



فصل ششم :

بکار گیری global.asax و بحث های آماری سایت

مقدمه :

یکی از فایلهایی که همواره توسط VS.NET به صورت خودکار ایجاد می شود global.asax است و برای تعریف اشیاء عمومی که برنامه ی وب از آنها استفاده می کند بکار می رود. از این فایل برای مدیریت پردازش هایی در سطوح بالاتر برنامه مانند Application_Start و Application_End و غیره استفاده می شود. این فایل در مرورگر وب توسط سایر کاربران قابل اجرا و مشاهده نیست. لازم به ذکر است که استفاده از این فایل اختیاری بوده و در صورت نیاز از آن استفاده می شود.

شرح رخدادهای تعریف شده درون global.asax :

: Application_Init

این رویداد هر بار که یک شیء جدید از HttpApplication ساخته می شود اجرا می گردد. کدهای آغاز گر را می توان در این قسمت تعریف کرد.

: Application_Disposed

این رویداد درست قبل از تخریب هر شیء ایجاد شده از HttpApplication فراخوانی می گردد. عموماً از آن برای پاکسازی منابع تخصیص داده شده استفاده می گردد.



: Application_Error

هنگامیکه خطایی مدیریت نشده رخ می دهد این رویداد فراخوانی می گردد.

: Application_Start

این رخداد تنها یکبار در طول عمر برنامه ی وب اتفاق می افتد و آن هنگامی است که اولین شیء HttpApplication ساخته می شود. از این رخداد برای برای به اشتراک گذاشتن یک شیء در بین تمام سشن ها استفاده می شود. در این رخداد متغیرهای محلی را تعریف نکنید زیرا متغیرهای محلی بین اشیاء HttpApplication به اشتراک گذاشته نمی شوند.

: Application_End

این رخداد تنها یکبار آنهم هنگامیکه آخرین شیء HttpApplication تخریب می شود اتفاق می افتد. از این رخداد برای پاکسازی اشیاء به اشتراک گذاشته شده می توان استفاده کرد.

: Application_BeginRequest

این رخداد اولین رخدادی است که ASP.NET هنگامیکه به یک درخواست پاسخ می دهد اتفاق می افتد.

: Application_EndRequest

این رخداد آخرین رخدادی است که ASP.NET هنگامیکه به یک درخواست پاسخ می دهد اتفاق می افتد.

: Application_PreRequestHandlerExecute

این رخداد درست قبل از اجرای یک اداره کننده (Handler) مانند صفحه یا web service رخ می دهد. در این زمان ، سشن مهیا شده است.

: Application_PostRequestHandlerExecute

این رخداد زمانی اتفاق می افتد که یک اداره کننده اجرایش خاتمه می یابد.



: Application_PreSendRequestHeader

این رخداد درست قبل از فرستاده شدن HttpHeaders به کلاینت ها رخ می دهد.

: Application_PreSendRequestContext

این رخداد درست قبل از اینکه ASP.NET محتویات صفحه را به کلاینت ها بفرستد رخ می دهد.

: Application_AcquireRequestState

این رخداد هنگامی اتفاق می افتد که ASP.NET وضعیت جاری را ذخیره می کند (مانند وضعیت سشن مربوط به درخواست جاری).

: Application_ReleaseRequestState

این رخداد هنگامی اتفاق می افتد که ASP.NET اجرای تمامی اداره کننده های درخواست ها را تمام کرده است. این رخداد سبب می شود که ماژولهای حالت ، داده های وضعیت جاری را ذخیره کنند.

