# Programming ASP.Net 2.0
# Introduzione

Pietro Brambati
Microsoft
pietro.brambati@microsoft.com

# Agenda (Mattina)

- Introduzione al .Net Framework
- Introduzione a ASP.NET 2.0
- Membership e controlli per il Log-in
- Le pagine Master, i Temi e gli Skin
- Personalizzazione
- Sorgenti dati e Controlli data-bound
- Gestione della Stato e Caching

Microsoft

# Agenda (Pomeriggio)

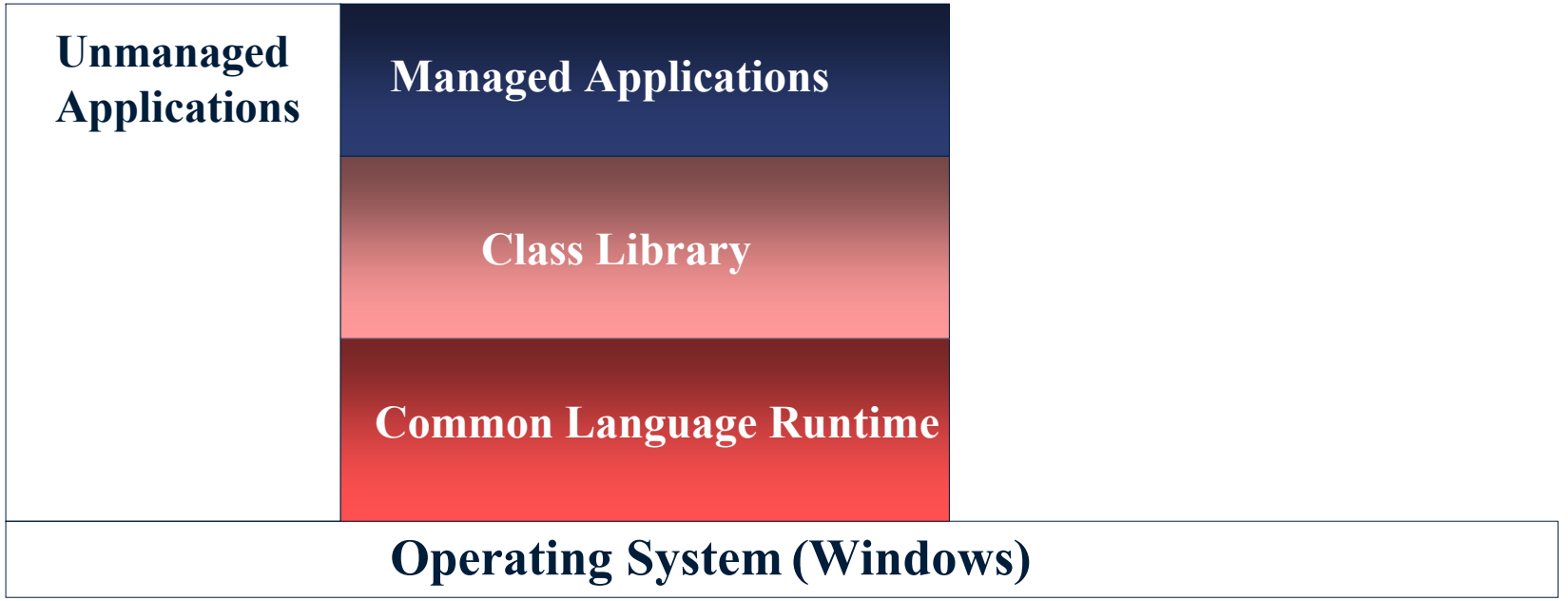
- Hands-on Lab

Microsoft

# Architettura del .Net Framework

Giusto un po’ per capire cosa sta alla base di tutto ...

