How to track traffic of your web site?

If you have control over the code on your **ASP**.**NET** site, there is an easy way:

The Session Context

The Session context nicely encapsulates the total time a visitor is on the site. As we mentioned earlier, the nature of HTTP protocol is that is connections are not maintained. Once all the content for a page request is transferred, the connection is terminated. ASP circumnavigates this problem by maintaining a session through the use of cookies. When a visitor first enters a site, they are given a unique id in a cookie. When they return within a fixed amount of time, the cookie is returned, making it possible for the server to identify them as the same visitor. Of course, this technique is useless if a visitor's browser has cookie support turned off. My experience, however, is that this is very rare. Fortunately, though, ASP.NET provides a work around by supporting cookieless sessions.

Keep in mind that only browsers support cookies, so when robots visit your site, each request will initiate a new session.

The Session context not only can hold values across page requests, it can also first fire events on Session Start and on Session End. These will be very useful in setting up our tracking object and handling notification and reporting.

The Request Object

The Request object gives us programmatic access to many of the same data items that are stored in the log file such as the requested URL, the Referrer, the UserHostAddress (IP), and the UserAgent. In addition, the object provides a Browser Capabilities object, which can give you very detailed information about what is implied by the user agent string.

Cookies

Cookies are a small collection of name and value pairs that can be set by a server to be stored on the visitor's computer by the browser. Cookies can be set on the server with the Response object and retrieved later by the Request object. As we mentioned, ASP has always used cookies to facilitate the management of a visitor's session. You can use cookies to manage tracking data across sessions. For our tracking purposes, we will store three pieces of data in cookies: the number of visits, the original URL and referrer requested by the visitor on their first visit.

Also, cookies are great way to keep track of the activities of registered users. You can store other identifying information about a visitor or user and associate this information with your tracking data. The less anonymous your visitors are, the better able you will be

to target particular demographics and eventually convert more visitors to customers. However, if you are storing sensitive data in cookies, you should look at protecting the data with some form of encryption. Cookie data is passed in the clear over non SSL connections.

Adding Comments to the IIS Log

A simple way to provide more information to the IIS log files is to append it yourself with the Response.AppendToLog method. By using Session and Application events you can place keywords that you can later use when search the files. You might even be able to train some of the analysis programs to understand you keywords and hopefully provide more meaningful stats.

```
Sub Application_BeginRequest(ByVal sender As Object, ByVal e As EventArgs)
    ' Fires at the beginning of each request
    Response.AppendToLog("test")
End Sub
```

There are some restrictions, however. Since the data you provide is appended to the URI Query portion of the log file, you are limited to 80 characters. In addition, you cannot use commas since they are a delimiter for some of the log file formats. Also, if you anticipate that there will be other querystring elements you may want to prepend your string with an ampersand.

Creating a Session Tracker Class in **ASP**.**NET**

Now we can finally get to some code. I have created a simple **ASP**.**NET** Web application using VB.**Net**. The application contains six **ASP**.**NET** Web forms that hold a menu user control to simplify navigation between the pages. Remember, our purpose to track activity.

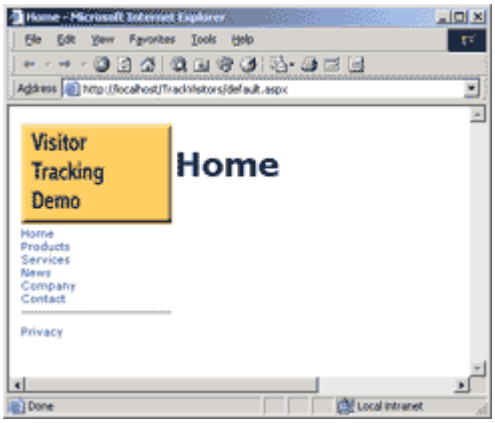

Fig 2 - The Home Page

The key element of the example application is the SessionTracker class. The class is designed to assemble all the necessary tracking data on its own and provide that data for reporting through a number of read-only properties:

