Tecnologie di Sviluppo per il Web

Applicazioni Web J2EE: ASP.NET

versione 3.0

G. Mecca – mecca@unibas.it – Università della Basilicata

Sommario

- Applicazioni ASP.NET
- Modello 1 con ASP.NET
- Web Forms
- Operazioni di Costruzione

Applicazioni ASP.NET

- ASP.NET
  - tecnologia Microsoft basata sul .NET Framework per lo sviluppo di applicazioni Web
  - il namespace principale è System.Web
  - discende da ASP (3.0), di cui è una significativa evoluzione
  - attualmente alla versione 1.1, in preparazione la versione 2.0

Applicazioni ASP.NET

- Architettura applicativa per ASP.NET
  - macchina Windows
  - .NET Framework SDK
  - server applicativo Internet Information Services
- In linea teorica
  - per lo sviluppo è adatto ogni tipo di browser
  - in concreto, alcune funzionalità sono accessibili solo usando Internet Explorer

Applicazioni ASP.NET

- Esempio: SmartNavigation
  - ⇨una funzionalità innovativa di ASP.NET che consente di evitare di rigenerare integralmente il contenuto della finestra del browser ad ogni transazione
  - ⇨aggiorna selettivamente le porzioni aggiornate
  - ⇨funziona solo con Internet Explorer 5.5 o superiori

Applicazioni ASP.NET

- Applicazione Web ASP.NET
  - ⇨cartella visibile sul file system virtuale di Internet Information Services
  - ⇨la cartella deve essere configurata come applicazione ASP.NET usando l'applet di configurazione di IIS
- Operazione di deployment
  - ⇨copia dei file dalla cartella di progetto alla cartella scelta

Applicazioni ASP.NET

- Struttura della cartella
  - ⇨collezione di pagine ASP.NET (con estensione .aspx)
  - ⇨collezione di assembly nella cartella /bin (che rappresenta la GAC dell'applicazione Web)
  - ⇨file web.config nella cartella radice che contiene i parametri di configurazione per l'applicazione

Applicazioni ASP.NET

- Pagina ASP.NET
  - ⇨pagina in cui è possibile mischiare codice HTML e codice dei linguaggi .NET
  - ⇨viene compilata in una classe di tipo System.Web.UI.Page
  - ⇨il ciclo di vita della pagina è gestito dal server applicativo, che la compila, la istanzia e la esegue per gestire le richieste all'URI corrispondente

Applicazioni ASP.NET

- Metodi di callback per il ciclo di vita
  - ⇨void Page_Init(object sender,EventArgs e) corrisponde all'evento di inizializzazione
  - ⇨void Page_Dispose(object sender,EventArgs e) corrisponde all'evento di distruzione della pagina
  - ⇨il server può decidere di inizializzare una singola istanza oppure (normalmente) un pool di istanze per ciascuna pagina

Applicazioni ASP.NET

- Sintassi delle pagine ASP.NET
  - ⇨codice HTML e commenti <!-- -->
  - ⇨blocchi di codice tra <% %>
  - ⇨espressioni tra <%= %>
  - ⇨dichiarazioni tra <script></script>
  - ⇨direttive di pagina tra <%@ %>
- Nota
  - ⇨non esiste l'equivalente dell'expression language

Applicazioni ASP.NET

- Attenzione
  - ⇨il linguaggio predefinito è VB.NET
  - ⇨per usare C# è necessario usare la direttiva <%@ Page Language="C#" %>
- Per importare un namespace

  <%@ Import Namespace="Unibas.Indovina" %>
- Per includere l'output di un'altra pagina

  <!-- #include file="intestazione.aspx" -->

Applicazioni ASP.NET

- Il file di configurazione dell'applicazione
  - ⇨web.config nella cartella radice
  - ⇨file xml di configurazione dell'applicazione Web
- Funzionalità principali
  - ⇨abilitare il tracing/logging nell'esecuzione delle pagine
  - ⇨stabilire il livello di debugging