# Cos'è il .Net Framework

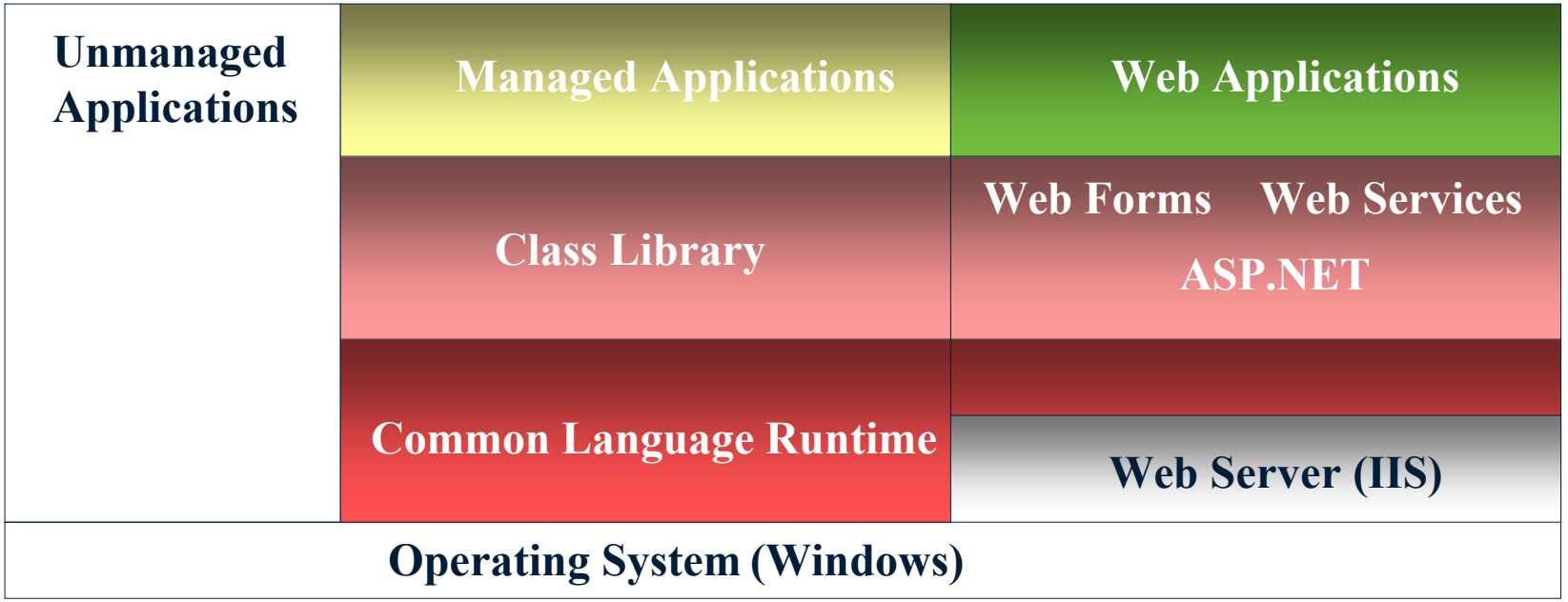



**Common Language Runtime** — **interoperability, security, garbage collection, versioning, ...**

**Class Library** — **GUI, collections, threads, networking, reflection, XML, ...**

Microsoft

# Cos'è il .Net Framework

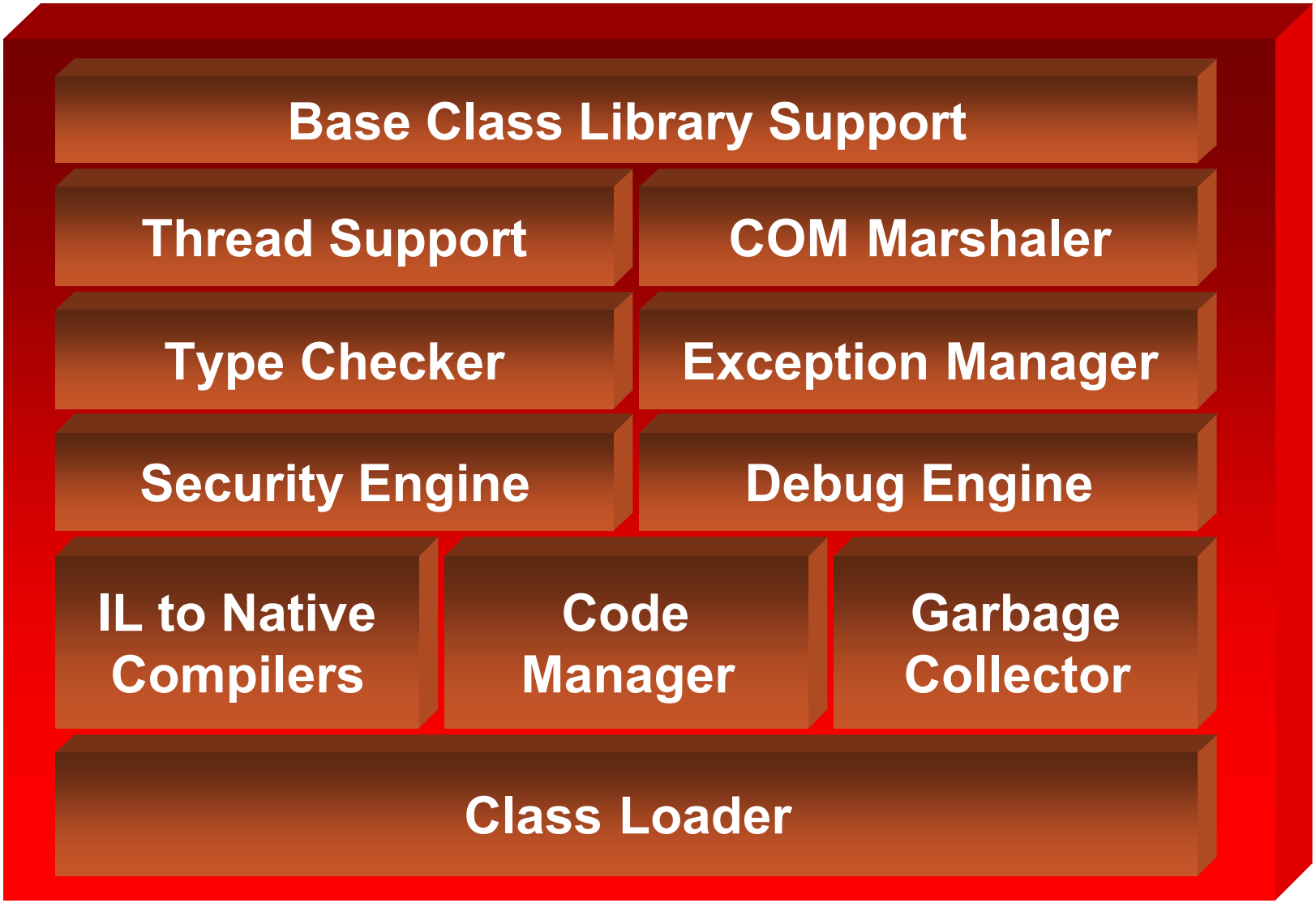



**ASP.NET, Web Forms** — Web GUI (object-oriented, event-based, browser-independent)

**Web Services** — distributed services over RPC (SOAP, HTTP)

Microsoft

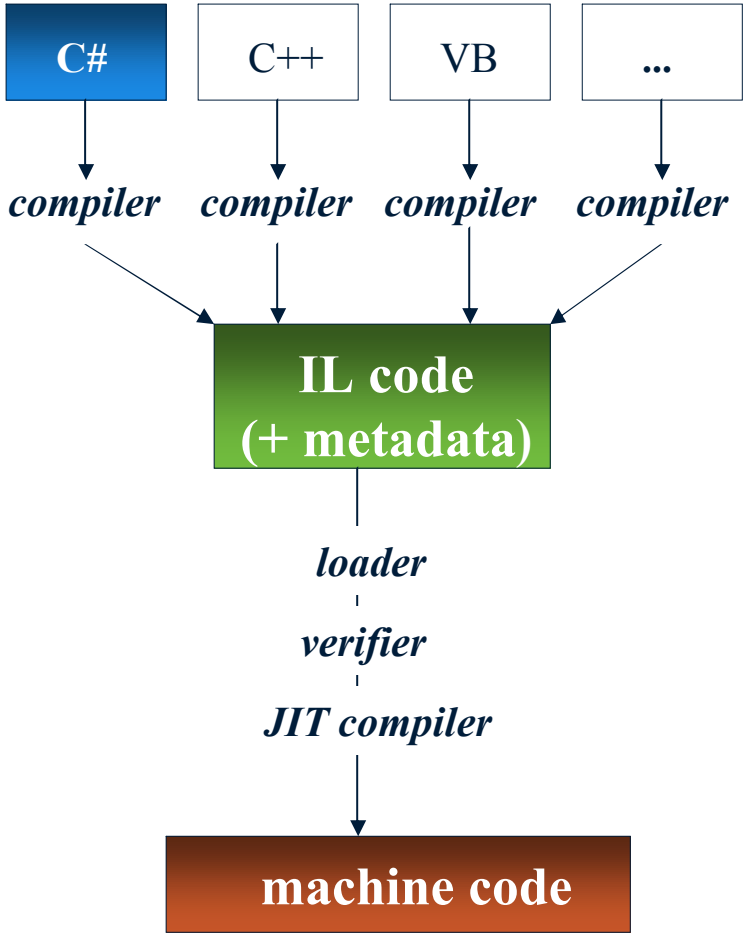
# Common Language Runtime

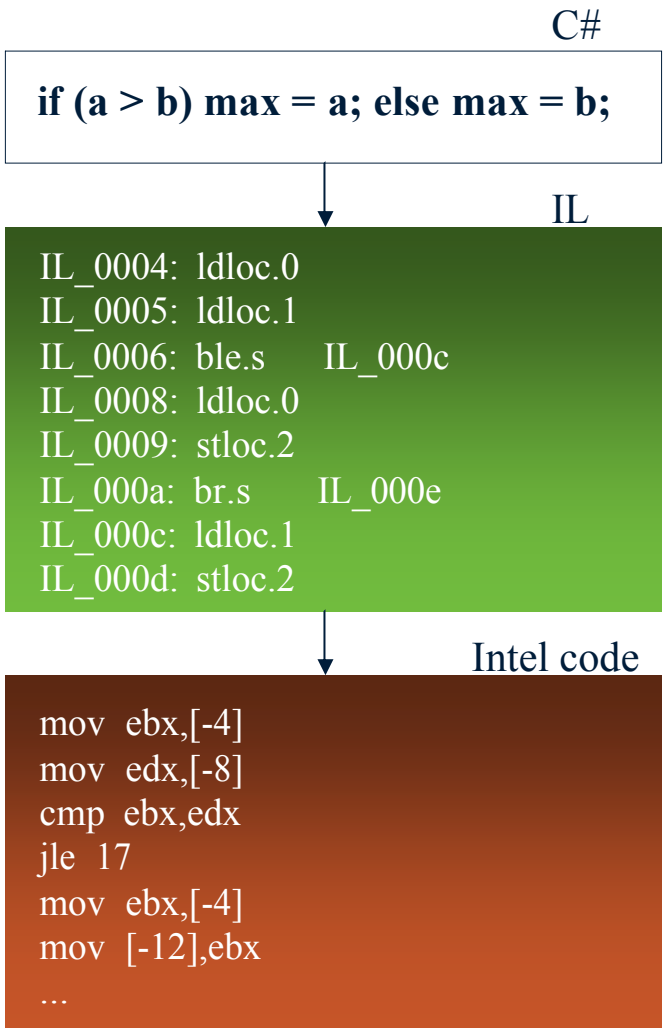



Microsoft

# Architettura

- Compatibilità binaria tra linguaggi




Microsoft

# Indipendenza dalla piattaforma e dal linguaggio

- .NET è un'implementazione di CLI
  - Common Language Infrastructure
- CLI è uno standard ECMA, definito con C#
  - ECMA-334, ECMA-335
- Esistono già altre implementazioni di CLI:
  - SSCLI (Microsoft per Windows, FreeBSD e Macintosh)
  - Mono (per Linux)
  - DotGNU
  - Intel OCL (Open CLI Library)
  - …

# Introduzione a Visual Studio 2005 e ASP.NET 2.0

## Esploriamo l'ambiente ...

# Visual Web Developer 2005 Express Edition


http://www.microsoft.com/italy/msdn/prodotti/vs2005/editions/download/wdd.mspx

- Download gratuito
- La registrazione offre molti vantaggi
  - “Hosting” gratuito del vostro sito

Microsoft

# Demo

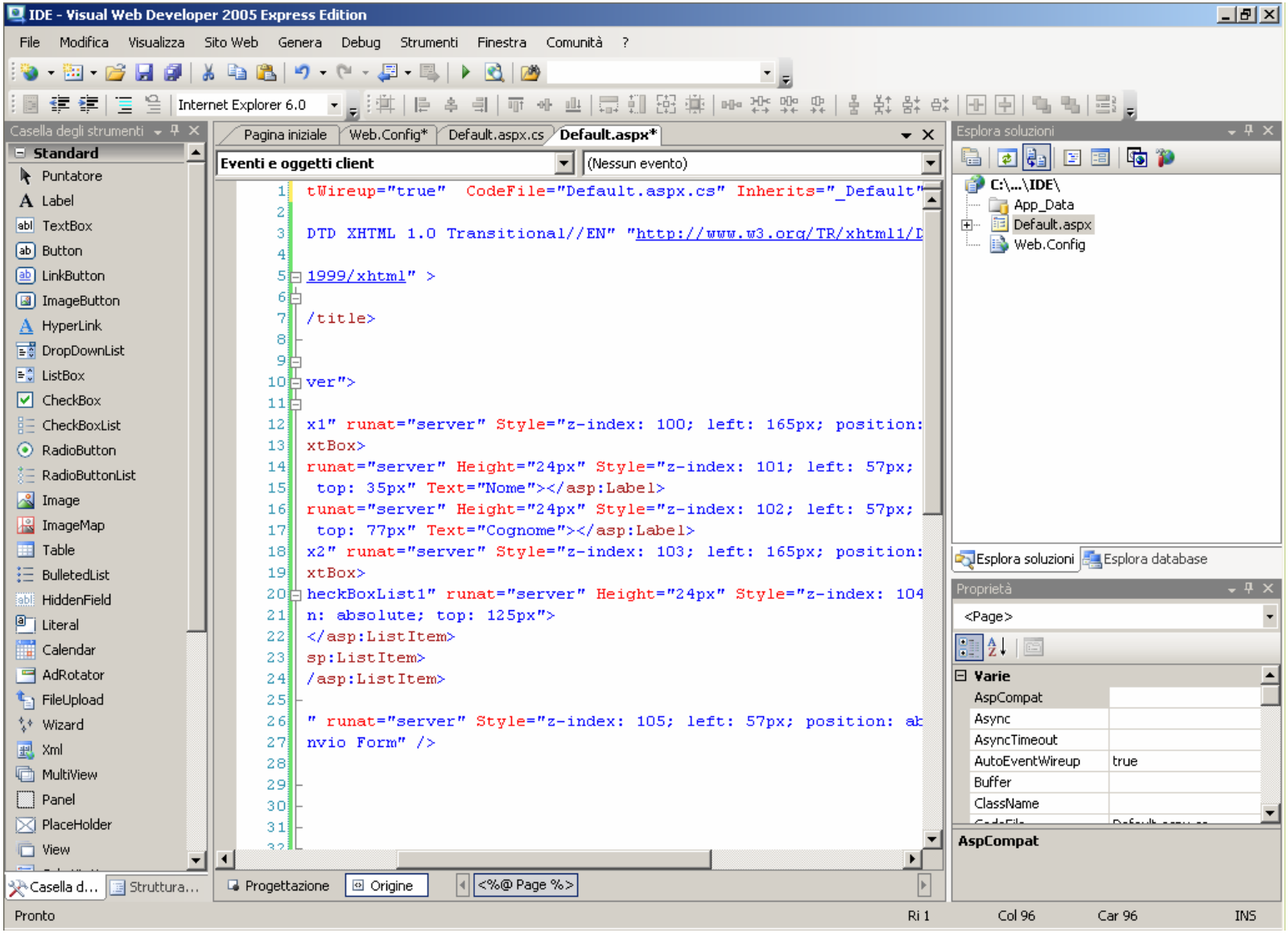
- Creazione di un sito web
  - Controlli server base
    - Lettura di un parametro da file di configurazione
- View State (usare un controllo calendar)
- Code-bihind
- Post-back

# Introduzione a ASP.NET 2.0

- Esploriamo l'IDE
  - Posizionamento assoluto dei controlli
- Creazione di un Web Site
- Controlli e funzionalità di base
  - Controlli server e output dipendente dal dispositivo
  - Gestione del ViewState
  - Validatori
  - Gestione del Post Back
- Diagnostica e Ciclo di Vita della pagina

Microsoft

# IDE

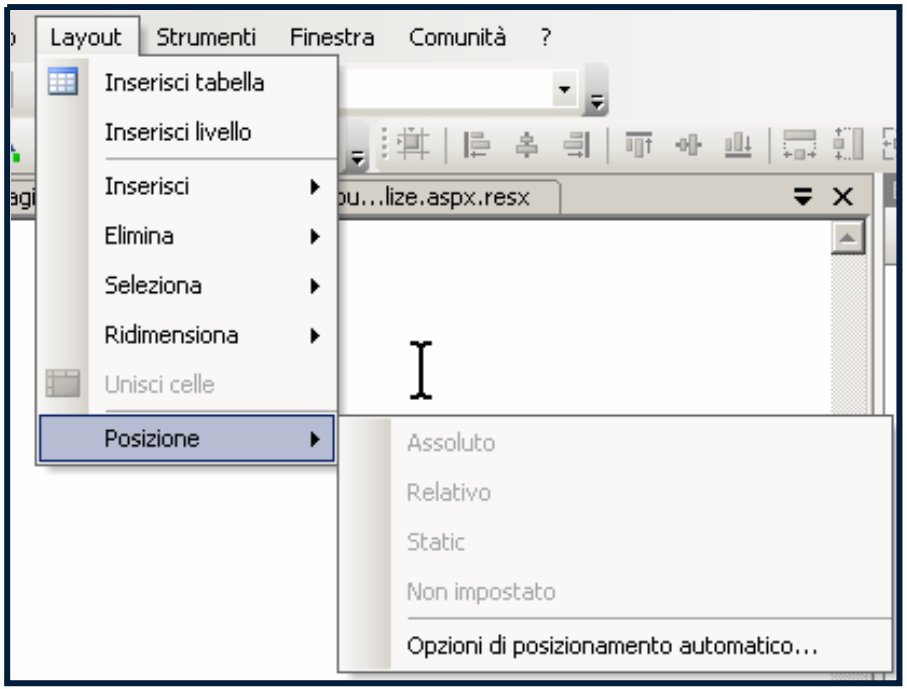



Microsoft

# Controlli

- Controlli HTML:
  - è l'HTML standard
    - <a >..., <label>...
- Lato-Server
  - Generano HTML in modo **dipendente** dal browser
  - Ce ne sono tantissimi !! Vediamone alcuni

# Posizionamento assoluto dei controlli




Microsoft

# ViewState

- Mantiene lo stato a livello di **Pagina**
  - È un dizionario nome/valore
- È un campo Hidden della pagina
- Può essere usato anche programmaticamente
  - ViewState.Add(“NomeUtente”, “Pietro”)
- Può essere disabilitato a livello di pagina
  - <%@ Page ... EnableViewState="false"%>
  - Attenzione che i controlli che usano il view state possono non funzionare più!

# PostBack

- È un evento che scatta la seconda volta che si arriva su una pagina
  - In seguito ad una POST HTTP che si verifica
    - Submit di un bottone
    - Controlli server-side possono avere la proprietà **AutoPostBack** abilitata
      - Può servire per popolare altri controlli o disabilitarli

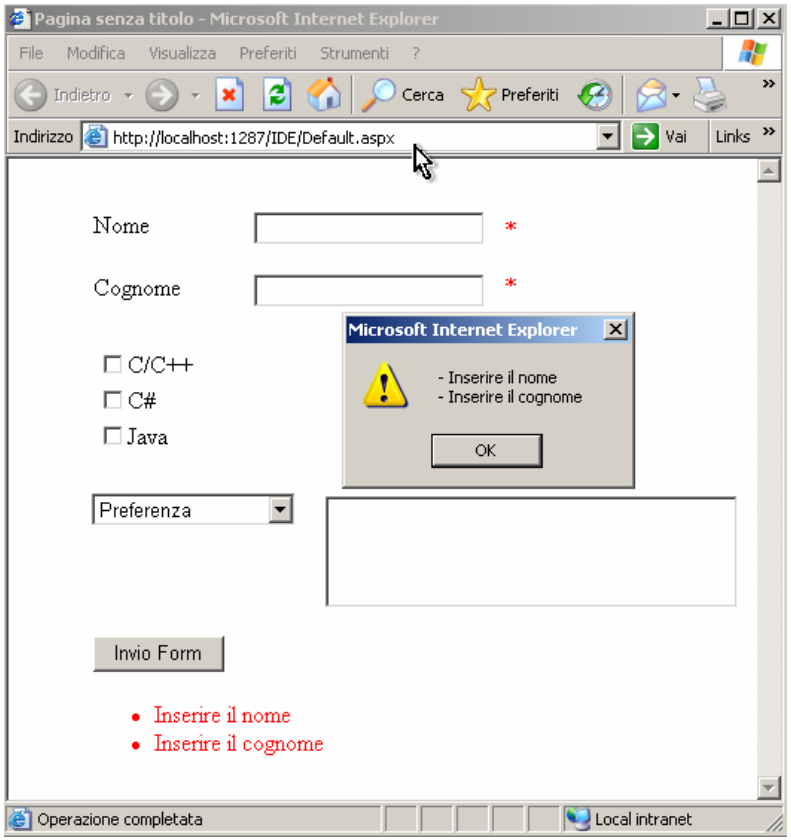
# PostBack: Uso tipico

- Posso ottimizzare il codice eseguito nella pagina
  - Accedo una sola volta alle risorse costose (database)

```
protected void Page_Load(..)
{
        if(Page.IsPostBack == false)
        {
                // E' il primo accesso alla pagina
                // Accesso al database
        }
}
```

# Validatori

- Controlli per la validazione dei controlli **lato server**
- Rilevano se il browser supporta la validazione **lato client**
  - RequiredFieldValidator
  - RangeValidator
  - RegularExpressionValidator
  - CompareValidator
  - CustomValidator
  - ValidatorSummary




Microsoft

# Demo

- Validatori

Microsoft

# Debugging (Server-side)

- Il debugging viene abilitato nel web.config

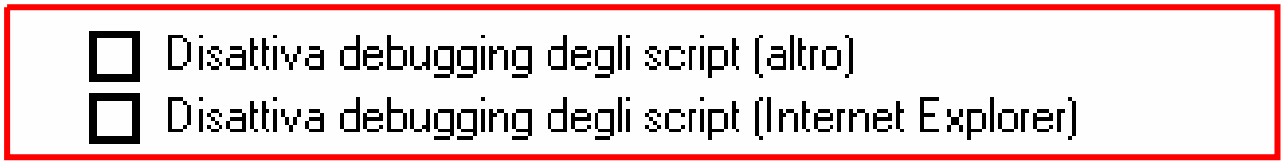
```
<compilation debug="true"/>
```

```
13      protected void Page_Load(object sender, EventArgs e)
14      {
15          if (Page.IsPostBack == false)
16          {
17              // E' la prima volta che accedo alla pagina
18              // Accesso a risorse costose (database)
19              Trace.Write("IsPostBack==false");
20
21              ViewState.Add("Mypro", "pietro");
22          }
23          else
24          {
25              string msg = "vuoto";
26              try
27              {
28                  msg = ViewState["Mypro"].ToString();
29              }
30              catch (Exception)
```

Microsoft

# Debugging (Javascript)

- Più complicato da impostare
  - Abilitare il browser
  - Quindi o si fa partire il debugger da IE e poi si mette il breakpoint sul javascript
  - o da Visual Studio ci si “attacca” al processo IE
    - Questa funzionalità non è supportata nella versione “Express”
  - “Trucco”: istruzione “debugger;” nel codice Javascrip
    - Attenzione a non lasciarla in produzione!




