Introducing Asp.Net

Ing. Gabriele Zannoni
gabriele.zannoni@unibo.it

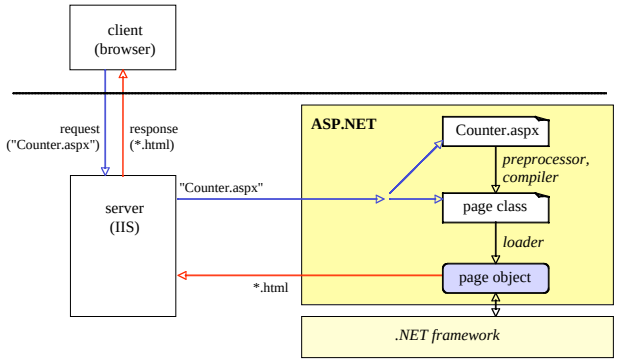
Cos'è?

- La risposta Microsoft alle esigenze del server side (leggi JSP)
- L'evoluzione (rivoluzione) di ASP
- La tecnologia per la scrittura di pagine dinamiche utilizzando il .Net Framework

Come funziona?

- Si appoggia su IIS (Internet Information Server) ◊ Web Server di casa Microsoft
- E' un'estensione ad IIS: tutte le richieste di pagine con una particolare estensione vengono passate al "filtro" ISAPI di Asp.Net
- Tale filtro è in grado di compilare le pagine Asp.Net in assembly e di mandarli in esecuzione
- Asp.Net è sostanzialmente un framework estendibile
  - Una pagina Asp.Net è elaborata da un HttpModule particolare: è possibile scrivere HttpModule personalizzati (leggi Servlet)

come funziona



counter.aspx

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
    <head> <title>Page counter</title> </head>
    <body>
        <h1>Welcome</h1>
        You are visitor number <%
        FileStream s = new FileStream("c:\\Data\\Counter.dat",
                FileMode.OpenOrCreate);
        int n;
        try {
                BinaryReader r = new BinaryReader(s);
                n = r.ReadInt32();
        } catch { n = 0; }  // if the file is empty
        n++;
        s.Seek(0, SeekOrigin.Begin);
        BinaryWriter w = new BinaryWriter(s);
        w.Write(n); s.Close();
        Response.Write(n);
        %> !
    </body>
</html>
```


counter.aspx in script tags

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.IO" %>
<html>
    <head>
        <title>Page counter</title>
        <script Language="C#" Runat="Server">
            int CounterValue() {
                FileStream s = new FileStream("c:\\Data\\Counter.dat",
                        FileMode.OpenOrCreate);
                ...
                n = r.ReadInt32();
                n++;
                ...
                return n;
            }
        </script>
    </head>
    <body>
        <h1>Welcome</h1>
        You are visitor number <%=CounterValue()%> !
    </body>
</html>
```

counter.aspx in “Code behind”

Counter.aspx

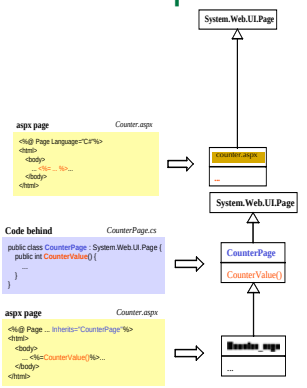
```
<%@ Page Language="C#" Inherits="CounterPage" Src="CounterPage.cs" %>
<html>
        <head> <title>Page counter</title> </head>
        <body>
                <h1>Welcome</h1>
                You are visitor number <%=CounterValue()%> !
        </body>
</html>
```

CounterPage.cs

```
using System.IO;

public class CounterPage : System.Web.UI.Page {
        public int CounterValue() {
                FileStream s = new FileStream("c:\\Data\\Counter.dat",
                                FileMode.OpenOrCreate);
                ...
                n = r.ReadInt32();
                n++;
                ...
                return n;
        }
}
```