Applicazioni ASP.NET

- Gestione delle sessioni
  - ⇨normalmente le sessioni sono gestite utilizzando i cookie
  - ⇨è possibile chiedere la riscrittura degli URI utilizzando web.config e specificando che le sessioni devono essere “cookieless”
  - ⇨in questo caso gli URI vengono sempre riscritti

Applicazioni ASP.NET

- Un esempio di file di configurazione

```
<configuration>
   <system.web>
      <compilation debug="true" />
      <trace enabled="false" requestLimit="10"
             localOnly="false" pageOutput="true" />
      <sessionState mode="InProc" cookieless="true" timeout="15" />
      <customErrors defaultRedirect="schermoErrore.aspx“
                    mode="RemoteOnly" />
   </system.web>
</configuration>
```

Modello 1 con ASP.NET

>> indovinaM1
>> aciM1

- Architettura modello 1
  - ⇨interfaccia e controllo sono contenute nelle pagine .aspx
  - ⇨modello e persistenza negli assemby in /bin
- Esempi
  - ⇨IndovinaM1
  - ⇨AciM1

Modello 1 con ASP.NET

- Oggetti predefiniti visibili nella pagina
  - ⇨proprietà pubbliche ereditate
  - ⇨System.Web.HttpRequest Request
  - ⇨System.Web.HttpResponse Response
  - ⇨System.Web.HttpSessionState Session
  - ⇨System.Web.HtpApplicationState Application
  - ⇨System.Web.HttpServerUtilities Server
  - ⇨System.Web.HttpContext Context
  - ⇨System.Web.TraceContext Trace

Modello 1 con ASP.NET

- I membri principali di HttpRequest
  - diverse mappe
  - Request.Form: mappa dei parametri provenienti da richieste post
  - Request.QueryString: mappa dei parametri provenienti da richieste get
  - Request.Cookie: mappa dei cookie
  - Request.Headers: mappa delle intestazioni
  - Request.ServerVariables

Modello 1 con ASP.NET

- L'indicizzatore di HttpRequest
  - consente di accedere in modo unificato a Form, QueryString, Cookie, ServerVariables
- Esempio
  - Request["nome"] equivale a Request.Form["nome"] se nome proviene dalla sottomissione di una form di tipo post

Modello 1 con ASP.NET

- I membri principali di HttpResponse
  - Response.Write()
  - Response.Redirect(URI)
  - Response.AppendHeader(nome, valore)
  - Response.ClearHeaders()
  - Response.Clear()
  - proprietà pubblica Response.OutputCache: imposta Cache-Control

Modello 1 con ASP.NET

- I membri principali di HttpSessionState
  - indicizzatore per gestire la mappa dei parametri, o, in alternativa, metodo Add(nome, valore)
  - metodo Session.Abandon() per invalidare la sessione
  - proprietà TimeOut
  - proprietà Session.IsNewSession

Modello 1 con ASP.NET

- I membri principali di HttpApplicationState
  - ⇨indicizzatore per gestire la mappa dei parametri, o, in alternativa, metodo Add(nome, valore)
- I membri principali di HttpServerUtility
  - ⇨metodo Server.Transfer(URI)
- I membri principali di HttpContext
  - ⇨mappa di parametri che surroga la mappa della richiesta

Modello 1 con ASP.NET

>> indovinaM1
>> aciM1

- I membri principali di TraceContext
  - ⇨metodo Trace.Write()
  - ⇨consente di usufruire del sistema di logging a livello di pagina e di applicazione che viene mostrato dal browser
  - ⇨IE mostra le informazioni di trace in fondo a ciascuna pagina e nel file trace.axd
  - ⇨per abilitarlo: <@% Page Trace="true %>
  - ⇨oppure web.config

Modello 1 con ASP.NET

- Nota
  - ⇨esistono (pochi) framework per lo sviluppo ASP.NET con il modello 2
  - ⇨ma sono relativamente poco utilizzati
- Esempio
  - ⇨maverick.net