VisitCount</TD	Number of times the visitor have visited the site
OriginalReferrer</TD	The Referrer from the visitor's first visit
OriginalURL</TD	The Requested URL from the visitor's first visit
SessionReferrer</TD	The Referrer for the current visitor session
SessionURL</TD	The Requested URL for the current visitor session
SessionUserHostAddress</TD	The IP Address of the visitor
SessionUserAgent</TD	The Browser or other application
Browser</TD	A reference to the HTTPBrowserCapabilites object which provides additional information inferred from the UserAgent
Pages</TD	an ArrayList of the names and timestamps of the ASP.NET web forms viewed during the sessions

Initializing the class

In the Session_OnStart method, we create an instance of the class.

```
Dim tracker As SessionTracker = New SessionTracker()
Session("Tracker") = tracker
```

When the New constructor method is called as the class is created, we grab an instance of the current HTTPContext with the HTTPContext.Current static method call. This allows us to get access to the Request and Response objects for acquiring the request info and cookies. A reference to the HTTPContext is held as a member variable. Three helper functions are used to deal with the cookie data: incrementVisitCount, setOriginalReferrer, and setOriginalURL. We also set a default expiration time to be used with all our cookies.

```
Public Class SessionTracker
    Private _context As HttpContext
    Private _expires As Date

    Private _VisitCount As String

    Private _UserHostAddress As String
    Private _UserAgent As String

    Private _OriginalReferrer As String
    Private _OriginalURL As String

    Private _SessionReferrer As String
    Private _SessionURL As String
```

```
Private _browser As HttpBrowserCapabilities

Private _pages As New ArrayList()

Public Sub New()
    'HttpContext.Current allows us to gain access to all
    'the intrinsic ASP context objects like Request, Response, Session, etc
    _context = HttpContext.Current

    'provides a default expiration for cookies
    _expires = Now.AddYears(1)

    'load up the tracker
    incrementVisitCount()

    _UserHostAddress = _context.Request.UserHostAddress.ToString
    _UserAgent = _context.Request.UserAgent.ToString

    If Not IsNothing(_context.Request.UrlReferrer) Then
        'set original referrer if not set
        setOriginalReferrer(_context.Request.UrlReferrer.ToString)
        _SessionReferrer = _context.Request.UrlReferrer.ToString
    End If

    If Not IsNothing(_context.Request.Url) Then
        'set original url if not set
        setOriginalURL(_context.Request.Url.ToString)
        _SessionURL = _context.Request.Url.ToString
    End If

    'set the browser capabilities
    _browser = _context.Request.Browser

End Sub

'increment the visit count and save in a cookie
Public Sub incrementVisitCount()
    Const KEY = "VisitCount"

    'check is cookie has been set yet
    If IsNothing(_context.Request.Cookies.Get(KEY)) Then
        _VisitCount = 1
    Else
        _VisitCount = _context.Request.Cookies.Get(KEY).Value + 1
    End If

    'set or reset the cookie
    addCookie(KEY, _VisitCount)
End Sub

'set the original referrer to a cookie
Public Sub setOriginalReferrer(ByVal val As String)
    Const KEY = "OriginalReferrer"

    'check is cookie has been set yet
    If Not IsNothing(_context.Request.Cookies.Get(KEY)) Then
```

```
            _OriginalReferrer = _context.Request.Cookies.Get(KEY).Value
        Else
            addCookie(KEY, val)
            _OriginalReferrer = val
        End If
    End Sub

    'set the original url to a cookie
    Public Sub setOriginalURL(ByVal val As String)
        Const KEY = "OriginalURL"

        'check is cookie has been set yet
        If Not IsNothing(_context.Request.Cookies.Get(KEY)) Then
            _OriginalURL = _context.Request.Cookies.Get(KEY).Value
        Else
            addCookie(KEY, val)
            _OriginalURL = val
        End If
    End Sub

    'add the page to an arraylist in the session
    Public Sub addPage(ByVal pageName As String)
        'create a new page tracker item
        Dim pti As New SessionTrackerPage()
        pti.PageName = pageName
        'set a time stamp
        pti.Time = Now

        'add the page tracker item to the array list
        _pages.Add(pti)
    End Sub


    Private Sub addCookie(ByVal key As String, ByVal value As String)
        Dim cookie As HttpCookie
        cookie = New HttpCookie(key, value)
        cookie.Expires = _expires
        _context.Response.Cookies.Set(cookie)
    End Sub


#Region "Properties"

    'Visit Count
    ReadOnly Property VisitCount() As Integer
        Get
            Return _VisitCount
        End Get
    End Property

    'Original Referrer
    ReadOnly Property OriginalReferrer() As String
        Get
            Return _OriginalReferrer
        End Get
    End Property
```

```
    'Original URL
    ReadOnly Property OriginalURL() As String
        Get
            Return _OriginalURL
        End Get
    End Property

    'Session Referrer
    ReadOnly Property SessionReferrer() As String
        Get
            Return _SessionReferrer
        End Get
    End Property

    'Session URL
    ReadOnly Property SessionURL() As String
        Get
            Return _SessionURL
        End Get
    End Property

    'Session User Host Address (IP)
    ReadOnly Property SessionUserHostAddress() As String
        Get
            Return _UserHostAddress
        End Get
    End Property

    'Session User Agent
    ReadOnly Property SessionUserAgent() As String
        Get
            Return _UserAgent
        End Get
    End Property

    'Pages - array list
    ReadOnly Property Pages() As ArrayList
        Get
            Return _pages
        End Get
    End Property

    'Browser Cap
    ReadOnly Property Browser() As HttpBrowserCapabilities
        Get
            Return _browser
        End Get
    End Property

#End Region

End Class
```