Comunque...




```
namespace ASP {
 using System.IO;
  ...
  public class Counter_aspx : CounterPage {
    private static bool __initialized = false;
    private static ArrayList __fileDependencies;
    public Counter_aspx() {
      ArrayList dependencies;
      if ((__initialized == false)) { ... }
    }
    public override string TemplateSourceDirectory {
      get { return "/Samples"; }
    }
    private void __BuildControlTree(Control __ctrl) {
        ctrl.SetRenderMethodDelegate(new
RenderMethod(this.__Render__control1));
    }
    private void __Render__control1(HtmlTextWriter __output,
            Control parameterContainer) {
      __output.Write("\r\n<html>\r\n\t<head> <title>Page
            counter</title>
</head>\r\n\t<body>\r\n\t\t" +
            "<h1>Welcome</h1>\r\n\t\tYou are visitor number ");
      __output.Write(CounterValue());
      __output.Write(" !\r\n\t</body>\r\n</html>\r\n");
    }
    protected override void FrameworkInitialize() {
      __BuildControlTree(this);
      this.FileDependencies = __fileDependencies;
      this.EnableViewStateMac = true;
this.Request.ValidateInput();
    }
    ...
  }
}
```


Class Page

```
public class Page: TemplateControl {
  //--- properties
  public  ValidatorCollection Validators { get; }
  public bool IsValid { get; }
  public bool IsPostBack { get; }
  public virtual string TemplateSourceDirectory { get; }
  public HttpApplicationState Application { get; }
  public virtual HttpSessionState Session { get; }
  public HttpRequest Request { get; }
  public HttpResponse Response { get; }
  ...
  //--- methods
  public string MapPath(string virtualPath);
  public virtual void Validate();
  ...
}
```

MapPath(virtPath)
maps the virtual directory to the physical one

Validate()
starts all validators on the page

IsValid
true, if none of the validators on the page reported an error

IsPostBack
true, if the page was sent to the server in a round trip. If the page was requested for the first time IsPostBack == false

TemplateSourceDirectory
current virtual directory, e.g. "/Samples"

Application and **Session**
application state and session state

Request und **Response**
HTTP request and HTTP response

Class HttpRequest

```
public class HttpRequest {
  public string UserHostName { get; }
  public string UserHostAddress { get; }
  public string HttpMethod { get; }
  public HttpBrowserCapabilities Browser { get; }
  public NameValueCollection Form { get; }
  public NameValueCollection QueryString { get; }
  public NameValueCollection Cookies { get; }
  public NameValueCollection ServerVariables { get; }
  ...
}
```

UserHostName
domain name of the client

UserHostAddress
IP number of the client

```
<body>
  <%= "address = " + Request.UserHostAddress %><br>
  <%= "method = " + Request.HttpMethod %><br>
  <%= "browser = " + Request.Browser.Browser %><br>
  <%= "version = " + Request.Browser.Version %><br>
  <%= "supports JS = " + Request.Browser.JavaScript %><br>
  <%= "server = " + Request.ServerVariables["SERVER_SOFTWARE"] %>
</body>
```

```
address = 127.0.0.1
method = GET
browser = IE
version = 6.0
supports JS = True
server = Microsoft-IIS/5.0
```


Class HttpResponse

```
public class HttpResponse {
  //--- properties
  public string ContentType { get; set; }
  public TextWriter Output { get; }
  public int StatusCode { get; set; }
  public HttpCookieCollection Cookies { get; set; }
  ...
  //--- methods
  public void Write(string s); // various overloaded versions
  public void Redirect(string newURL);
  ...
}
```

ContentType
MIME type (e.g. text/html)

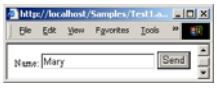
Output
HTML response stream; can be written to with Write

StatusCode
e.g. 200 for "ok" or
404 for "page not found"

Test1.aspx

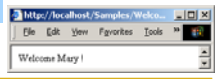
```
<form Runat="server">
  Name: <asp:TextBox ID="name" Runat="server" />
  <asp:Button Text="Send" OnClick="DoClick" Runat="server" />
</form>

void DoClick (object sender, EventArgs e) {
  Response.Redirect("Welcome.aspx?name=" + name.Text);
}
```