Microsoft

# Tracing

- Si può abilitare a livello di web.config e di Pagina
  - pageOutput abilita l'output sulla pagina o richiamando **trace.axd**

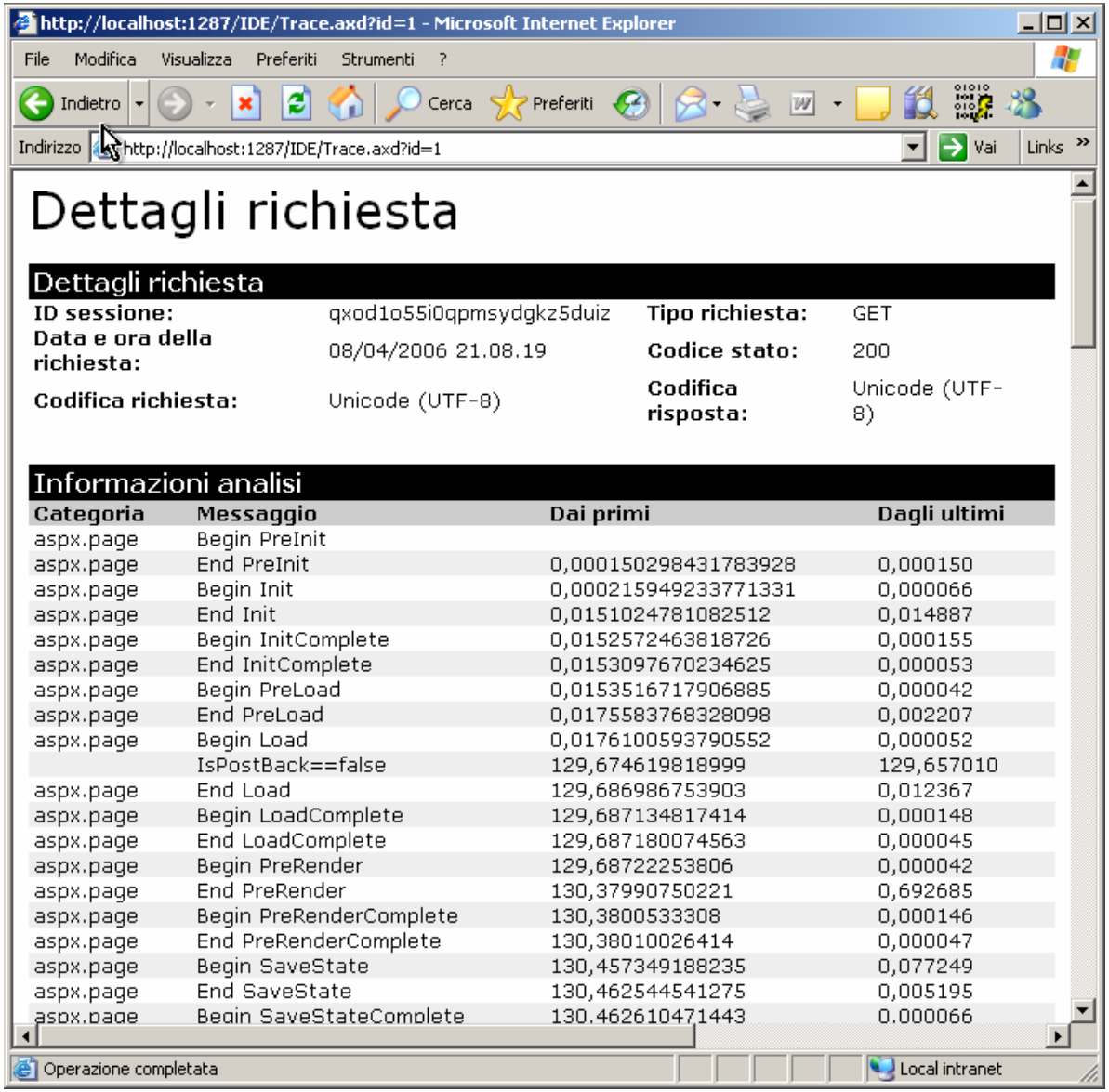
```
<trace enabled="true" pageOutput="false" />
```

```
<%@ Page Language="C#" ... Trace="true"%>
```

- Per scrivere

```
Trace.Write (categoria, messaggio, eccezione);
Trace.Warn (categoria, messaggio, eccezione);
Trace.Write (messaggio);
...
```

# Esempio di Trace




Microsoft

# Ciclo di Vita di una Pagina (Cenni)

- **PreInit**: serve per
  - Usare la proprietà IsPostBack
  - Creare controlli dinamici
  - Applicare temi e pagine master dinamicamente
  - Leggere e scrivere profili utente
- **Init**: leggere e inizializzare le proprietà dei controlli
- **Load**: leggere e aggiornare le proprietà dei controlli
- **PreRender**: apportare modifiche ai contenuti della pagina
- **UnLoad**: operazioni di chiusura finale

# Membership, Ruoli e controlli per il log-in

## Autenticazione e autorizzazione

# Autenticazione: scenari

- Riconoscere **chi** si sta loggando al nostro sito web
- Due scenari tipici per l'autenticazione:
  - Intranet: si appoggia su sistemi di autenticazioni della intranet aziendale
    - Internet Information Server (IIS) usa la Integrated Authentication (ad esempio)
    - Tipicamente gli utenti sono su Active Directory.
  - **Internet**: può appoggiarsi su un database per la gestioni degli utenti

Microsoft

# Demo

- Creazione di un sito:
  - Usare il sito di amministrazione
  - Usare i controlli di login
  - Usare le API di Memership e Role

Microsoft

# Usare il sito di amministrazione

- Per definire il tipo di autenticazione
  - Modifica il file web.config
- Gestire utenti
  - Creare, cancellare, modificare ...
- Gestire i ruoli e regole
  - Per distinguere l'autorizzazione

# Controlli per il log-in

- Interagiscono con un provider per la gestione delle funzionalità di membership
- **Login**: permette di effettuare la login usando nickname e password
- **LoginView**: permette di inserire contenuto diverso per utenti autenticati e non
- **PasswordRecovery**: posso recuperare la password (mail) rispondendo ad una domanda
- **LoginStatus**: dice se l'utente è loggato o no
- **LoginName**: nome dell'utente in logon
- **ChangePassword**: per cambiare password
- **CreateUserWizard**: molto codice risparmiato!

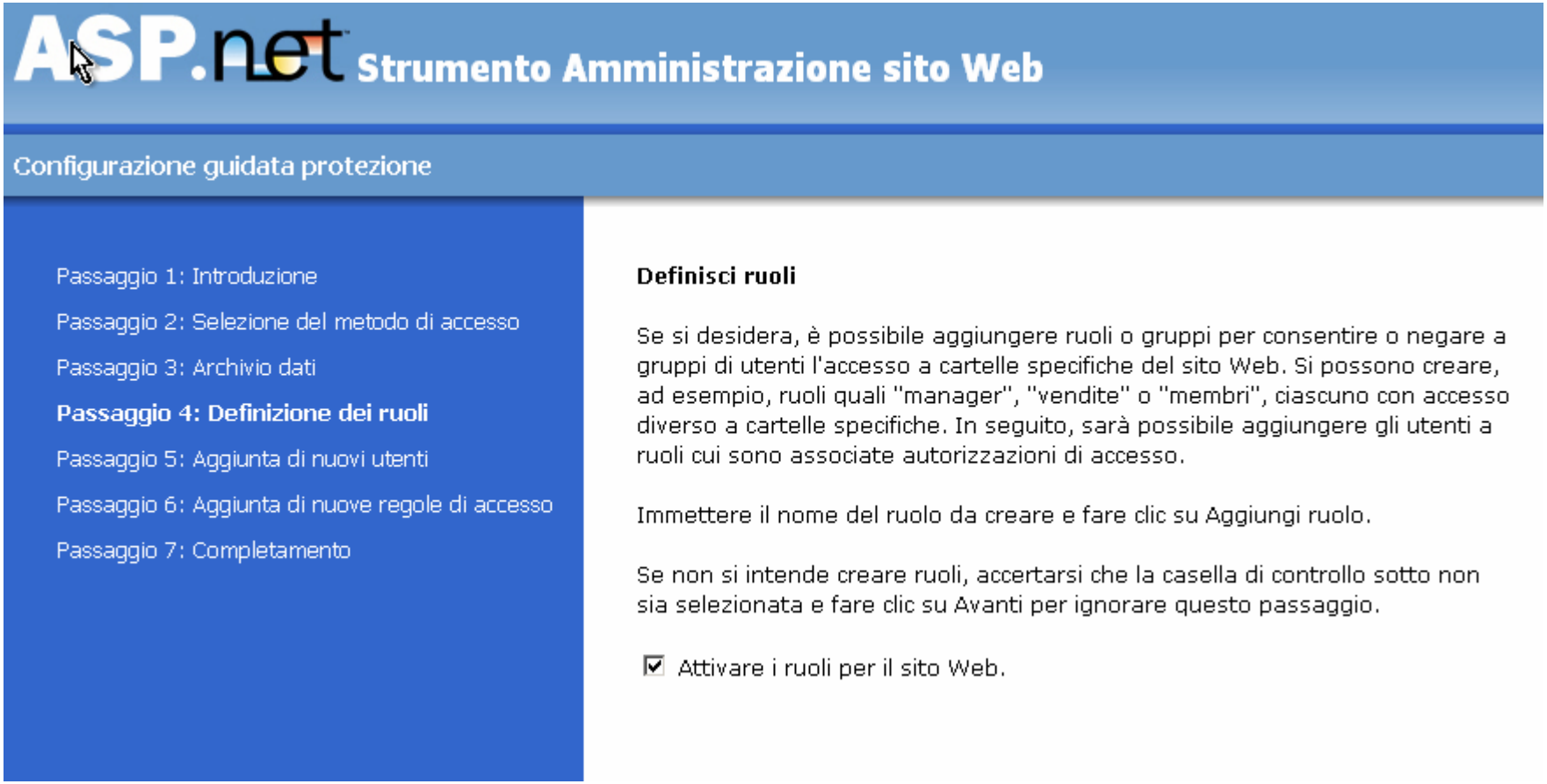
# Autorizzazione

- Permette di distinguere i diversi ruoli di un utente
  - Dopo che si è autenticato!
  - Usando il sito web di amministrazione: associazione di directory a ruoli
    - Il ruolo viene abilitato nel web.config
    - E' possibile fare il cache del ruolo in un cookie cacheRolesInCookie="true"
  - Via codice nella pagina:
    - Usare **User.Identity.IsAuthenticated**
    - Usare **Roles.IsUserInRole("UserRole")**