Storing the SessionTracker in the Session context

Being able to create a class like the SessionTracker and store it in the Session context is one of the great advantages of ASP.NET. Of course, in ASP 3.0 we could have easily created a similar COM object; however, if it was developed with VB, we could not have saved it in the Session. In case you were not aware of the problems in saving Single Threaded Apartment (STA) components in a session or application context, read the following.

http://support.microsoft.com/default.aspx?scid=KB;en-us;q243543

Tracking Each Page View

Our next goal is to track every page that our visitors hit. This can be done easily by utilizing one of the Application event handlers in the Global.asax.

Application_PreRequestHandlerExecute

There are two events that are called on the application context before a page request begins: Application_PreRequestHandlerExecute and Application_BeginRequest. Unfortunately, the session context is not available when the Application_BeginRequest is called (I suspect this is a minor bug). For this reason we use the PreRequestHandlerExecute event. In the event, we extract the tracker from the session and pass the current URL through the addPage method.

```
   Sub Application_PreRequestHandlerExecute(ByVal sender As Object, _
ByVal e As EventArgs)
      If Not IsNothing(Context.Session) Then
         Dim tracker As SessionTracker
         tracker = Session("Tracker")

         'kick out if there is no session tracker
         If Not IsNothing(tracker) Then
            tracker.addPage(Request.Url.ToString())
         End If
      End If
   End Sub
```

Remember, only requests for ASPX files will initiate this event.

AddPage method

In the SessionTracker's addPage method, we receive the name of the page then create an instance of the small SessionTrackerPage class which has two public members:

PageName and the Time.

Of course, we could have accessed the current request URL string from the HTTPContext in the class; however, this approach means you can manipulate the URL as you see fit. For instance, you can crop it down to just the file name. Or replace the file name by using a Map collection to a reference a common name for the page.

Sending Notifications

Now that we have this data neatly collected, it would be nice if we had some way to see it.

Sending e-mails to notify webmaster of activity

E-mail notifications can give you are a real-time sense of the activity patterns on the site. It can also make you a little nervous when you haven't received an e-mail in a while.

The MailUtil class

To simplify the e-mail notification, I've encapsulated the functionality into a MailUtil class. Since the object does not require any state, the class has two public shared methods: SendSessionStartAlert and SendSessionEndAlert. These functions should be called from the Session_OnStart and the Session_OnEnd events.

```
Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
    Try
        Dim tracker As SessionTracker
        tracker = Session("Tracker")

        'kick out if there is no session data
        If IsNothing(tracker) Then
            Exit Sub
        Else
            MailUtil.SendSessionEndAlert(tracker)
        End If
    Catch

    End Try
End Sub
```

Using configuration settings

In order to keep the MailUtil class very flexible, I used the appSetting section of the web.config file to set a number of parameters for the e-mail process. This allows you to quickly change the behavior of the notifications with having to change code.

```
<appSettings>
    <add key="emailAlertOnSessionStart" value="true"/>
    <add key="emailAlertOnSessionEnd" value="true"/>
    <add key="SMTPServer" value=""/>
```

```
    <add key="AlertEmail" value="me@my.email.com"/>
  </appSettings>
```

Be sure to change the e-mail once you set up the sample application.