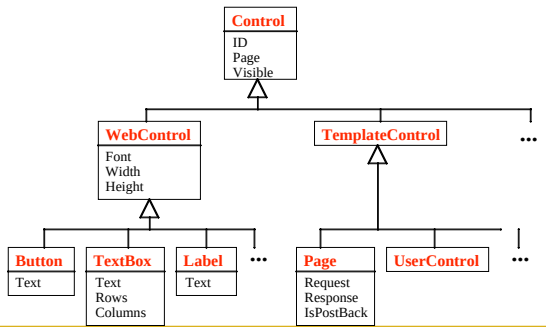


Welcome.aspx

```
Welcome <%= Request.QueryString["name"] %> !
```

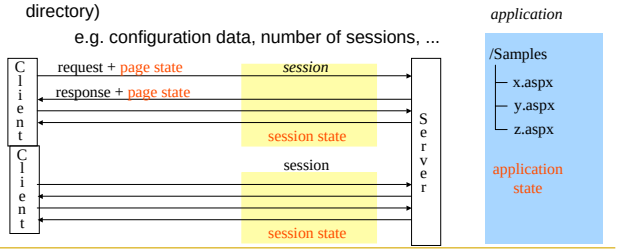


Controlli Visuali



3 Kinds of States

- Page state
  e.g. contents of TextBoxes, state of CheckBoxes, ...
- Session state (**session** = all requests from the same client within a certain time)
  e.g. shopping cart, email address of a client, ...
- Application state (**Application** = all aspx files in the same virtual directory)
  e.g. configuration data, number of sessions, ...



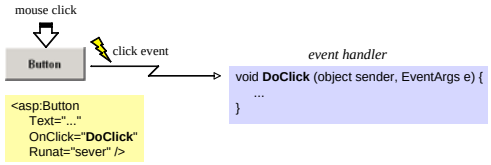
How to Access State Information

- Page state
  writing: ViewState["counter"] = counterVal;
  reading: int counterVal = (int) ViewState["counter"];
- Session state
  writing: Session["cart"] = shoppingCart;
  reading: DataTable shoppingCart = (DataTable) Session["cart"];
- Application state
  writing: Application["database"] = databaseName;
  reading: string databaseName = (string)Application["databaseName"];

ViewState, *Session* and *Application* are properties of the *Page* class

Eventi

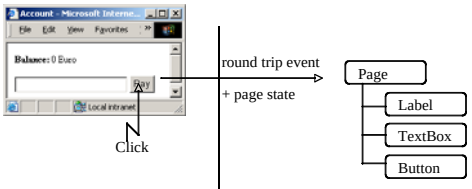
- Modello ad eventi per il web!



Eventi

Control	Event	When does the event occur?
all	**Init** **Load** **PreRender** **Unload**	¥ when the control is created ¥ after the data that were sent by the browser have been loaded into the control ¥ before HTML code for this control is generated ¥ before the control is removed from memory
Button	**Click**	when the button was clicked
TextBox	**TextChanged**	when the contents of the TextBox changed
CheckBox	**CheckedChanged**	when the state of the CheckBox changed
ListBox	**SelectedIndexChanged**	when a new item from the list has been selected

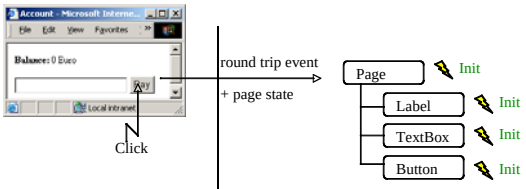
Round Trip



1. **Creation**
   create page object and its controls

Client *Server*

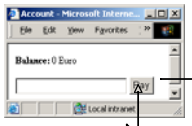
Round Trip



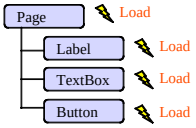
2. **Initialisation**
   - raise *Init* events

Client *Server*

Round Trip



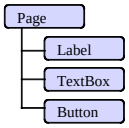
round trip event
+ page state



3. **Loading**
   - load controls with the values that the user has entered (page state)
   - raise *Load* events

Client *Server*

Round Trip



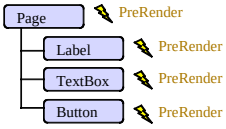
4. **Action**
   handle event(s)
   (Click, TextChanged, ...)

Client *Server*

Round Trip


```
<html>
   ...
   <input type="text" ...>
   <input type="button" ...>
   ...
</html>
```

HTML
+ page state

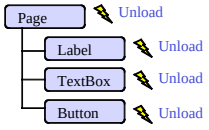


5. **Rendering**
   - raise *PreRender* events
   - call *Render* methods of all controls, which render the controls to HTML

Client *Server*

Round Trip


```
<html>
   ...
   <input type="text" ...>
   <input type="button" ...>
   ...
</html>
```



6. **Unloading**
   - raise *Unload* events for cleanup actions

Client *Server*

Peculiarità del modello

- La pagina è un oggetto ed è uno dei possibili handlers (?!)
- Gli elementi visuali sono oggetti
  - E' possibile implementare nuovi elementi visuali da zero o per composizione
- Il modello di sviluppo è unico ◊ eventi anche nel mondo web (web.forms) come nel mondo a "finestre" (windows.forms)
- Lo stato di ogni elemento nella pagina è persistente in modo automatico (a meno che non si richieda esplicitamente il contrario)
- Page Controller Pattern gratis… E il Front Controller?