: Application_ResolveRequestCache

این رخداد هنگامی اتفاق می افتد که ASP.NET رخداد تعیین اعتبار را تمام کرده است و اجازه می دهد به ماژولهای caching تا اجرای اداره کننده ها را متوقف کرده و خود ارائه ی سرویس نمایند. این مورد سبب بالارفتن کارایی سایت می گردد و می توان با استفاده از آن قضاوت کرد که آیا محتویات از Cache خوانده می شوند یا خیر؟

: Application_UpdateRequestCache

این رخداد هنگامی اتفاق می افتد ASP.NET اجرای یک اداره کننده را تمام می کند و به ماژولهای Caching اجازه می دهد تا مواردی را که باید در درخواست های بعدی از Cache خوانده شوند ذخیره کنند.



: Application_AuthenticationRequest

این رخداد هنگامی رخ می دهد که مازول امنیتی مشخص مشخص کرده است که کاربر جاری معتبر است.

: Application_AuthorizeRequest

این رخداد هنگامی رخ می دهد که مازول امنیتی مشخص کرده است که کاربر تعیین اعتبار شده مجوز دسترسی به منابع را دارد یا خیر؟

: Session_Start

این رخداد هر بار که یک کاربر جدید به وب سایت دسترسی پیدا می کند ، فراخوانی می گردد.

: SessionEnd

این رخداد هنگامیکه سشن یک کاربر زمانش تمام می شود و یا خاتمه می یابد فراخوانی می گردد.

ترتیب رویدادن رخدادهای ذکر شده در سطح Application :

- BeginRequest
- AuthenticateRequest
- AuthorizeRequest
- ResolveRequestCache
- AcquireRequestState
- PreRequestHandlerExecute
- PreSendRequestHeaders
- PreSendRequestContent
- (actual processing)
- PostRequestHandlerExecute
- ReleaseRequestState
- UpdateRequestCache
- EndRequest



این ترتیبی است که رویدادهای نامبرده شده یکی پس از دیگری اتفاق می افتند. منشاء آنها شیء `HttpApplication` و یا شیء `HttpModule` است که در `Web.Config` و یا `Machine.Config` تعیین می شوند.

مثال ۱ :

یک پروژه جدید وب اپلیکیشن را آغاز نمایید و سپس کد زیر را در فایل `global.asax` آن بنویسید:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Web;
using System.Web.SessionState;

namespace ex01
{
    /// <summary>
    /// Summary description for Global.
    /// </summary>
    public class Global : System.Web.HttpApplication
    {
        public Global()
        {
            InitializeComponent();
        }

        protected void Application_Start(Object sender, EventArgs e)
        {
            Application["Hits"] = 0;
            Application["Sessions"] = 0;
            Application["TerminatedSessions"] = 0;
        }

        protected void Session_Start(Object sender, EventArgs e)
        {
            Application.Lock();
            Application["Sessions"] = (int) Application["Sessions"] + 1;
            Application.Unlock();
        }

        //The BeginRequest event is fired for every hit to every page in the site
        protected void Application_BeginRequest(Object sender, EventArgs e)
        {
            Application.Lock();
            Application["Hits"] = (int) Application["Hits"] + 1;
            Application.Unlock();
        }
    }
}
```



```
}

protected void Application_EndRequest(Object sender, EventArgs e)
{
}

protected void Application_AuthenticateRequest(Object sender, EventArgs e)
{
}

protected void Application_Error(Object sender, EventArgs e)
{
}

protected void Session_End(Object sender, EventArgs e)
{
    Application.Lock();
    Application["TerminatedSessions"] =
        (int) Application["TerminatedSessions"] + 1;
    Application.Unlock();
}

protected void Application_End(Object sender, EventArgs e)
{
    //Write out our statistics to a log file
    //...code omitted...
}