# Esempio: protezione del sito

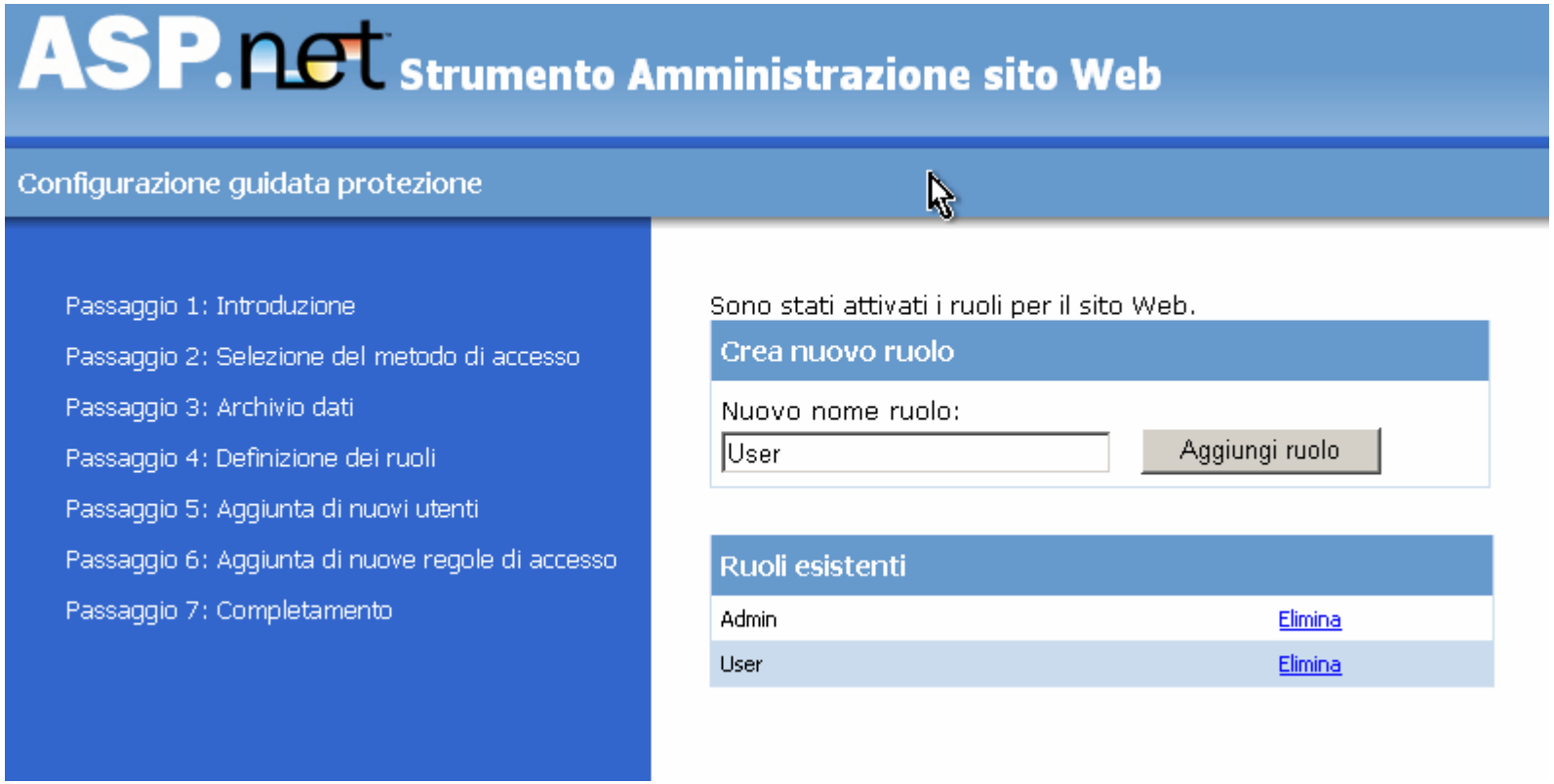

Microsoft

# Esempio: attivazione ruoli




Microsoft

# Esempio: creazione ruoli

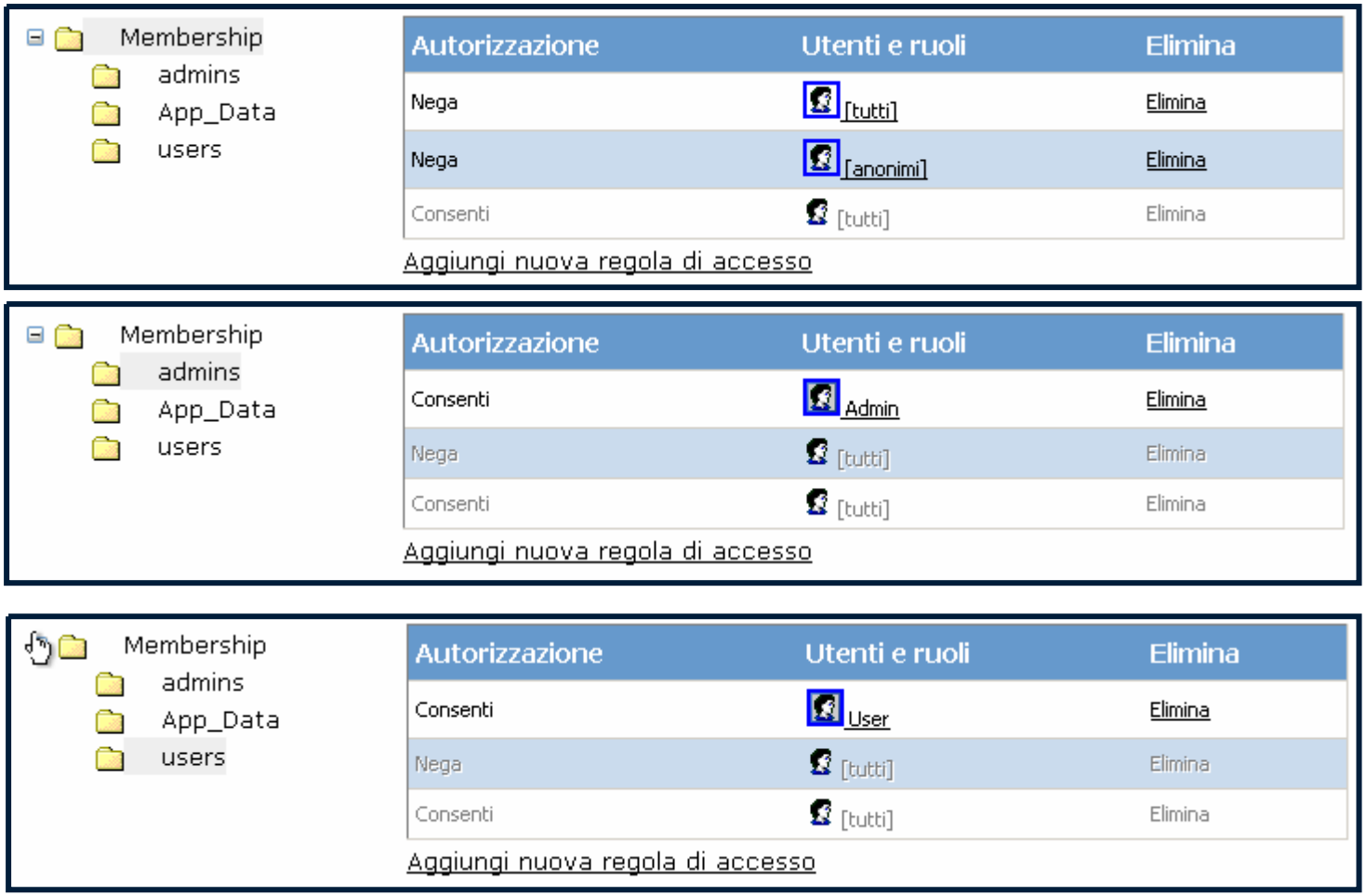



Microsoft

# Esempio: creazione utenti


Microsoft

# Esempio: regole di accesso




Microsoft

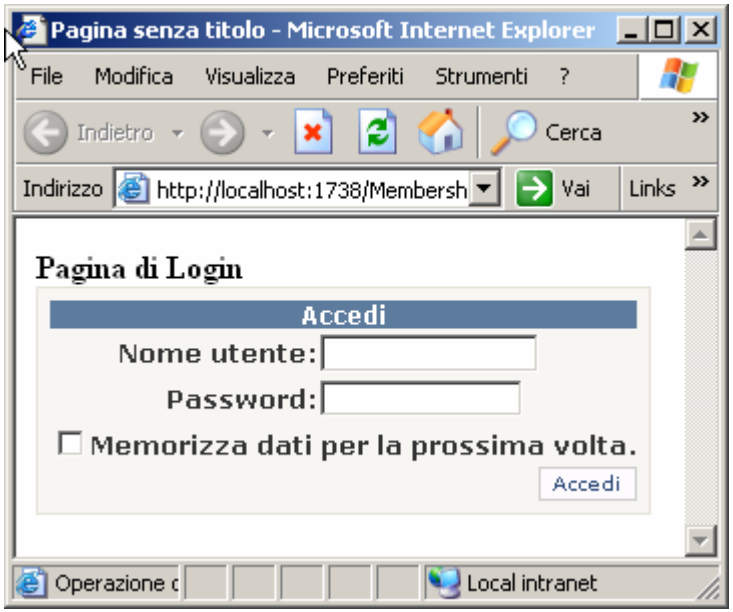
# Regole di Accesso: web.config

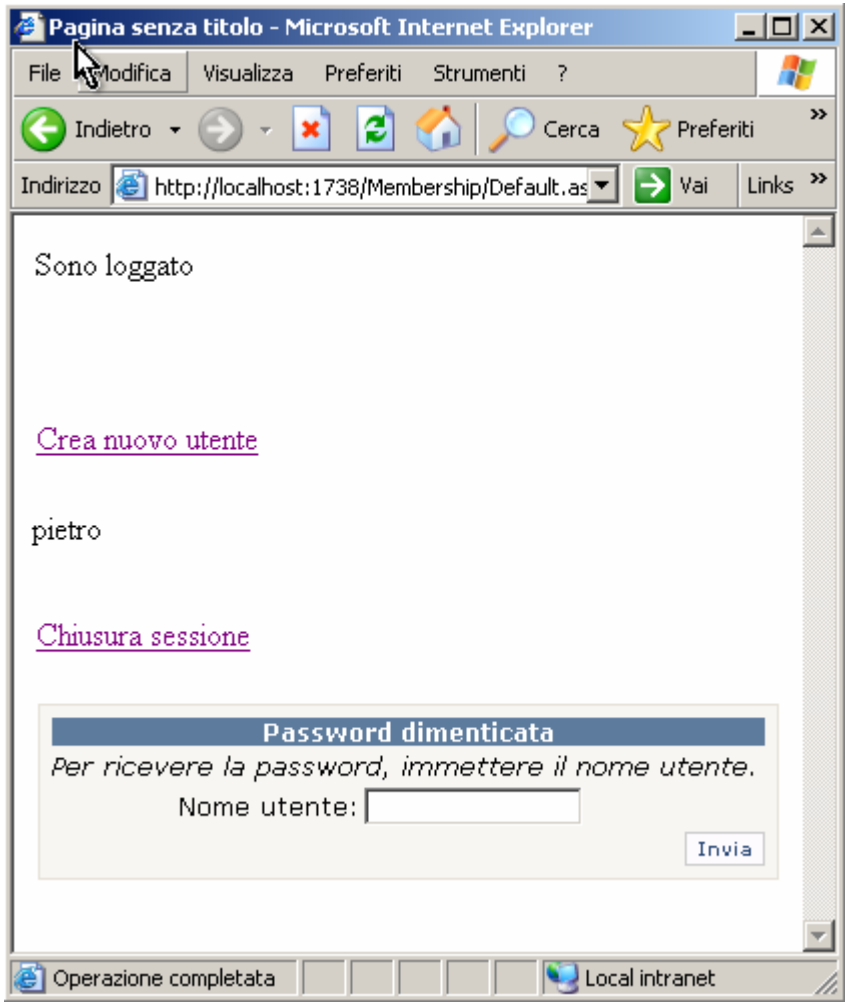
- Nella directory in cui si vuole gestire gli accessi

```
<system.web>
      <authorization>
         <allow roles="User" />
         <deny users="*" />
      </authorization>
  </system.web>
```

Microsoft

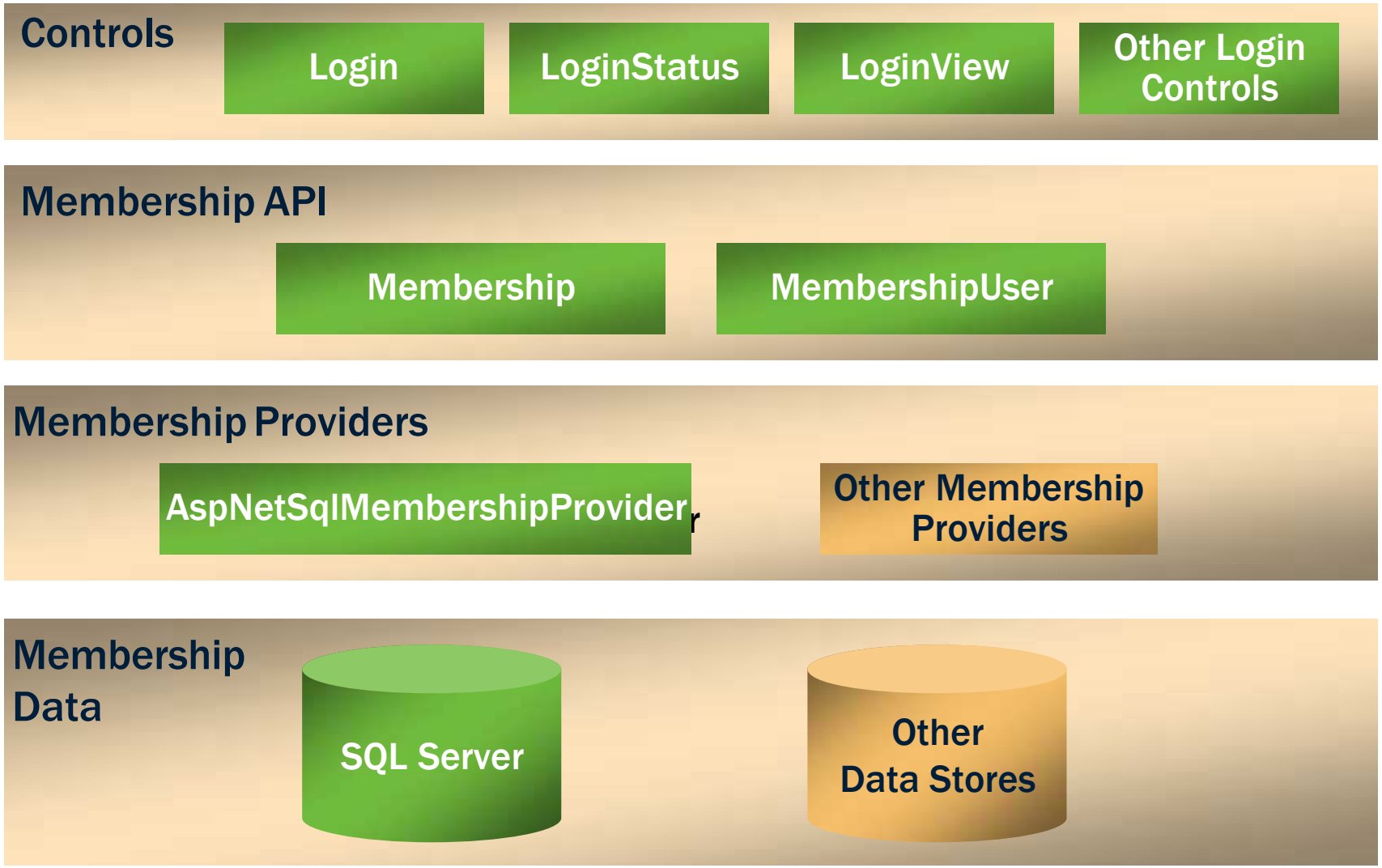
# Esempio di uso di controlli




Microsoft

# Un modello estendibile




Microsoft

# Esempio di uso API Membership

```
MembershipCreateStatus ms;
MembershipUser user=Membership.CreateUser(
    “pietro",
    “pietro",
    “pietro@yeah.com",
    “Colore?",
    “Rosso",
    true,out ms);

if (user==null)
  Label1.Text = "Non è possibile creare un utente";
else
  Label1.Text = "Utente creato";
```