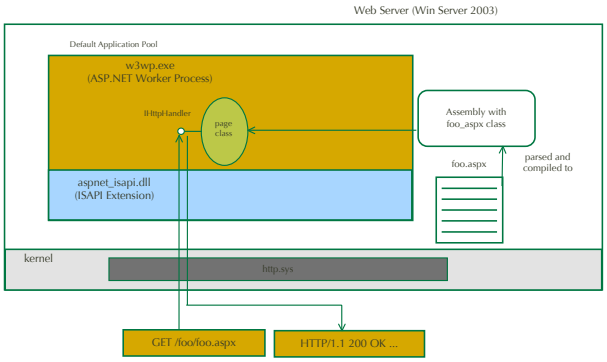
Debolezze del modello

- Nel framework non è disponibile alcuna agevolazione a lavorare con il pattern MVC (Front Controller)

Asp.Net inside - Pipeline

- Quando IIS riceve una richiesta HTTP
  - A seconda dell'estensione della richiesta (.htm, .php, .aspx, .asmx…) IIS decide a chi demandarne la gestione
  - aspx, asmx e altre sono le estensioni di default di Asp.Net ma sono cambiabili

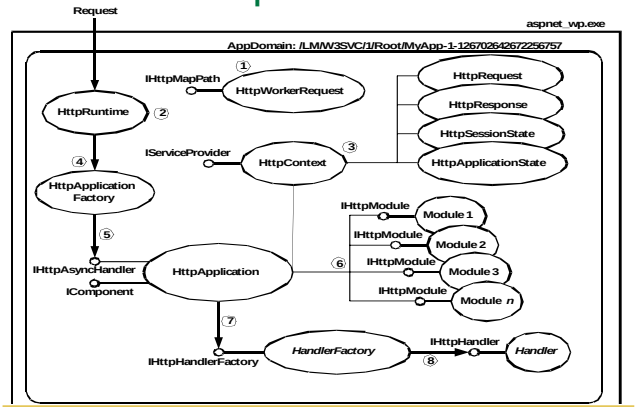
HttpPipeline – Win2003Server



Dentro la Pipeline

- Una volta dentro il worker process, una richiesta segue una serie di passi:
  - Inizialmente è diretta verso l'AppDomain associato con l'applicazione
  - Un insieme di classi all'interno dell'AppDomain interagiscono per servire la richiesta:
    - HttpRuntime
    - HttpWorkerRequest
    - HttpApplication
    - HttpContext
    - Modules
    - Handlers
    - …

Dentro la Pipeline



HttpContext

- `System.Web.HttpContext`
  - **`HttpContext`** contiene informazioni relativa a richiesta, risposta, stato di applicazione, stato di sessione
  - Esiste un **`HttpContext`** per ogni richiesta servita
  - L'**`HttpContext`** "giusto" è esplicitamente passato agli Http Handlers
  - L'**`HttpContext`** corrente è esposto tramite la proprietà statica **`HttpContext.Current (?!)`**

HttpContext

Name	Type	Description
`Current (static)`	`HttpContext`	`Context for request currently in progress`
`Application`	`HttpApplicationState`	`Application-wide property bag`
`ApplicationInstance`	`HttpApplication`	`Active application instance`
`Session`	`HttpSessionState`	`Per-client session state`
`Request`	`HttpRequest`	`HTTP Request object`
`Response`	`HttpResponse`	`HTTP Response object`
`User`	`IPrincipal`	`Security ID of caller`
`Handler`	`IHttpHandler`	`Handler for the request`
`Items`	`IDictionary`	`Per-request property bag`
`Server`	`HttpServerUtility`	`HTTP Server object`
`Error`	`Exception`	`Unhandled exception object`

Estensione della Pipeline

- La pipeline può essere estesa in tre posizioni tramite:
  - La definizione di una classe derivata da **`HttpApplication`** (usando global.asax)
  - La definizione di moduli custom creando classi che implementino **`IHttpModule`**
  - La deinizione di handlers custom creando classi che implementano **`IHttpHandler`**

Estensione: Applicazione

- L'applicazione è l'entry point per una richiesta
- Serve come repository di risorse "globali"
  - application state
  - cache
  - session state
- Fornisce accesso a eventi che avvengono durante il ciclo di vita dell'applicazione