Web Forms

- Infatti
  - ⇨ASP.NET nasce come tecnologia per lo sviluppo Web orientato ai componenti e agli eventi
- Obiettivi
  - ⇨sviluppare una tecnologia “Web forms” del tutto analoga a Windows Forms
  - ⇨consentire lo sviluppo di ambienti RAD anche per le applicazioni Web

Web Forms

>> WebMatrix

- L'esempio per eccellenza
  - ⇨Visual Studio.NET
- Un altro esempio
  - ⇨Microsoft ASP.NET Web Matrix
  - ⇨prodotto distribuito gratuitamente su gotdotnet.com
  - ⇨un ambiente RAD per lo sviluppo di applicazioni Web Forms

Web Forms

- Caratteristiche di Web Forms
  - ⇨fornisce un insieme ricco di controlli lato server per la creazione applicazioni Web
  - ⇨i controlli sono implementati nel namespace System.Web.UI.WebControls
  - ⇨es: Button, TextBox, Label, DataGrid, Calendar, ...
  - ⇨a ciascun controllo corrisponde un tag di tipo <asp:...> utilizzabile nelle pagine aspx

Web Forms

- Caratteristiche i Web Forms (continua)
  - ⇨a ciascun controllo è possibile associare gestori di eventi
  - ⇨le richieste eseguono tipicamente postback per la gestione degli eventi
  - ⇨la richiesta ha un ciclo di vita complesso gestito da Internet Information Services

Web Forms

- Caratteristiche di Web Forms (continua)
  - ⇨esistono vari controlli di convalida
  - ⇨es: asp:RequiredFieldValidator, asp:CompareValidator ecc.
  - ⇨il framework consente di effettuare convalide sia sul client (con JavaScript e DHTML) che sul server
  - ⇨il codice JavaScript prevede un meccanismo di “down level” basato sullo user agent

Web Forms

>> indovinaWebForms

- Caratteristiche di Web Forms (continua)
  - ⇨il framework consente di utilizzare diverse versioni dei controlli lato server
  - ⇨per ottenere rendering diversi
  - ⇨es: utilizzando i controlli del namespace System.Web.UI.MobileControls è possibile ottenere il rendering WML invece che HTML

Web Forms

- Controlli lato server
  - ⇨sono caratterizzati dal prefisso asp: e dall'attributo Runat="server"
  - ⇨devono essere contenuti in un elemento <form Runat="server"> (al più una form in ciascuna pagina)
  - ⇨ciascun controllo ha una serie di eventi predefiniti, a cui è possibile associare gestori

Web Forms

- Organizzazione del codice
  - ⇨basata sul concetto di “code behind”
- A ciascuna pagina ASP.NET
  - ⇨viene associata una classe che contiene il codice per la gestione degli eventi
  - ⇨il codice della classe e quello generato dalla pagina .aspx vengono uniti per generare la pagina finale

Web Forms

- Le regole del “code behind”
  - ⇨la classe di “code behind” deve estendere System.Web.UI.Page
  - ⇨deve contenere una proprietà (protetta) per ciascun controllo lato sever utilizzato nella pagina aspx
  - ⇨deve contenere i metodi per la gestione degli eventi
  - ⇨NOTA: la classe NON deve essere compilata

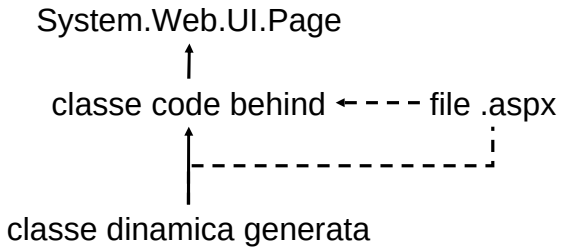
Web Forms

- Le regole del “code behind” (continua)
  - ⇨la pagina aspx deve contenere una direttiva di pagina che dichiara che estende la classe code behind, e un riferimento al file di codice sorgente
- Sintassi
  - ⇨<%@ Page Inherits=“*Namespace.Classe*” Src=“*file.cs*” %>