#region Web Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
}
#endregion
}
}
```

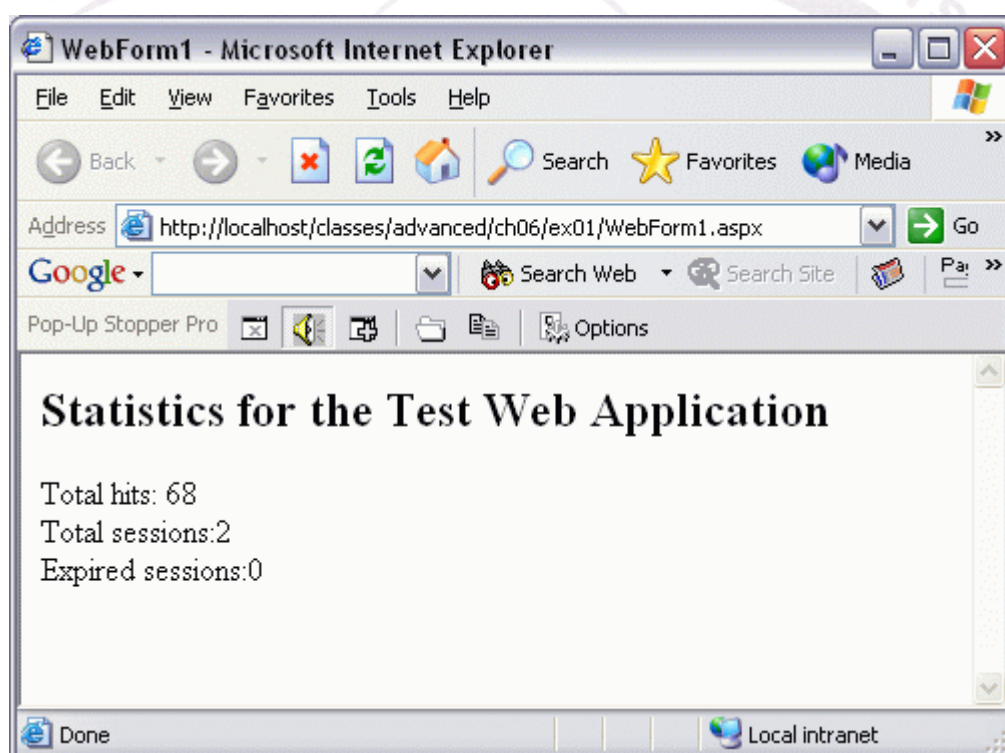
اکنون در رویداد page_load وب فرم کد زیر را برای نمایش آمار سایت بنویسید:

```
private void Page_Load(object sender, System.EventArgs e)
{
    Response.Write("<h2> Statistics for the Test Web Application </h2>");
    Response.Write(" Total hits: ");
    Response.Write(Application["Hits"].ToString());
    Response.Write("<br> Total sessions:");
    Response.Write(Application["Sessions"].ToString());
    Response.Write("<br> Expired sessions:");
}
```

```
Response.Write(Application["TerminatedSessions"].ToString());  
}
```

کد فوق رخدادهایی را که تاکنون بررسی کرده ایم را به شکلی عملی نمایش می دهد. همانطور که پیش تر نیز ذکر شد ، BeginRequest هر بار که صفحه مشاهده می شود اتفاق می افتد و از آن برای نوشتن Hit Counter ها می توان استفاده نمود. هنگامیکه صفحه اجرا شد ، آنرا چندبار ریفرش کنید تا نتیجه را ملاحظه نمایید.

برای اینکه چیزی شبیه به قسمت Who is on line? را که در اکثر فروم ها می توان مشاهده کرد ایجاد نماییم، می توان از Session_Start و End استفاده کرد. با استفاده از این رخدادهای می توان دریافت که چه نوع کاربران متفاوتی به سایت رجوع کرده اند. با کمی توسعه ی کد فوق می توان بحث های آماری جالبی را در سایت پدید آورد.



شکل ۱- نمایش از اجرای برنامه ی نمایش دهنده ی آمار سایت.



نکته :

اگر فایل global.asax تغییر کند ، برنامه ی وب Restart خواهد شد و تمام مقادیر ذخیره شده ی آن صفر خواهند شد.

اضافه کردن اشیاء به فایل Global.asax :

برای این منظور کلاسی ساده را اضافه می کنیم که رشته ها را دریافت و ذخیره کند. از شیء Application برای اینکار می خواهیم استفاده نماییم ، بنابراین کلاس باید Thread-safe باشد. معنای Thread-Safe این است که تعداد زیادی از کلاینت ها می توانند به کلاس بدون تخریب داده های آن دسترسی داشته باشند. از آنجائیکه ASP.NET یک Thread را به ازای هر صفحه بکار می برد ، اطمینان حاصل کردن از Thread-safe بودن کلاس اگر چندین کاربر در یک زمان به سایت مراجعه نمایند بسیار مهم و بحرانی است. برای اطمینان حاصل کردن از این امر از روش Synchronization (همزمانی) استفاده می شود که مربوط است به کلاس Hashtable .

مثال ۲:

یک پروژه ی وب اپلیکیشن جدید را آغاز نمایید. سپس از منوی پروژه یک کلاس جدید به نام MyClass را به برنامه اضافه نموده و کد زیر را در آن بنویسید.