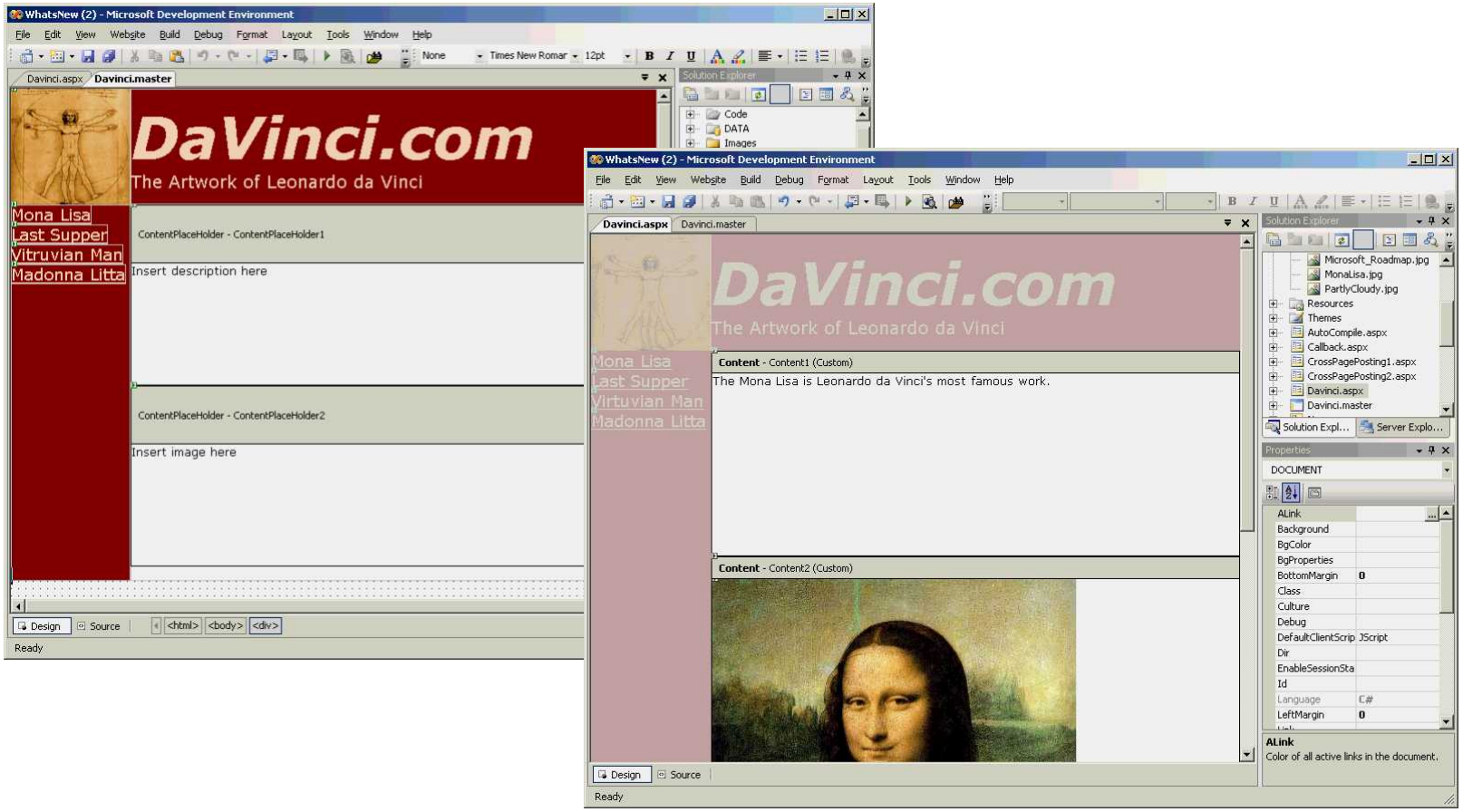
```
if (Membership.ValidateUser(username.Text, password.Text))
  FormsAuthentication.RedirectFromLoginPage(username.Text, false);
else
  Label1.Text = "Username e password non corretti";
```

Microsoft

# Pagine Master, Temi e Skin

Come creare una grafica del sito consistente ...

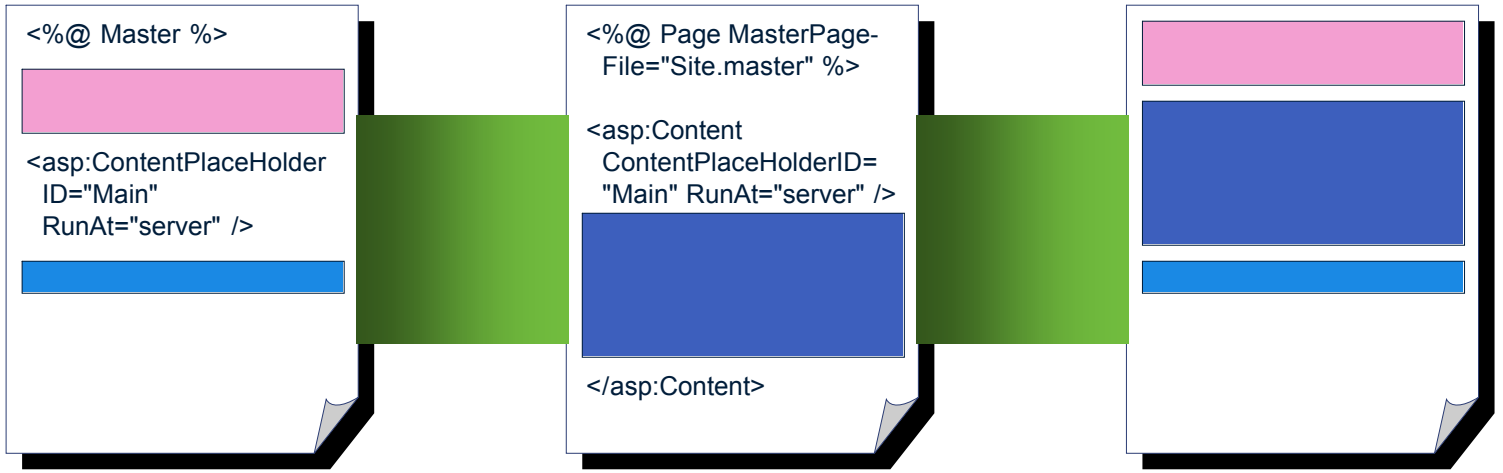
# Pagine Master




Microsoft

# Pagine master

- Le pagine master definiscono la struttura e dei place holder (<asp:ContentPlaceHolder>)
- Le pagine “figlie” referenziano la master e creano il contenuto (<asp:Content>)




Microsoft

# La proprità Page.Master

- Ottiene un riferimento alla pagina master dalla pagina figlia
- Usata per avere accesso programmatico al contenuto della pagina master
  - Usare FindControl per weak typing
  - Usere public property nella master page per strong typing

# Accedere ad un Controllo della Master Page
## Weak Typing

*Nella master page...*

```
<asp:Label ID="Title" RunAt="server" />
```

*Nella pagina figlia...*

```
((Label) Master.FindControl ("Title")).Text = "Orders";
```

Microsoft

# Accedere ad un Controllo della Master Page
# **Strong Typing**

***Nella master page...***

```
<asp:Label ID="Title" RunAt="server" />
  .
  .
  .
<script language="C#" runat="server">
public string TitleText
{
    get { return Title.Text; }
    set { Title.Text = value; }
}
</script>
```

***Nella pagina figlia…***

```
Master.TitleText = "Orders";
```

Microsoft

# Demo

- Creazione di un sito
  - Uso di pagine master

Microsoft

# Temi (CSS e Skin)

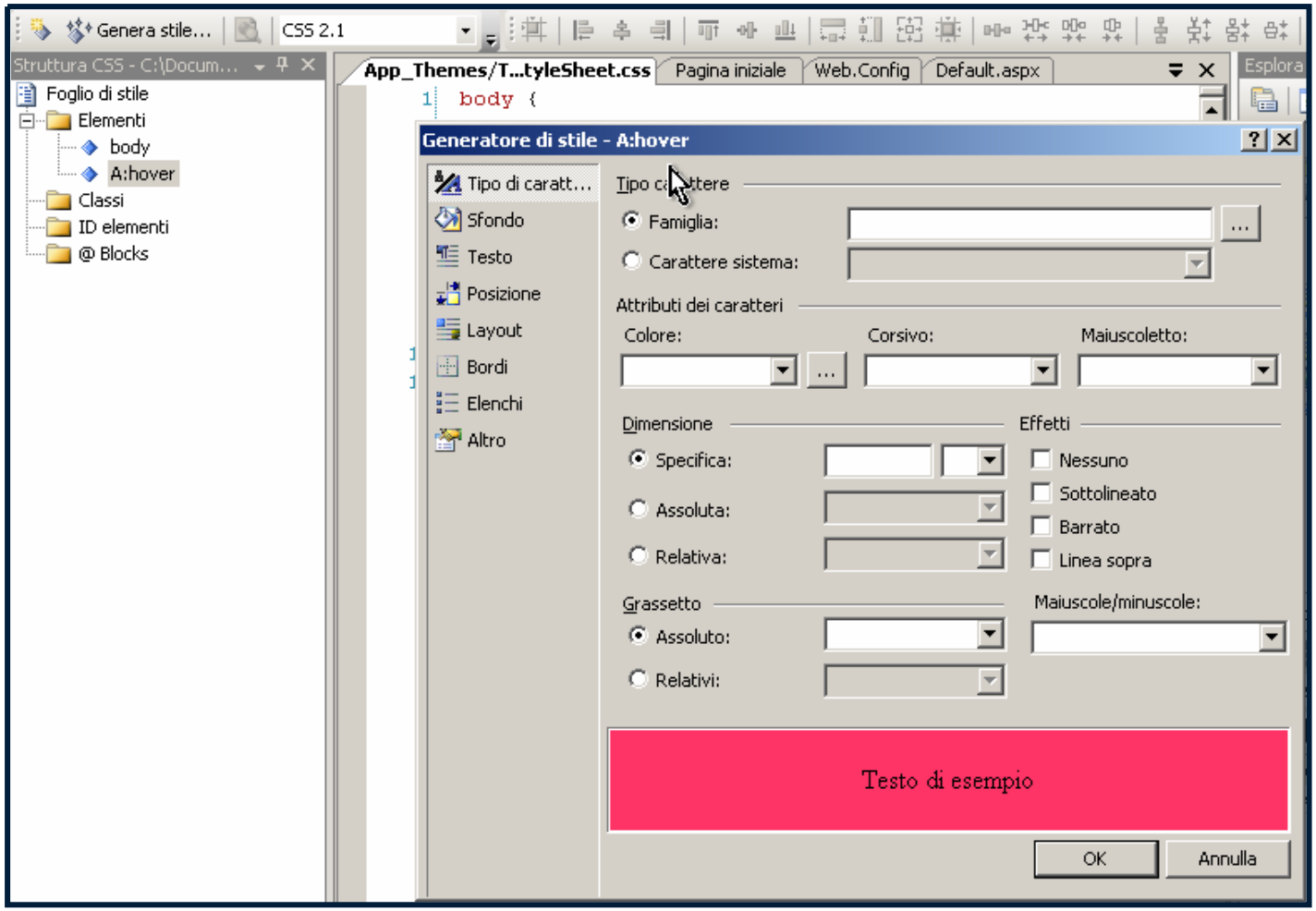
- Definiscono la grafica della pagina
- Sono formati da
  - CSS (Cascading Style Sheets)
    - Definisce l'aspetto degli elementi HTML
  - Immagini
  - Skins
    - Definisce l'aspetto dei "controlli server"
      - un calendar è difficile con i CSS !
  - Tema
    - È composta da uno o più Skin

# Temi Locali e Globali

- Locali: sono nella directory App_Themes
- Globali sono visti da tutte le applicazioni web della macchina
  - C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\ASP.NET ClientFiles\Themes
    - In questo caso per usarli in IIS : aspnet_regiis –c per rendere i temi visibili nelle applicazioni web