Web Forms

>> default.aspx
>> tentativo.aspx

- Le regole del “code behind” (continua)
  - ⇨il server applicativo compila la classe dinamica mettendo assieme il codice dei due file



Web Forms

- In sintesi
  - ⇨il codice della vista e del controllo è sostanzialmente ancora accoppiato
  - ⇨l'architettura resta un'architettura di base
  - ⇨ma organizzativamente è possibile evitare di scrivere assieme il codice C#/VB.NET e il codice HTML

Web Forms

- Lo stato della vista
  - ⇨viene mantenuto attraverso un campo di tipo hidden VIEWSTATE aggiunto a tutte le form lato server
  - ⇨consente di ripristinare l'albero di componenti all'arrivo delle richieste http
- Il ciclo di vita della richiesta
  - ⇨per le richieste di postback è analogo a quello di una richiesta Java Server Faces

Web Forms

- Ciclo di vita della richiesta
  - I fase: ripristino viewstate (e applicazione parametri query string)
  - II fase: convalida (>>)
  - II fase: esecuzione gestore evento Page_Load
  - III fase: esecuzione eventi
  - IV fase: esecuzione evento Page_Unload
  - V fase: invio della risposta

Web Forms

- Il meccanismo di esecuzione degli eventi
  - è possibile scegliere tra la modalità di caching ed esecuzione cumulativa (default)
  - oppure la modalità “autopostback”, specificando l’attributo AutoPostBack per i controlli i cui eventi devono essere gestiti immediatamente generando richieste al server

Web Forms

- Il meccanismo di convalida
  - è pesantemente basato su JavaScript
  - normalmente il codice JavaScript impedisce l'invio al server di form che non sono valide
  - questo a volte rende macchinoso l'uso dell'applicazione
  - è possibile disabilitare il meccanismo specificando nella direttiva page l'attributo ClientTarget="downlevel"

Web Forms

- Convalida downlevel
  - pensata per browser che non supportano versioni adeguate di JavaScript (in questo caso viene attivato automaticamente)
  - in questo caso la convalida viene effettuata sul server
  - ma attenzione: gli errori di convalida non impediscono l'esecuzione degli eventi

Web Forms

- Per verificare se la convalida è superata
  - è possibile utilizzare la proprietà predefinita IsValid sul riferimento Page
  - in questo modo è possibile evitare di effettuare operazioni in caso la convalida sia fallita

Web Forms

- Eventi dell'applicazione
  - il sistema utilizza un pool di oggetti di tipo System.Web.HttpApplication per la gestione della richiesta
  - è possibile gestire il ciclo di vita di questi oggetti (es: per operazioni di filtraggio)
  - per farlo è possibile utilizzare il file global.asax

Web Forms

- global.asax
  - ⇨contiene codice (es: C#) per la gestione degli eventi predefiniti dell'applicazione
- Esempi
  - ⇨Init, Dispose: creazione di un oggetto appl.
  - ⇨OnStart, OnEnd: creazione di una sessione
  - ⇨BeginRequest: inizio di una transazione

Operazioni di Costruzione

- Per le operazioni di costruzione
  - ⇨è possibile utilizzare Nant
- In effetti
  - ⇨Nant non ha task specificamente orientati allo sviluppo ASP.NET
  - ⇨Nant-contrib ne contiene alcuni, ma orientati allo sviluppo di web services

Operazioni di Costruzione

- Il file web-template.build
  - ⇨target install: costruisce l'applicazione e la copia in ${wwwroot}/${nant.project.name} – dipende da build
  - ⇨target deploy: ricostruisce l'applicazione e la copia in ${wwwroot}/${nant.project.name} – dipende da rebuild
  - ⇨prevedono entrambi l'utilizzo di dependent.config

Riassumendo

- Applicazioni ASP.NET
- Modello 1 con ASP.NET
- Web Forms
- Operazioni di Costruzione

Termini della Licenza