```
using System;
using System.Collections; // for Hashtable
namespace ex02
{
    /// <summary>
    /// Summary description for MyClass.
    /// </summary>
    public class MyClass
    {
        private Hashtable m_col;
```



```
//m_colSync will be a thread-safe container for m_col
private Hashtable m_colSync;

public MyClass()
{
    m_col = new Hashtable();
    m_colSync = Hashtable.Synchronized(m_col);
}

public void AddItem(String Name, String Value)
{
    m_colSync[Name] = Value;
}

public String GetItem(String Name)
{
    return (String) m_colSync[Name];
}
}
```

سپس کد زیر را به فایل global.asax اضافه نمایید:

```
protected void Application_Start(Object sender, EventArgs e)
{
    MyClass m_var = new MyClass();
    m_var.AddItem("FirstUser", "Vahid");
    Application["myVar"] = m_var;
}
```

حوضه ی دید یک شیء در این فایل می تواند Application ، Session ، و یا AppInstance باشد. در حالتی که حوضه ی دید AppInstance باشد فقط به یک نمونه از HttpApplication مربوط شده و به اشتراک گذاشته نمی شود. متغیر های تعریف شده توسط شیء Session فقط برای یک کاربر در طول سشن او معتبر هستند و متغیر های تعریف شده توسط شیء Application در دسترس تمام کاربران سایت می باشد.

صفحه ی وب زیر از این شیء استفاده می کند:

```
private void Page_Load(object sender, System.EventArgs e)
{
    MyClass m_var = new MyClass();
    m_var = (MyClass) Application["myVar"];
    Response.Write(m_var.GetItem("FirstUser"));
}
```



مثال ۳:

بهترین مکان برای نوشتن شدن یک سری از رخدادهای برنامه در Event log سیستم (شخصی) است و یا سروری که دسترسی به یک سری از منابع آن برای شما تعریف شده است. برای این منظور یک پروژه ی وب اپلیکیشن جدید را آغاز کرده و سپس به کد زیر دقت فرمایید:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Web;
using System.Web.SessionState;

using System.Diagnostics;

namespace ex03
{
    /// <summary>
    /// Summary description for Global.
    /// </summary>
    public class Global : System.Web.HttpApplication
    {
        public Global()
        {
            InitializeComponent();
        }

        protected void Application_Start(Object sender, EventArgs e)
        {
            EventLog.WriteEntry("Sample Application",
                "Application Started!", EventLogEntryType.Information);
        }

        protected void Session_Start(Object sender, EventArgs e)
        {
            Session.Contents.Add("TimeStart", DateTime.Now);
        }

        protected void Application_Error(Object sender, EventArgs e)
        {
            EventLog.WriteEntry("Sample Application",
                "Application Error Occured!", EventLogEntryType.Error);
        }

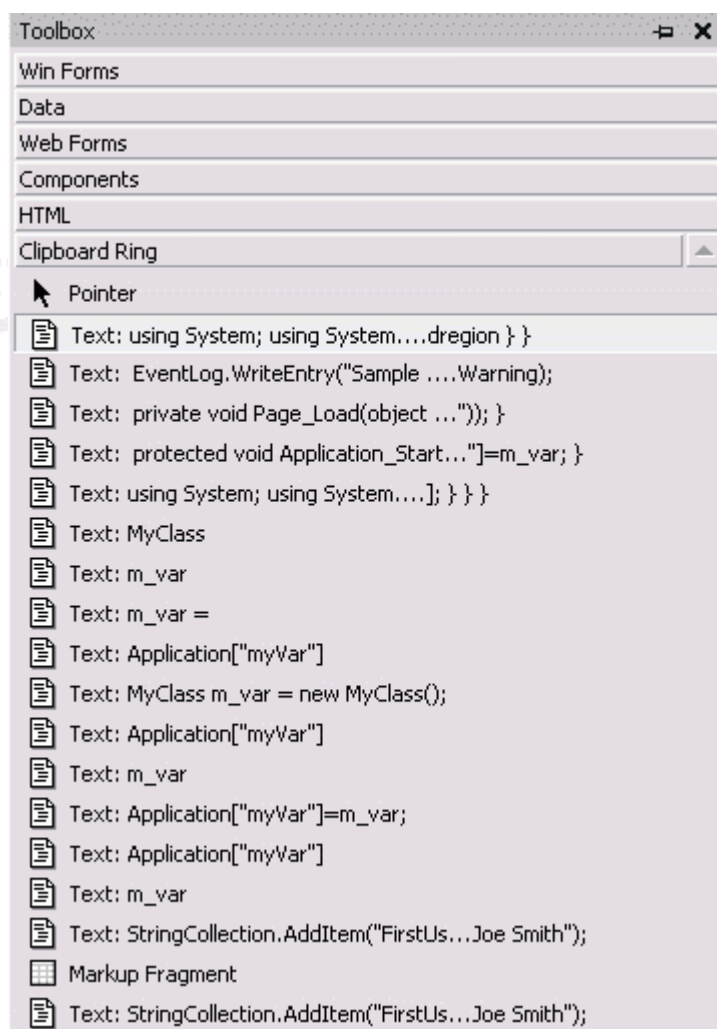
        protected void Application_End(Object sender, EventArgs e)
        {
            EventLog.WriteEntry("Sample Application",
                "Application Ended!", EventLogEntryType.Warning);
        }

        #region Web Form Designer generated code
```

```

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
    }
    #endregion
}

```



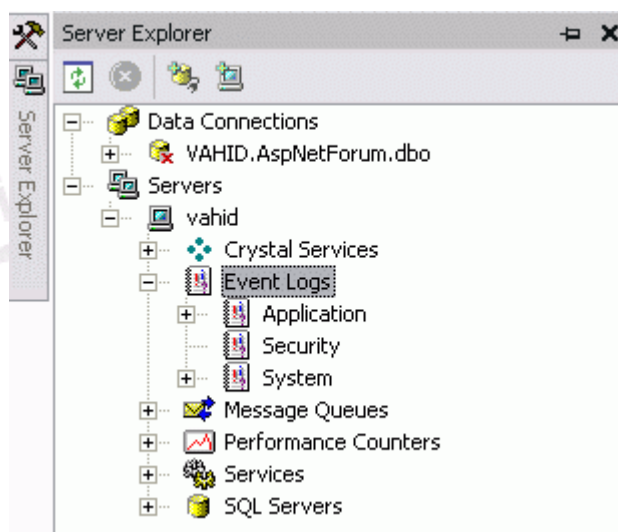
شکل ۲- مواردی را که تاکنون در برنامه و در Global.asax بکار بردیم ، می توان در Clipboard Ring جعبه ابزار استاندارد VS.NET نیز مشاهده نمود.