Applicazione: Eventi

Event	Reason for firing	Order
`BeginRequest`	`New request received`	`1`
`AuthenticateRequest`	`Security identity of the user has been established`	`2`
`AuthorizeRequest`	`User authorization has been verified`	`3`
`ResolveRequestCache`	`After authorization but before invoking handler, used by caching modules to bypass execution of handlers if cache entry hits`	`4`
`AcquireRequestState`	`To load session state`	`5`
`PreRequestHandlerExecute`	`Before request sent to handler`	`6`
`PostRequestHandlerExecute`	`After request sent to handler`	`7`
`ReleaseRequestState`	`After all request handlers have completed, used by state modules to save state data`	`8`
`UpdateRequestCache`	`After handler execution, used by caching modules to store responses in cache`	`9`
`EndRequest`	`After request is processed`	`10`
`Disposed`	`Just prior to shutting down the application`	`-`
`Error`	`When an unhandled application error occurs`	`-`
`PreSendRequestContent`	`Before content sent to client`	`-`
`PreSendRequestHeaders`	`Before HTTP headers sent to client`	`-`

Applicazione: Eventi

Event	Reason for firing
`Application_Start`	`Application first starting`
`Application_End`	`Application ending`
`Session_Start`	`User session begins`
`Session_End`	`User session ends`

HttpHandlers

- Asp.Net usa gli oggetti handler per delegare le richieste HTTP a codice "utente"
  - Gli URI possono essere legati a classi che implementano l'interfaccia **`IHttpHandler`**
  - Gli URI paths non legati a classi vengono "compilati" per produrre handlers (se possibile, come accade per i file aspx)
  - Gli handlers devono essere elencati nella sezione **`httpHandlers`** del file di configurazione (web.config)

HttpHandlers: config

```
<configuration>
  <system.web>
    <httpHandlers>
      <add verb="*" path="*.foo"
           type="DM.AspDotNet.FooHandler, FooHandler" />
    </httpHandlers>
  </system.web>
</configuration>
```

HttpHandlers: definizione

```
namespace System.Web
{
  public interface IHttpHandler
  {
    void ProcessRequest(HttpContext ctx);
    bool IsReusable {get;}
  }

  public interface IHttpHandlerFactory
  {
    IHttpHandler GetHandler(HttpContext context,
      string requestType,
      string url,
      string pathTranslated);
    void ReleaseHandler(IHttpHandler handler);
  }
}
```


HttpHandlers: implementazione

```
using System;
using System.Web;

namespace DM.AspDotNet
{
  public class FooHandler : IHttpHandler
  {
    public void ProcessRequest(HttpContext context)
    {
      context.Response.Write("My cust handler!");
    }

    public bool IsReusable
    {
      get { return true; }
    }
  }
}
```


HttpModules

- Gli HttpModules forniscono la possibilità di effettuare pre/post processing su ogni richiesta
  - Esistono a livello di applicazione (non di richiesta)
  - Implementano l'interfaccia IHttpModule
  - Il metodo **`Init`** viene invocato quando l'applicazione parte permettendo al modulo di registrarsi agli eventi di applicazione che gli interessano
  - Il sistema fornisce già una serie di moduli (**`SessionStateModule`**,, **`UrlAuthorizationModule`**, …)

```
public interface IHttpModule
{
  void Dispose();
  void Init(HttpApplication context);
}
```


HttpModules: config

- Anche gli HttpModules al pari degli handlers devono essere "registrati" nel file di configurazione:

```
<httpModules>
 <add name="OutputCache"
  type="System.Web.Caching.OutputCacheModule, ..." />
 <add name="Session"
  type="System.Web.SessionState.SessionStateModule..." />
 <add name="WindowsAuthentication"
  type="System.Web.Security.WindowsAuthenticationModule.."/>
 <add name="FormsAuthentication"
  type="System.Web.Security.FormsAuthenticationModule..." />
 <add name="PassportAuthentication"
 type="System.Web.Security.PassportAuthenticationModule.."/>
 <add name="_rlAuthorization"
  type="System.Web.Security.UrlAuthorizationModule..." />
 <add name="FileAuthorization"
  type="System.Web.Security.FileAuthorizationModule..." />
</httpModules>
```


And so on…

- Cosa manca?
  - Caching
  - Gestione della sicurezza
  - Databinding
  - Master Pages, Themes, Skins
  - Ajax.Net
  - …

Webography

- www.asp.net
- www.msdn.com + MSDN Library
- www.google.com