# Creazione di CSS

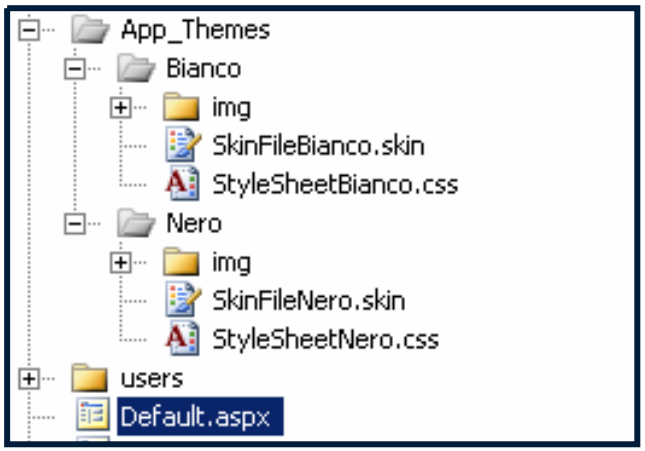
- Si usa un tool integrato nell' ambiente




Microsoft

# Creazione di Skin

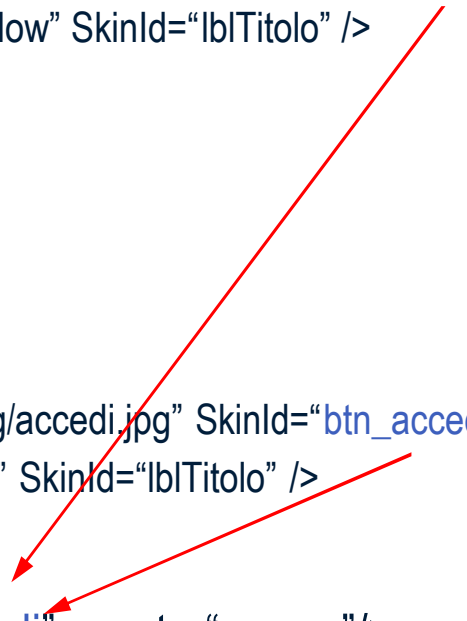
- File .Skin
  - Definisce la grafica dei controlli server
  - Definisco uno SkinID altrimenti lo skin viene applicato a tutti i controlli di quel tipo
  - <asp:**ImageButton** runat="server" **SkinId**="bottoneConImg" ImageUrl="~/App_Themes/Nero/img/home_btn.gif" />




Microsoft

# Temi e Skin (Esempio)

- App_Theme
  - **Black**
    - Img
      - Accedi.jpg
    - Default.css
    - Default.skin
      - <asp:image imageUrl="img/accedi.jpg" SkinId="btn_accedi" />
      - <asp:label backColor="Yellow" SkinId="lblTitolo" />
  - **White**
    - Img
      - Accedi.jpg
    - Default.css
    - Default.skin
      - <asp:image imageUrl="img/accedi.jpg" SkinId="btn_accedi" />
      - <asp:label backColor="red" SkinId="lblTitolo" />
- Default.aspx
  - <asp:image SkinId="btn_accedi" runat= "server"/>



Microsoft

# Applicazione di un Tema

- Si imposta nella direttiva **@Page** o **web.config**
  - Attributo **Theme**
    - Può essere globale o della pagina, quest'ultimo ha però la precedenza
  - Attibuto **StyleSheetTheme**
    - I controlli locali hanno la precedenza sulla definizione del tema
  - Cioè: prima StyleSheetTheme, poi Page, poi Theme
- Si può impostare anche da codice
  - Evento **Page_PreInit** : Page.Theme = "Mytheme"
- Si imposta a livello di controllo lo skin
  - Attributo **SkinID**
    - Permette di applicare ai soli controlli voluti lo skin
  - Altrimenti viene applicato di default a tutti i controlli del tipo definito nello skin

Microsoft

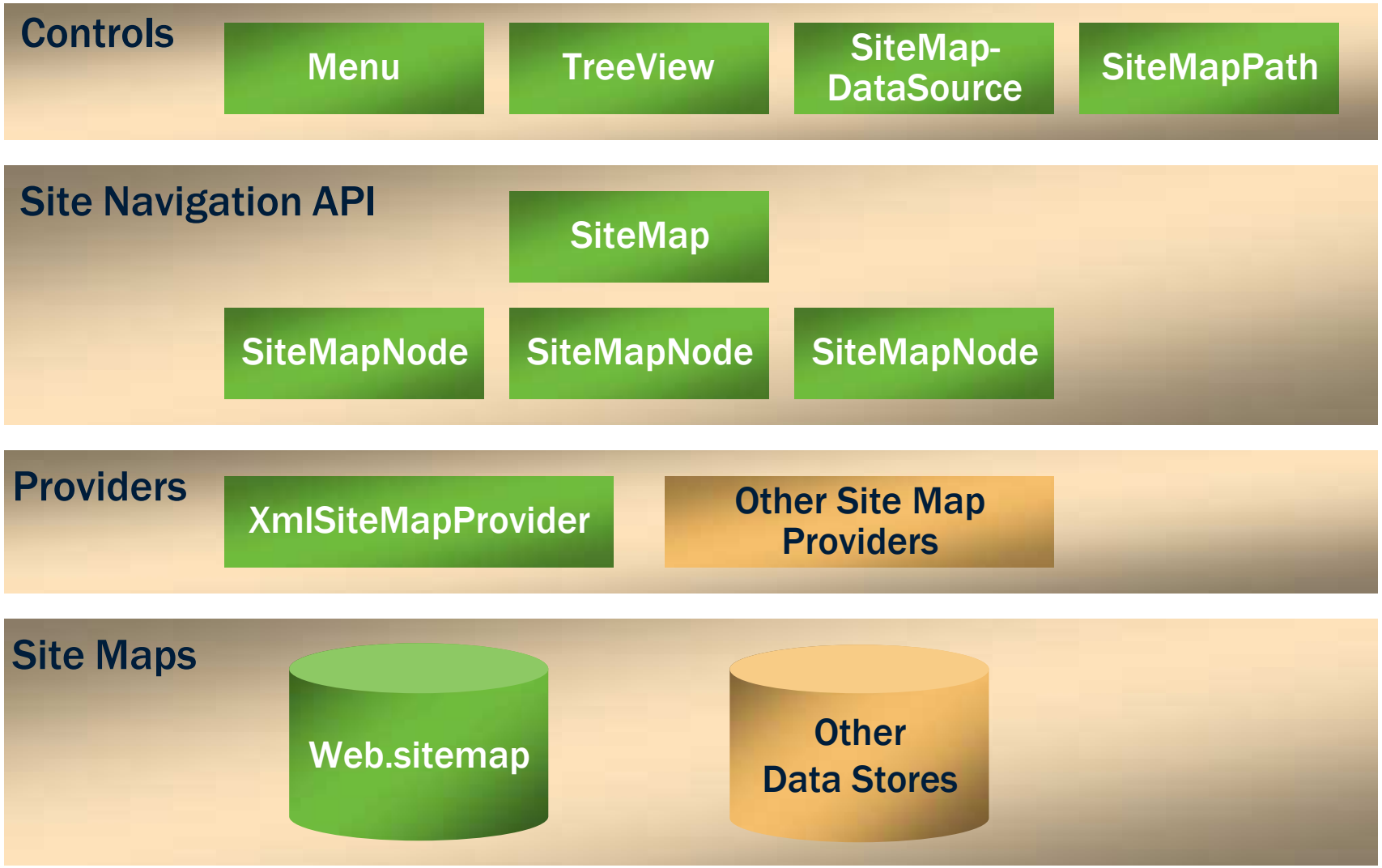
# Demo

- Uso di CSS, Skin e Temi

# Controlli di Esplorazione

- **SiteMap**: Definice la struttura di navigazione del sito in XML
- **SiteMapPath**: Home->Prodotti ...
- **Menù**: Quanto codice una volta!
- **TreeView:** rappresentazione gerarchica ad albero

# Site Navigation Schema




Microsoft

# Demo

- Creazione di un sito
  - Controlli di Navigazione

Microsoft

# Gestione dei ruoli e navigazione

```
<siteMap >
        <providers>
        <remove name="AspNetXmlSiteMapProvider" />
        <add name="AspNetXmlSiteMapProvider"
        type="System.Web.XmlSiteMapProvider"
        securityTrimmingEnabled="true" siteMapFile="Web.sitemap" />
      </providers>
    </siteMap>
```

```
<siteMapNode title="Services" roles="*">
      <siteMapNode title="Training" url="~/Training.aspx" />
      <siteMapNode title="Consulting"  url="~/Consulting.aspx" />
      <siteMapNode title="Support"  url="~/Customers/Support.aspx" />
</siteMapNode>
```

Microsoft

# Personalizzazione

Il nostro sito diverso per ogni utente ...

# Profilo utente

- ASP.NET 2.0 permette di memorizzare informazioni sui diversi utenti che accedono al sito
- Il sito diventa personalizzato per utente!
- Il sito può essere personalizzato utente per utente
  - Utenti autenticati
  - Utenti anonimi
- Anche gli utenti anonimi vengono distinti
  - Usato un ID utente in un cookie (.ASPXANONIMOUSUSER)
- È possibile “migrare” il profilo di un anonimo in autenticato

# Profilo Utente: creazione

- Definisce come vengono gestiti i diversi profili

```
<configuration>
<system.web>
  <anonymousIdentification enabled="true" />
  <profile enabled="true" defaultProvider="AspNetSqlProvider">
    <properties>
      <add name="ZipCode" allowAnonymous="true" />
      <add name="CityAndState" allowAnonymous="true" />
      <group name="Address">
            <add name="Street"  allowAnonymous="true" />
            <add name="City"  allowAnonymous="true" />
      </group>
    </properties>
  </profile>
</system.web>
</configuration>
```

Microsoft

# Profilo: <property>

- Definisce una proprietà dell'utente
  - Deafult : System.String
  - Default: allowAnonymous = false
- Attributi

| | |
|---|---|
| name | Nome proprietà |
| readOnly | Sola lettura |
| serializeAs | Come si serializzano i dati |
| provider | Provider Utilizzato |
| Type | .Net Data Type |
| allowAnonymous | Supporto per profili anonimi |

# Profilo Utente: Uso

- Le proprietà definite in precedenza sono compilate dal run-time in una classe che le rende visibili nella pagina web che si sta creando!
- I valori salvati sono memorizzati in un database

```
If (Profile.Name != null)
{
   Response.Write (“Ciao” + Profile.Name);
   Response.Write (“Indirizzo” + Profile.Address.Street);
}
```

Microsoft

# Migrazione da utente anonimo ad autenticato

- Passo da essere un utente anonimo ad essere autenticato
- Uso l'evento nel **Global.asax** per "migrare" il profilo
  - Sposto i dati del profilo che mi servono

```
<script runat="server">
   void Profile_MigrateAnonymous(Object s, ProfileMigrateEventArgs e)
   {
      ProfileCommon anonym = Profile.GetProfile(e.AnonymousID);
      Profile.Tema = anonym.Tema;

      // Bisogna cancellare il profilo anonimo
      ProfileManager.DeleteProfile(e.AnonymousID);
      AnonymousIdentificationModule.ClearAnonymousIdentifier();
   }
</script>
```

Microsoft

# Demo

- Esempio di sito con Profilo utente
  - Riconoscimento utenti anonimi
  - “Migrazione” del profilo

Microsoft

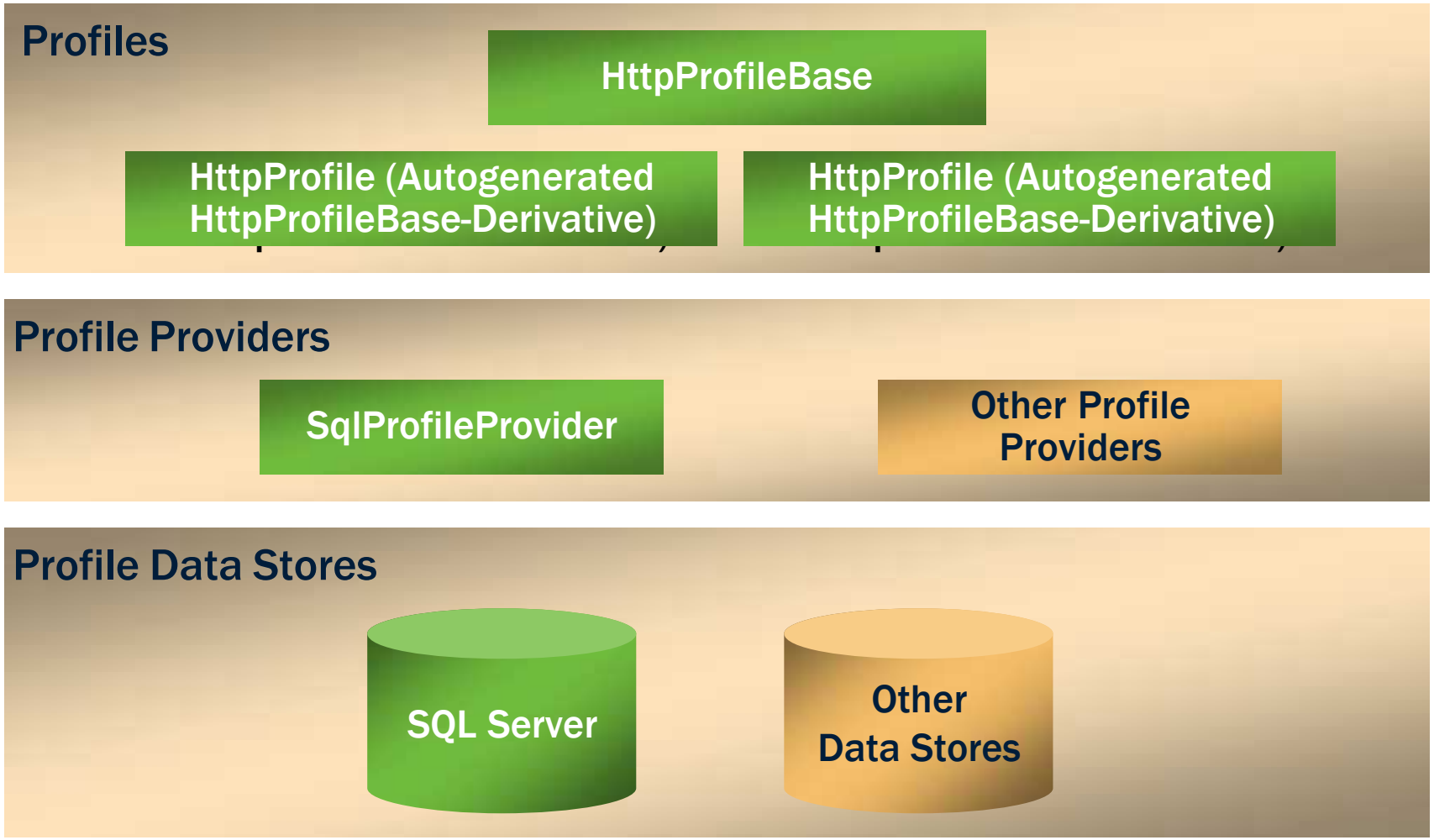
# Profilo: dove è memorizzato ?