SendSessionStartAlert method

The SendSessionStartAlert method is passed to the SessionTracker object. From there we assemble the mail message.

```
Public Shared Sub SendSessionStartAlert(ByVal tracker As SessionTracker)
    If ConfigurationSettings.AppSettings("emailAlertOnSessionStart") = "true" Then
        Dim alertEmail As String = ConfigurationSettings.AppSettings("alertEmail")
        If Not IsNothing(alertEmail) Then

            Try
                Dim msg As New MailMessage()

                msg.From = alertEmail
                msg.To = alertEmail

                msg.Subject = "Visitor Alert: " & tracker.SessionUserHostAddress

                msg.Body = createTrackerMessageBody(tracker)
                msg.BodyFormat = MailFormat.Html
                MailUtil.Send(msg)
            Catch exc As Exception
                'no need to do anything since this is not a critical function
                Debug.WriteLine(exc.ToString)

            End Try
        End If
    End If
End Sub
```

Note that I use HTML for the BodyFormat. The body of the e-mail is assembled as HTML by the createTrackerMessageBody method. See the **MailUtil.vb** file in the downloadable source code.

One of the interesting pieces of information provided in the e-mail are the WHOIS and NSLOOKUP links in the e-mail output. You might say this is a poor man's way of researching visitors by automating links to the following public web services.

ARIN	American Registry for Internet Numbers manage the Internet numbering resources for North and South America, the Caribbean, and sub-Saharan Africa.http://arin.**net**/
RIPE	Réseaux IP Européens - Large WHOIS database for Europe and Asiahttp://ripe.**net**/ripencc/pub-services/db/whois/whois.html
NSLookup or	http://www.zoneedit.com/lookup.html

Reverse DNS	

SendSessionEndAlert method

The Session end e-mail is very similar to the Session start e-mail, except at the session end we can provide a listing of the pages visited. Fig 3 is an example of a session end notification.

