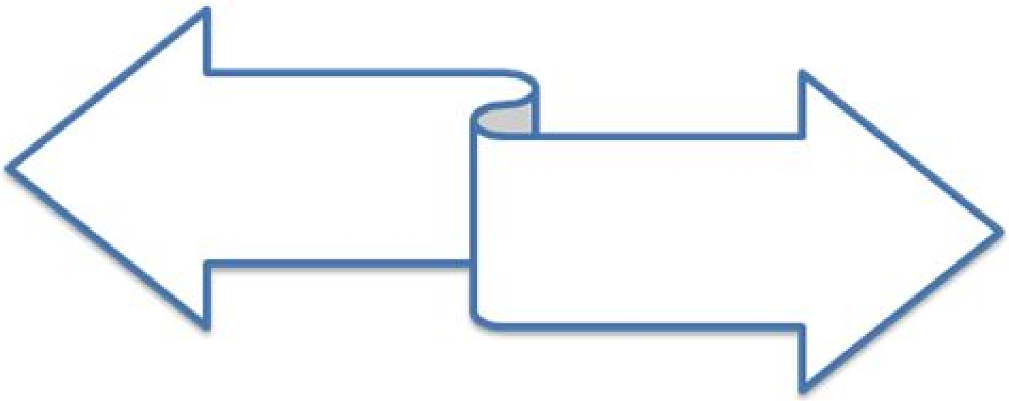


Figure: [Cookie Management]

Snippet

```
int postbacks = 0;
if (Request.Cookies["number"] != null)
{
    postbacks = Convert.ToInt32(Request.Cookies["number"].Value) + 1;
}
// Generating Response
```

C# Corner Discontinues The Code Snippet Section



Persistent Cookie

Cookies having an expiration date is called a persistent cookie. This type of cookie reaches their end as their expiration dates comes to an end. In this cookie we set an expiration date.

Snippet

```
Response.Cookies["UserName"].Value = "Abhishek";
Response.Cookies["UserName"].Expires = DateTime.Now.AddDays(1);

HttpCookie aCookie = new HttpCookie("Session");
aCookie.Value = DateTime.Now.ToString();
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

Non-Persistent Cookie

Non-persistent types of cookies aren't stored in the client's hard drive permanently. It maintains user information as long as the user access or uses the services. Its simply the opposite procedure of a persistent cookie.

Snippet