ممکن است بار اول در هنگام اجرای کد فوق با خطای زیر مواجه شوید که به تنظیمات امنیتی سیستم مربوط می گردد:

Security Exception

Description: The application attempted to perform an operation not allowed by the security policy. To grant this application the required permission please contact your system administrator or change the application's trust level in the configuration file.

Exception Details: System.Security.SecurityException: Requested registry access is not allowed.



شکل ۳- برای مشاهده ی رخدادهای ثبت شده می توان به Server explorer مراجعه نمود و قسمت Event logs را بررسی کرد.

اضافه کردن رخدادهای سفارشی به **global.asax** :

علاوه بر رخدادهای تعریف شده و استانداردهای فوق ، ما نیز می توانیم رخدادهایی را برای فایل **global.asax** تعریف کنیم. اینکار از طریق **HttpModule** به صورت زیر انجام می شود:



پروژه ی وب اپلیکیشن جدیدی را آغاز نموده و سپس یک کلاس جدید به آن به نام **SyncModule.cs** اضافه نمایید. محتویات این کلاس به صورت زیر است:

```
using System;
using System.Web;

namespace ex04
{
    public class SyncModule : IHttpModule
    {
        public delegate void MyEventHandler(Object s, EventArgs e);

        private MyEventHandler _eventHandler = null;

        public SyncModule()
        {
        }

        public void Init(HttpApplication app)
        {
            app.BeginRequest += new EventHandler(OnBeginRequest);
        }

        public void Dispose()
        {
        }

        public event MyEventHandler MyEvent
        {
            add { _eventHandler += value; }
            remove { _eventHandler -= value; }
        }

        public void OnBeginRequest(Object s, EventArgs e)
        {
            HttpApplication app = s as HttpApplication;
            app.Context.Response.Write(
                "Hello from OnBeginRequest in custom module.<br>");
            if(_eventHandler!=null)
                _eventHandler(this, null);
        }
    }
}
```

در ادامه کد زیر را در رخداده **Page_Load** مربوط به وب فرم بنویسید:



```
private void Page_Load(object sender, EventArgs e)
{
    Response.Write("Hello from Test.aspx.<br>");
}
```

سپس فایل وب کانفیگ را در برنامه گشوده و کد زیر را به آن اضافه نمایید:

```
<httpModules>
< "add name="MyModule" type="ex04.SyncModule, ex04"/>
</httpModules>
```

فایل global.asax را گشوده و کد زیر را در آن بنویسید:

```
protected void MyModule_OnMyEvent(Object src, EventArgs e)
{
    Context.Response.Write(
        "Hello from MyModule_OnMyEvent called in Global.asax.<br>");
}
```

برنامه را اجرا نمایید.

استفاده از Context در Global.asax :

به صورت پیش فرض اشیاء Response ، Request و Page تعریف شده در کلاس HttpApplication در فایل global.asax قابل دستیابی نیستند. از Context برای این منظور استفاده می شود.

مثال ۵:

یک پروژه ی وب اپلیکیشن جدید را آغاز نمایید و سپس کد زیر را در فایل Global.asax آن بنویسید:

```
protected void Application_Start(Object sender, EventArgs e)
{
    Application["SessionCounter"]=0;
```



```
}  
  
protected void Session_Start(Object sender, EventArgs e)  
{  
    /// Fires when the session is started  
    Application.Lock();  
    Application["SessionCounter"] = (int)Application["SessionCounter"]+1;  
    Application.UnLock();  
    Context.Response.Write("No of session(s): "+Application["SessionCounter"]);  
}
```





تمرین:

- ۱- یک Hit Counter را با دیتابیس اکسس طراحی نمایید.
- ۲- بر مبنای اطلاعات از ۵ دقیقه پیش تا کنون یوزر کنترلی را طراحی کنید که کاربران وارد شده به سایت را نمایش دهد.