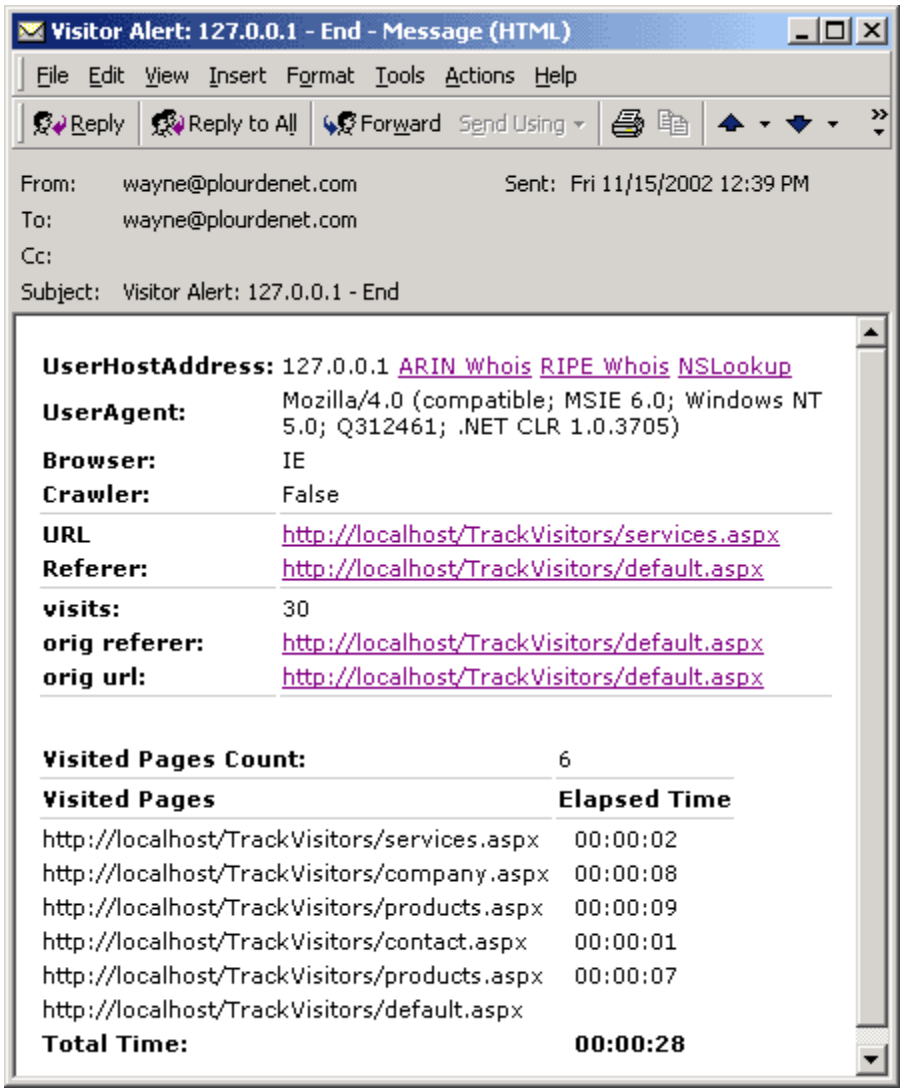

Fig 3 - Session Email Notification

System.Web.Mail

The core of our e-mail functionality is from the .**Net** Framework's System.Web.Mail namespace. We use both the MailMessage object and the static SMTPServer object.

On some machines, you may have a problem getting the e-mail to send properly. If this is the case, make sure that the SMTP service is running. You can find the SMTP

configuration in the Internet Service Manager. If it is running, then you need to make sure that the SMTP Server is configured properly to allow e-mail to be sent from the local machine. Under the "Access" tab, select Relay Restrictions. Then either add settings to allow localhost or 127.0.0.1 or check the "Allow all computers which successfully authenticate to relay, regardless of the list above" option.

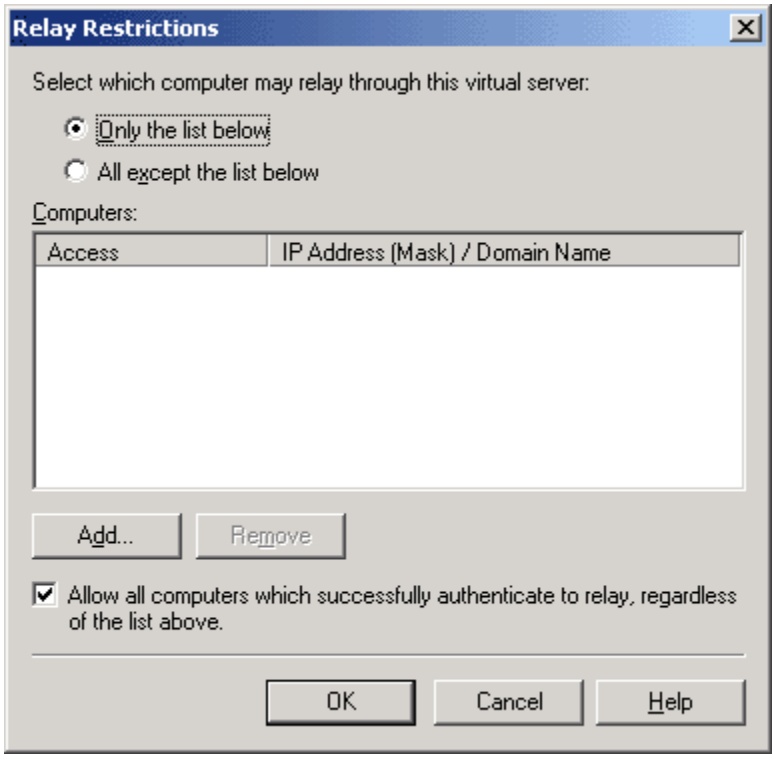

Fig4 - SMTP configuration

If this is not possible, you can always specify an SMTPserver on another machine, if available.