```
HttpCookie aCookie = new HttpCookie("Session");
aCookie.Value = DateTime.Now.ToString();
aCookie.Expires = DateTime.Now.AddDays(1);
Response.Cookies.Add(aCookie);
```

Points to Remember

Some features of cookies are:

- Store information temporarily
- It's just a simple small sized text file
- Can be changed depending on requirements

- User Preferred
- Requires only a few bytes or KBs of space for creating cookies

Control State

Control state is based on the custom control option. For expected results from CONTROL STATE we need to enable the property of view state. As I already described you can manually change those settings.

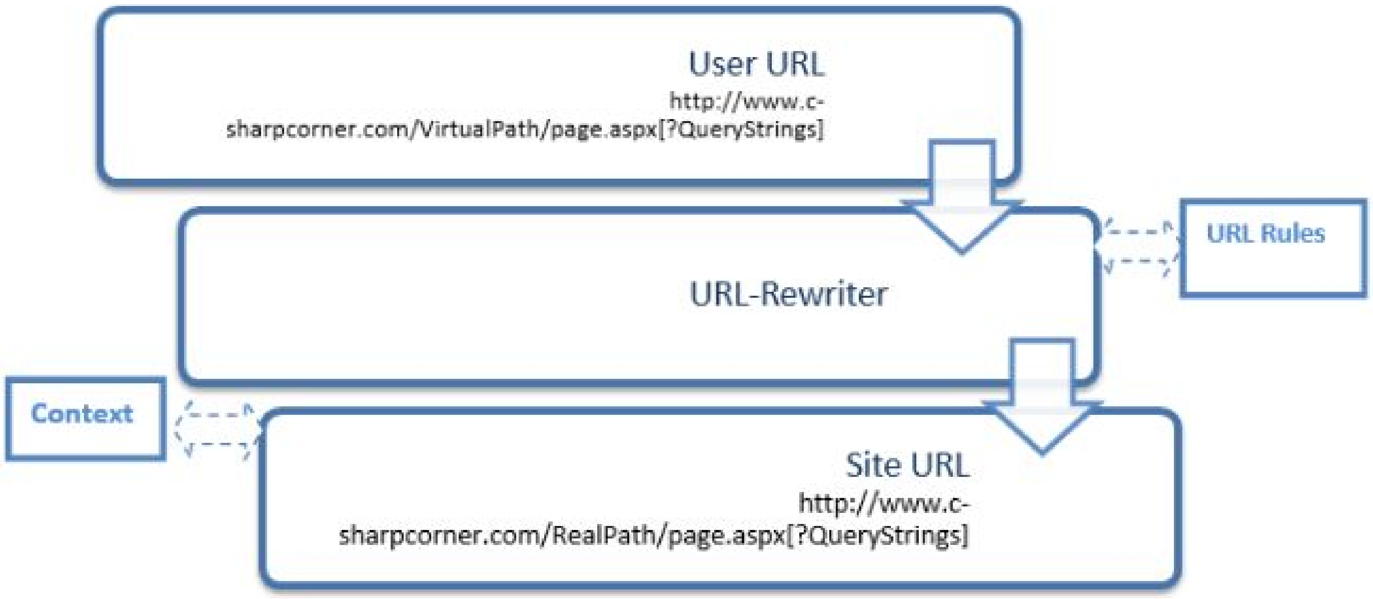
Points to Remember

Some features of query strings are:

- Used for enabling the View State Property
- Defines a custom view
- View State property declaration
- Can't be modified
- Accessed directly or disabled

Query Strings

Query strings are used for some specific purpose. These in a general case are used for holding some value from a different page and move these values to the different page. The information stored in it can be easily navigated to one page to another or to the same page as well.


User URL
http://www.c-sharpcorner.com/VirtualPath/page.aspx[?QueryStrings]
URL-Rewriter
URL Rules
Context
Site URL
http://www.c-sharpcorner.com/RealPath/page.aspx[?QueryStrings]

[Query Strings]

Figure: [Query Strings]

Snippet

```
// Getting data
if (Request.QueryString["number"] != null)
{
    View.Text = Request.QueryString["number"];
}

// Setting query string
int postbacks = 0;

if (Request.QueryString["number"] != null)
{
    postbacks = Convert.ToInt32(Request.QueryString["number"]) + 1;
}
else
{
    postbacks = 1;
}

Response.Redirect("default.aspx?number=" + postbacks);
```

Points to Remember

Some of the features are:

- It is generally used for holding values
- Works temporarily
- Switches info from one to another page
- Increase performance
- Uses real and virtual path values for URL routing

I hope you have enjoyed reading this article. I'll write about Server-Side State Management very soon.

ASP.NET Client Side Techniques Server Side Techniques State Management

ABHISHEK JAISWAL :) TOP 50

Geek | Blogger | Data Science Scholar | TechSavvy | Developer | Painter | Author | Trainer | Tech Evangelist | Philanthropist

http://letustweak.com/

50 Rank 2.8m Read Platinum Member 3 Times

COMMENTS (25)

TRENDING UP

01 Using Generics With C#

02 When To Use Abstract Class and Interface In Real Projects

03 Web Crawling With C#

04 What Is The Future Of XAML

05 Best Practices For ASP.NET MVC Application

06 C# Version 7.0 Announcements

07 What Is New In Visual Studio "15" Preview 4

08 Introduction To .NET Core 1.0

09 Factory Design Pattern Real World Example

10 Highest Paying Tech Jobs Of 2016

View All

Follow @csharpcorner 75.1K followers

C# Corner
402,589 likes

2 friends like this