- Se non specificato viene usato il provider di default
  - Nel **machine.config** viene specificato
  - ASPNETDB.MDF, Tabelle:
    - aspnet_Profile, aspnet_Users

```
<profile>
    <providers>
     <add name="AspNetSqlProfileProvider" connectionStringName="LocalSqlServer"
          applicationName="/" type="System.Web.Profile.SqlProfileProvider,
          System.Web, Version=2.0.0.0, Culture=neutral,
          PublicKeyToken=b03f5f7f11d50a3a" />
    </providers>
  </profile>
```

Microsoft

# Profile Schema



# Sorgenti dati e controlli data-bound

## Come usare SQL Server in una applicazione web

# Demo

- Creazione di un database clienti
- Accesso ai dati da ASP.NET

Microsoft

# Controlli Data-bound

- Alcuni controlli hanno la capacità di collegarsi a sorgenti dati e di rappresentarne il contenuto...
  - ListBox, BulletedList, RadioButtonList, CheckBoxList
  - TreeView, Menu, FormView, GridView, DetailsView
  - Datalist, Repeater
- Molte volte basta un semplice Drag & Drop !
  - Zero code !

# Controlli DataSource

- Approccio dichiarativo per ottenere i dati

| Nome | Descrizione |
|---|---|
| SqlDataSource | Connects data-binding controls to SQL databases |
| AccessDataSource | Connects data-binding controls to Access databases |
| XmlDataSource | Connects data-binding controls to XML data |
| ObjectDataSource | Connects data-binding controls to data components |
| SiteMapDataSource | Connects site navigation controls to site map data |

# SqlDataSource

- Collegamento a database SQL in modo dichiarativo
- Mascherano l'uso delle classi ADO.Net per l'accesso ai dati (Command, Connection, etc)
- Data binding bi-direzionale
  - SelectCommand
  - InsertCommand, UpdateCommand, and DeleteCommand
- Caching opzionale per il risultato delle query
- Supporto di parametri nei comandi (Select, etc)

# Usare SqlDataSource

```
<asp:SqlDataSource ID="Titles" RunAt="server"
  ConnectionString="server=localhost;database=pubs;integrated security=true"
  SelectCommand="select title_id, title, price from titles" />
```

Microsoft

# Caching dei risultati

```
<asp:SqlDataSource ID="Countries" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select distinct country from customers order by country"
  EnableCaching="true" CacheDuration="60" />

<asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
  DataTextField="country" AutoPostBack="true" RunAt="server" />
```

# Comandi con Parametri

- Le proprietà XxxParameters di parametrizzare le query fatte al database
  - Esempio: valore della clausola WHERE nella SelectCommand il cui paramtetro è preso dalla query string o da un drop-down box
- XxxParameter specificano la sorgente del parametro

# Usare i ControlParameter

```
<asp:SqlDataSource ID="Countries" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select distinct country from customers order by country" />
<asp:SqlDataSource ID="Customers" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select * from customers where country=@Country">
  <SelectParameters>
    <asp:ControlParameter Name="Country" ControlID="MyDropDownList"
      PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>
<asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
  DataTextField="country" AutoPostBack="true" RunAt="server" />
<asp:DataGrid DataSourceID="Customers" RunAt="server" />
```

# Chiamare le Stored Procedures

```
<asp:SqlDataSource ID="Countries" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="proc_GetCountries" />
<asp:SqlDataSource ID="Customers" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="proc_GetCustomers">
  <SelectParameters>
    <asp:ControlParameter Name="Country" ControlID="MyDropDownList"
      PropertyName="SelectedValue" />
  </SelectParameters>
</asp:SqlDataSource>
<asp:DropDownList ID="MyDropDownList" DataSourceID="Countries"
  DataTextField="country" AutoPostBack="true" RunAt="server" />
<asp:DataGrid DataSourceID="Customers" RunAt="server" />
```

```
CREATE PROCEDURE proc_GetCustomers
@Country nvarchar (32) AS
  SELECT * FROM Customers
  WHERE Country = @Country
GO
```

```
CREATE PROCEDURE proc_GetCountries AS
  SELECT DISTINCT Country
  FROM Customers
  ORDER BY Country
GO
```

Microsoft

# ObjectDataSource

- Permette di creare applicazioni con uno strato in più per l'accesso ai dati
  - È possibile inserire della business login
  - Il codice di accesso ai dati è separato dalla UI
- Binding bidirezionale
  - SelectMethod, InsertMethod, UpdateMethod, and DeleteMethod
- Caching dei risultati opzionale
- Parametri

# Inizializzazione e Clean-Up

- ObjectDataSource.SelectMethod possono usare metodi statici e di istanza
- Se sono usati i metodi statici:
  - ODS crea una nuova istanza ad ogni chiamata
  - La classe deve avere un costruttore di default pubblico
- Usare gli eventi ObjectCreated  e ObjectDisposing per fare operazioni di inizializzazione e pulizia prima della creazione degli oggetti che contengono i metodi per Select etc.

# Il controllo GridView

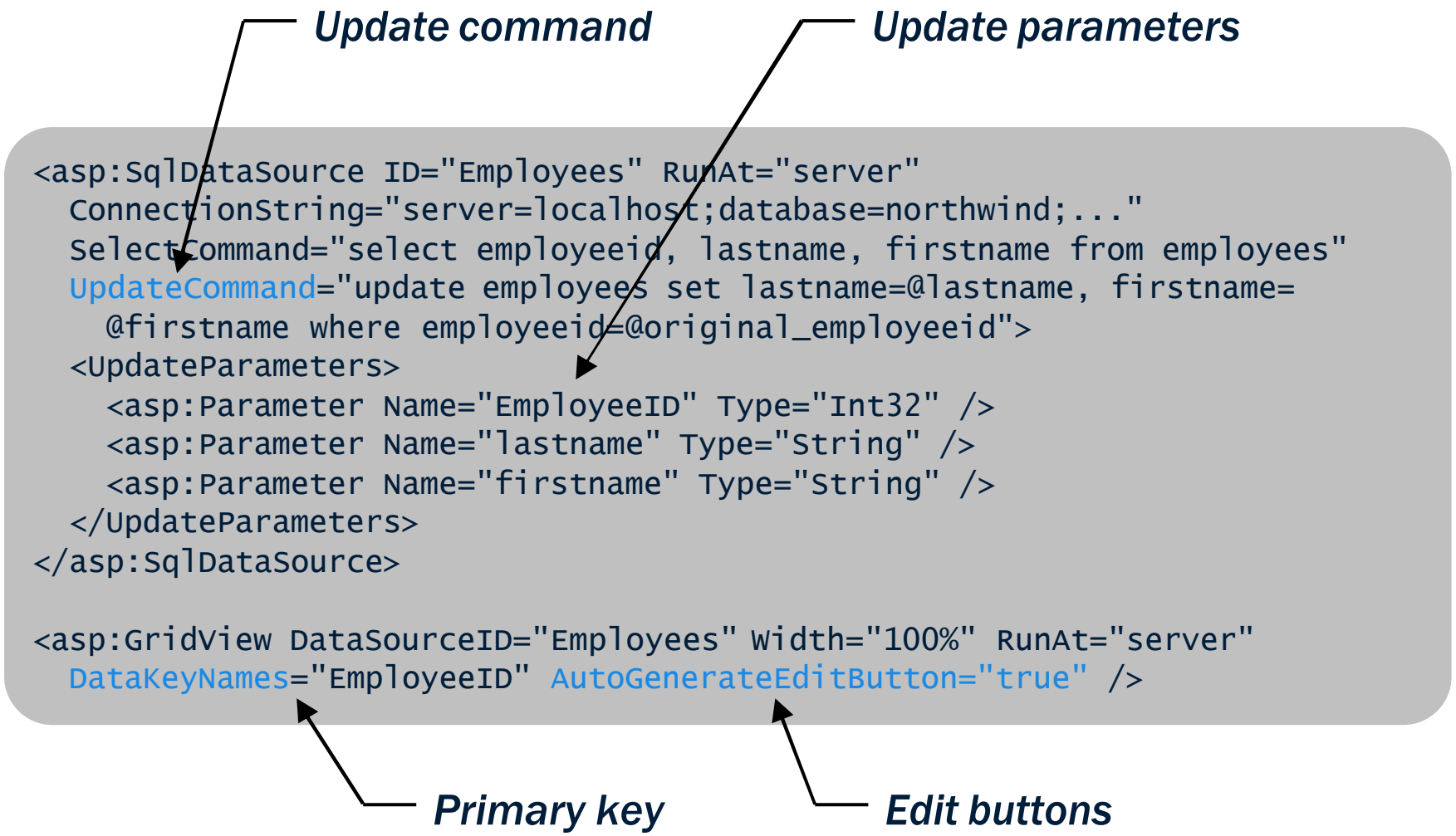
- Permette il sorting, paging, selecting, updating, ed il deleting
- Supporta colonne fatte con molti tipi, compresi i CheckBoxFields
- Interfaccia customizzabile

# Specificare il tipo dei campi

```
<asp:SqlDataSource ID="Employees" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select photo, lastname, firstname, title from employees" />
<asp:GridView DataSourceID="Employees" Width="100%" RunAt="server"
  AutoGenerateColumns="false" >
  <Columns>
     <asp:TemplateField HeaderText="Name">
      <ItemTemplate>
        <%# Eval ("firstname") + " " + Eval ("lastname") %>
      </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField HeaderText="Title" DataField="title" />
  </Columns>
</asp:GridView>
```

Microsoft

# Editing con GridViews




Microsoft

# Esempio per DetailsView

```
<asp:SqlDataSource ID="Employees" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select employeeid, photo, ... from employees" />
<asp:DetailsView DataSourceID="Employees" RunAt="server"
  AllowPaging="true" AutoGenerateRows="false"
  PagerSettings-Mode="NextPreviousFirstLast">
  <Fields>
    <asp:BoundField HeaderText="Employee ID" DataField="employeeid" />
    <asp:BoundField HeaderText="Date Hired" DataField="hiredate" />
    <asp:TemplateField HeaderText="Name">
      <ItemTemplate>
        <%# Eval ("firstname") + " " + Eval ("lastname") %>
      </ItemTemplate>
    </asp:TemplateField>
    <asp:BoundField HeaderText="Title" DataField="title" />
  </Fields>
</asp:DetailsView>
```

Microsoft

# Conflitti

- First-in-Wins(ottimistico)
  - Update fallisce se i dati sono cambiati
  - Imposta ConflictDetection="CompareAllValues"
- Last-in-Wins (pessimistico)
  - Update va a buon fine se i dati sono cambiati
  - Imposta ConflictDetection="OverwriteChanges"

# First-In-Wins Updates

```
<asp:SqlDataSource ID="Employees" RunAt="server"
  ConnectionString="server=localhost;database=northwind;..."
  SelectCommand="select employeeid, lastname, firstname from employees"
  UpdateCommand="update employees set lastname=@lastname, firstname=
    @firstname where employeeid=@original_employeeid and lastname=
    @original_lastname and firstname=@original_firstname"
  ConflictDetection="CompareAllValues">
  <UpdateParameters>
    <asp:Parameter Name="EmployeeID" Type="Int32" />
    <asp:Parameter Name="lastname" Type="String" />
    <asp:Parameter Name="firstname" Type="String" />
  </UpdateParameters>
</asp:SqlDataSource>

<asp:GridView DataSourceID="Employees" Width="100%" RunAt="server"
  DataKeyNames="EmployeeID" AutoGenerateEditButton="true" />
```

Microsoft

# Errori

- Vengono generati eventi se modifico database
  - GridView.RowUpdated
  - DetailsView.ItemUpdated
  - SqlDataSource.Updated, etc.
- Gli Event handlers ricevono un oggetto "status"
  - Rivela se ci sono state delle eccezioni nel database

# Gestire gli errori in Update

```
<asp:SqlDataSource ID="Employees" RunAt="server" ...
  UpdateCommand="..." OnUpdated="OnUpdateComplete">
    ...
</asp:SqlDataSource>
  ...
void OnUpdateComplete (Object source, SqlDataSourceStatusEventsArgs e)
{
    if (e.Exception != null) {
        // Exception thrown. Set e.ExceptionHandled to true to prevent
        // the SqlDataSource from throwing an exception, or leave it set
        // to false to allow SqlDataSource to rethrow the exception
    }
    else if (e.AffectedRows == 0) {
        // No exception was thrown, but no records were updated, either.
        // Might want to let the user know that the update failed
    }
}
```

Microsoft

# Gestione dello stato

Come preservare le informazioni dell'utente da un click al successivo ?
Come ottimizzare le performance ?