Creating Your Own Log File

If all this e-mail stuff makes you nervous, you can always create your own log file. There are just a few considerations you should keep in mind if you decide to do so.

- Make sure your app settings allow you to write to the file system.
- Make sure you serialize writing to the file. In a multi-user environment, each thread must wait its turn before writing. You can accomplish this by calling Application.Lock directly before the write and Application.Unlock directly after the write.
- Although the IIS logging function is highly optimized to have little affect on the performance of the sites, you still may want to turn it off. No need to use resources to write to two files.
- Optimize your log file access by storing the file stream object in the application context. This way the file can remain open for faster writes.

Putting the data in a high-end database may be a better idea since it will remove some of the contention issues and allow you to create dynamic queries on the data. But whether you use a file or a database, remember that most hosting services have a limit on the amount of disk space or database space you can use. Exceeding your limits can compromise your site.

Excluding Items from Notifications and Logging

Once everything is set up, you can decide on what data to filter where. For instance, you may want to create some filter logic to only send e-mails when the visitor is a particular robot you are expecting. Everything that is not a robot can be put in the custom log file. You can also set up exclusion filters. For instance, you may want to exclude logging for traffic that comes from yourself. Or, if you use a site-monitoring tool to ensure your sites operation, you can eliminate that as well.

Changing Your Session Timeout for Testing

The sessionState element of the web.config file provides easy access to the session timeout without having to write code.

```
<sessionState
        mode="InProc"
        stÿteConnectionString="tÿpip=127.0.0.1:42424"
        sqlConnectionString="data source=127.0.0.1;user id=sa;password="
        cookieless="false"
        timeout="1"
   />
```

I recommend setting the timeout to 1 minute while testing the implementation of your tracking system. Be sure to change it back to your designated default before sending your site to its production host.

Establishing Privacy Policies

When you start collecting data about visitors it is important to protect your interests by disclosing to your visitors what data you are collecting and what you plan to do with the data. This is called a privacy policy. You simply need to create a page with your statement and link to it from your homepage or any forms that collect data on the site. Here is a link to JupiterMedia's (parent of 15seconds.com) privacy policy:

http://www.internet.com/corporate/privacy/privacypolicy.html

Privacy policies are especially important if your site is geared towards children. In this case, it falls under the jurisdiction of the Children's Online Privacy Protection Act, which has very strict guidelines about what you can and can't do with information you collect on your site. Here is an excellent article that gives an overview of the subject:

http://html.about.com/library/weekly/aa043001a.htm

Controlling privacy in Internet Explorer 6

Browsers are now stepping in and providing tools to alert you to privacy issues that you might not normally be aware of. Here is an overview of the many privacy features in Internet Explorer 6.

http://www.microsoft.com/windows/ie/evaluation/overview/privacy.**asp**

Platform for Privacy Preferences

The Platform for Privacy Preferences or P3P has established a standard for creating both natural language and XML based privacy documents. The following article from MSDN describes the process for deploying a privacy policy on your site.

http://msdn.microsoft.com/library/default.**asp**?url=/workshop/security/privacy/overview/createprivacypolicy.**asp**

P3P files

There are also a number of free P3P editors

http://www.alphaworks.ibm.com/tech/p3peditor

I have created both a privacy.htm and a privacy.xml file using the Alpha works editor and have included then into the sample web app. Once you have created the files, just link the HTML page to your home page and target the xml file in the head of each of your content pages as follows:

```
<meta http-equiv="P3P" content='policyref="privacy.xml"'>
<link rel="P3Pv1" href="privacy.xml" type="text/xml">
For a broad overview of P3P issues, read the following:
http://html.about.com/library/weekly/aa040802a.htm
```

Deploying the Sample Project

There are just a few things you need to do.

- Create a new virtual directory in the Internet Services Manager
- Make sure the application has been created under the new Virtual Directory's properties dialog.
- Expand the zip file into the directory
- Open the solution file in Visual Studio .Net to build and run the app.

Conclusion

The more you know about your potential customers the better able you will be to convert them into paying customers. Now there is no excuse not to know who is coming to your site.

Thanks to: Wayne Plourde
Published by www.khademi.com Ali Khademi