# Cosa si intende per stato

- HTTP è un protocollo state-less
- Come si memorizza cosa fa l'utente?
  - Applicazione che ha un "cestino acquisti"
- Come si inizializza una applicazione web?
  - Configurazione comune
- Come si aumentano le performance di un sito?

# Implementazione in ASP.NET 2.0

- *Campi nascosti*
- *Cookie*
- *Query String*
- Proprietà di profilo (già vista)
- ViewState
- ControlState
- Oggetto **Session**
- Oggetto **Application**
- Oggetto **Cache**
- Cache della **pagina**

Microsoft

# ViewState

- Mantiene lo stato a livello di **Pagina**
  - È un dizionario nome/valore
- È un campo Hidden della pagina
  - Viene fatto l'hash per evitare il tampering
- Può essere usato anche programmaticamente
  - ViewState.Add("NomeUtente", "Pietro")
- Performance ...
  - Migliore rispetto alla versione ASP.NET 1.x (~ 20%)
  - Attenzione esegue round-trip e può essere costoso
  - Può essere disabilitato, ma attenzione al funzionamento dei controlli
  - Attenzione quando si hanno grosse moli di dati
  - Dati sensibili (può essere cifrato)

Microsoft

# ViewState: quando serve ?

- Con le propietà dei controlli
  - Se disattivato (EnableViewState="false“) il valore non viene mantenuto

```
protected void Page_Load(object sender, EventArgs e)
   {
      if (!IsPostBack)
      {
         TextBox1.ReadOnly = true;
      }
   }
```

Microsoft

# Control State (vs ViewState)

- ViewState utile ma può essere costoso
  - **Tutte** le proprietà del controllo vengono salvate
    - DataGrid: OK per paginazione, KO per performance
  - Se creo o uso un controllo per cui le proprietà servono al suo funzionamento e la pagina ha il viewstate disabilitato allora il controllo non funziona!
- **Control State**: permette di salvare solo le proprietà che servono
  - ko: bisogna implementare del codice!
  - Ok: non può essere disabilitato

# Control State: Esempio

```
public class SampleControl : Control
{
  int _idx = 0;
  protected override void OnInit(EventArgs e)
  {
    Page.RegisterRequiresControlState(this);
    base.OnInit(e);
  }
  protected override void LoadControlState(object savedState)
  {
    object[] rgState = (object[])savedState;
    base.LoadControlState(rgState[0]);
    _idx = (int)rgState[1];
  }
  protected override object SaveControlState()
  {
    object[] rgState = new object[2];
    rgState[0] = base.SaveControlState();
    rgState[1] = _idx;
    return rgState;
  }
}
```

Microsoft

# Oggetto Session

- ASP.NET associa un ID a più richieste fatte dallo stesso browser
  - ID identifica una sessione associata al browser
    - Più browser aperti, più session ID
  - ID della sessione è memorizzato in un **cookie** o nella **query string**
  - Può memorizzare il contenuto della sessione in diversi storage (vedi dopo)
  - Configurazione nel **web.config ...**

# Memorizzazione dello stato (Session)

- **In Process**: Internet Information Server
  - Pro: Efficiente
    - Non c'è serializzazione, il contenuto è l'istanza viva dell'oggetto
  - Con: Non va bene in una Web Farm, non c'è persistenza.
- **State Server**: in un Servizio NT
  - Pro: può essere usato in una Web Farm
  - Con: Occupa banda di rete per la comunicazione
    - Meno efficiente perchè serializza e deserializza
    - Il contenuto deve essere serializzabile
- **SQL Server**: all'interno di un database
  - Pro: Soluzione efficiente, scalabile e sicura
  - Con: Serve un database
    - Come sopra per la serializzazione

Microsoft

# Session vs. Profile (1)

| | Session | Profile |
|---|---|---|
| Durata | “Durata della sessione” | Perenne |
| Strongly-Typed | No | Sì |
| Partizionabile | Sì | Sì |
| Estendibile | Sì * | Sì |
| Serializzazione | BinarySerializer | XML, string, Binary, Custom |

* Dalla versione 2.0 la classe Session è derivabile

# Session vs. Profile (2)

- **Session**
  - Recupera i dati ad **ogni richiesta**
    - Per disattivare mettere off a livello di pagina
  - Per default scrive i dati al termine della richiesta
    - Impostare a ReadOnly per ottimizzare
  - Recupera tutti i dati o nessuno
- **Profile**
  - Recupera i dati quando richiesti
  - Recupera i dati di un provider
    - Partitioning per proprietà

# Oggetto Application

- È legato alla vita dell'applicazione web
  - Non alla "Sessione"
  - Ogni Sessione condivide l'oggetto Applicazione
  - "Singleton pattern"
- Viene mantenuto in memoria, solo.
- Attenzione all'uso in una web farm

# Application – Eventi (base)

- Application_Start
  - Inizializzare l'applicazione
- Application_End
  - Pulire (Dispose) di risorse
- Application_Error
  - Per gestire in modo personalizzato gli errori

# Application – Eventi (tutti)

- BeginRequest
- AuthenticateRequest / PostAuthenticateRequest
- AutorizeRequest / Post ...
- ResolveRequestCache / Post ...
- AcquireResquestState / Post ...
- ReleaseRequestState / Post ...
- UpdateRequestCache / Post ...
- EndRequest
- Disposed
- PreSendRequestContext
- PreSendRequestHeaders

# Oggetto Cache

- Simile ad Application, ma
- Puoi controllare per quanto tempo la cache è valida
  - **Sliding Expiration**: per quanto è valida
  - **Absolute Expiration**: quando scade
- La scadenza può dipendere dalle risorse:
  - **Key**: un elemento della cache dipende da un altro
  - **File**: se file esterno viene modificato la cache è invalidata
  - **SQL Server**: dipendenza da una tabella
  - **Aggregate**: da più elementi, basta che uno cambi ...
  - **Custom**: ..

# Cache Dependency

- Da file o directory
- Da un altro elemento in cache
- Da un array di elementi in cache
- Da un' altra dependency

# Oggetto Cache: uso e dipendenze

```
Cache.Insert(“MyCache", "Cached Item 1");
```

```
Cache.Insert("MyCache", "Cached Item 2", new CacheDependency(
Server.MapPath("XMLFile.xml")));
```

```
CacheDependency dep1 = new
CacheDependency(Server.MapPath("XMLFile.xml"));

string[] keyDependencies2 = { "CacheItem1" };

CacheDependency dep2 = new CacheDependency(null, keyDependencies2);

AggregateCacheDependency aggDep = new AggregateCacheDependency();
aggDep.Add(dep1);
aggDep.Add(dep2);
Cache.Insert(“MyCache", "Cached Item 3", aggDep);
```

Microsoft

# Oggetto Cache: scadenza

```
Cache.Insert("CacheItem6", "Cached Item 6", null,
DateTime.Now.AddMinutes(1d), NoSlidingExpiration);
```

```
Cache.Insert("CacheItem7", "Cached Item 7", null, NoAbsoluteExpiration,
new TimeSpan(0, 10, 0));
```

# Demo

- FileCache Dependency

# Oggetto Cache: altre funzionalità

- CacheItemRemovedCallback
  - Si  può essere notificati quando un item viene rimosso dalla Cache da una funzione di call back, dove può eseguire la logica per ripristinare la cache.
- CacheItemPriority
  - Permette di indicare quale priorità hanno gli elementi della cache, questo permette al run-time di poter scegliere quali elementi rimuovere quando la memoria del sistema deve essere liberata

# SQLCacheDependency

- È possibile impostare la dipendenza da un database
  - **Polling**: SQL Server 7/2000/2005
    - Bisogna abilittare il database e la tabella
      - aspnet_regsql -E -d database /ed
      - aspnet_regsql -E -d database -t tabella –et
    - Usando le API
      - SqlCacheDependencyAdmin.EnableNotifications(connstring);
      - SqlCacheDependencyAdmin.EnableTableForNotifications(con, "Cliente");
  - **Notifica**: SQL Server 2005 (SQL Server Broker Service)
    - System.Data.SqlClient.SqlDependency.Start() deve essere chiamato da qualche parte

# Impostare la SQL Dependency

- La dipendenza è impostabile in
  - Nell' oggetto Cache
  - Nei controlli Data Source
  - Pagina (@OutputCache)
    - Vedi dopo

# Impostare la SQL Dependency

- La dipendenza è impostabile in
  - Nell' oggetto Cache

```
string var = "valore";
Cache.Insert("Var", var, new SqlCacheDependency("dbcache", "tabella"));
```

# Impostare la SQL Dependency

- La dipendenza è impostabile in
  - Nei controlli Data Source

```
<asp:sqldatasource
  enablecaching="True"
  sqlcachedependency="dbcache:tabella"
  ...
/>
```

```
dataSourceControl.SqlCacheDependency = "dbcache:tabella";
```

# Impostare la SQL Dependency con SQL 2005

```
using (SqlConnection conn = new SqlConnection(ConnectionString))
{
  conn.Open();
  SqlCommand comm = new SqlCommand("select campo from tabella", conn);
  Response.AddCacheDependency(new SqlCacheDependency(comm));
}
```

Microsoft

# Demo

- SQL Cache Dependency

# Cache della pagina

Performance ...

# Cache della pagina (Output Cache)

- Serve a migliorare le performance
- Viene fatto il cache della pagina o di parte di essa
  - È possibile specificare la dipendenza da certi parametri
- La pagina in cache viene memorizzata
  - Sul Client, sul server ma anche sui proxy

# Impostare la cache

```
<%@ OutputCache Duration="60" VaryByParam="None"%>
```

```
<%@ OutputCache CacheProfile="Cache30Seconds" %>
```

```
<caching>
  <outputCacheSettings>
    <outputCacheProfiles>
      <add name="Cache30Seconds" duration="30" varyByParam="none" />
    </outputCacheProfiles>
  </outputCacheSettings>
</caching>
```

# Cache della pagina: più versioni

- É possibile avere più versioni della pagina in dipendenza da
  - VaryByParam: query string
  - VaryByControl: controlli usati (textbox, etc)
  - VaryByHeader: Header http
  - VaryByCustom: dal tipo di browser o da stringhe definite

# Usare la SQL Dependency

- La dipendenza è impostabile in
  - Nella Pagina (@OutputCache)

```
<%@ OutputCache SqlDependency="dbcache:tabella" Duration="60000"
VaryByParam="none" >
```

```
SqlCacheDependency dependency = new SqlCacheDependency("dbcache",
"tabella")
Response.AddCacheDependency(dependency)
```

# Sostituzione Post-Cache

- Problema: se uso il cache della pagina tutti gli utenti vedono gli stessi banner pubblicitari!
- Soluzione: uso controllo **Substitution**
  - Scrive nella pagina i dati che NON devono essere messi in Cache

# Demo

- Controllo substitution

# Configurazione OutputCache

```
<cache percentagePhysicalMemoryUsedLimit="25" />
```

```
<outputCache>
  <diskCache enabled="true" path="C:\DiskCache"/>
</outputCache>
```

```
<outputCacheSettings>
  <outputCacheProfiles>
    <add name="Profilo" diskCacheable="false" duration="60" enabled="true"
location="Server" varyByParam="none" />
  </outputCacheProfiles>
</outputCacheSettings>
```

**Nella pagina aspx:**
**<%@ OutputCache CacheProfile="profilo" %>**

Microsoft

# Link utili

- **Gestione dello stato**
  - **Speed Up Your Site with the Improved View State in ASP.NET 2.0**
    - http://msdn.microsoft.com/msdnmag/issues/04/10/ViewState/
  - **Understanding ASP.NET ViewState**
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspp/html/viewstate.asp
- **Profilo Utente**
  - **Storing User Information with ASP.NET 2.0 Profiles**
    - http://msdn.microsoft.com/asp.net/default.aspx?pull=/library/en-us/dnvs05/html/UserProfiles.asp
- **Localizzazione**
  - **ASP.NET 2.0 Localization Features: A Fresh Approach to Localizing Web Applications**
    - http://msdn.microsoft.com/asp.net/reference/ui/default.aspx?pull=/library/en-us/dnvs05/html/asp2local.asp
- **Accesso ai dati**
  - **Data Access in ASP.NET 2.0**
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/dataaccess.asp
- **ASP.NET 2.0 Internals**
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnvs05/html/internals.asp
- **Provider toolkit**
  - http://msdn.microsoft.com/asp.net/downloads/providers/default.aspx

Microsoft

# Link utili

- **Script Debugging**
  - http://blogs.msdn.com/ie/archive/2004/10/26/247912.aspx
- **10 Tips for Writing High-Performance Web Applications**
  - http://msdn.microsoft.com/msdnmag/issues/05/01/ASPNETPerformance
- **Improving ASP.Net Performance**
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenetchapt06.asp

Microsoft