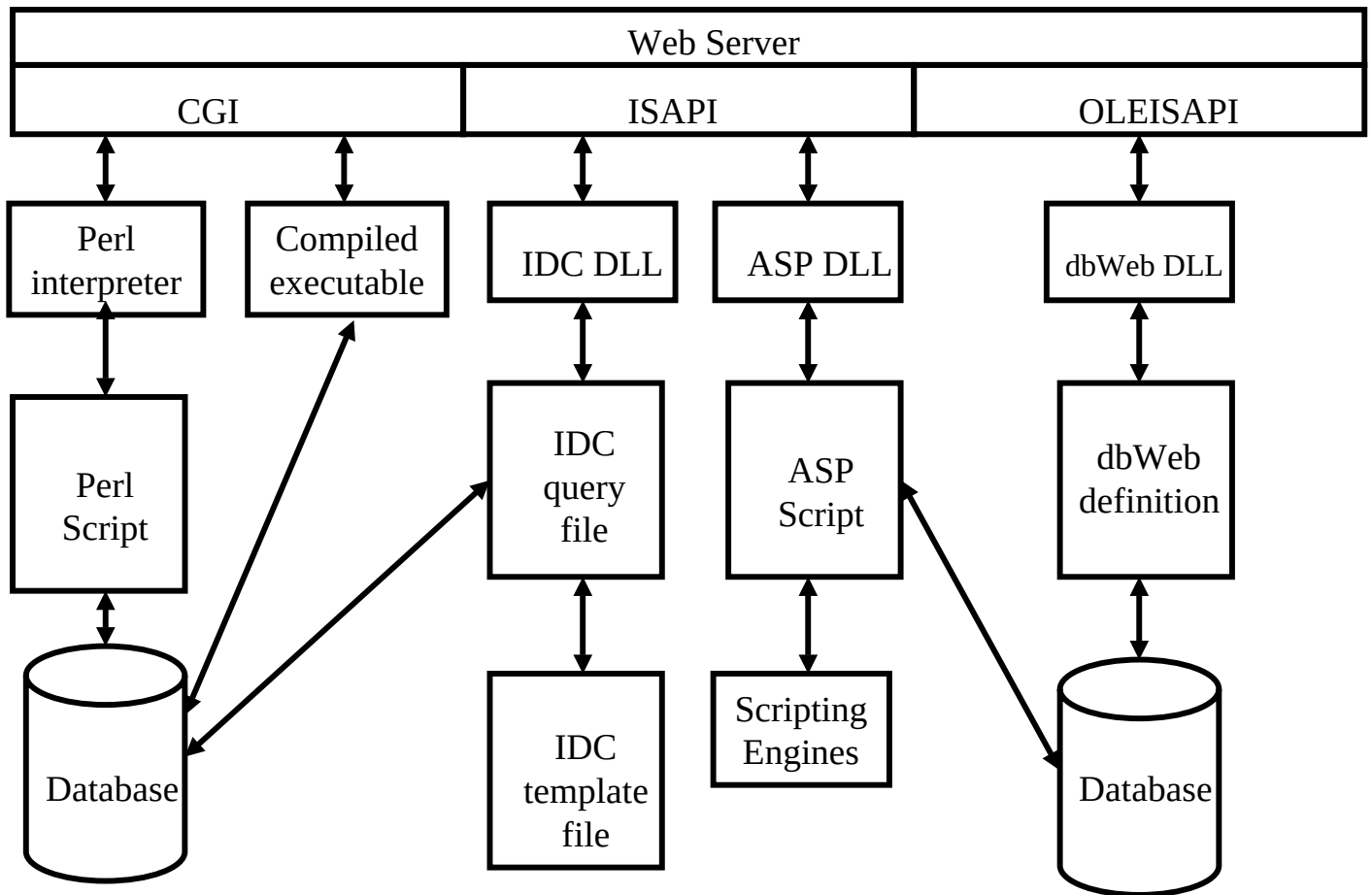
Introduzione a ASP.NET

- ASP.NET è una tecnologia lato *server* per lo sviluppo di Web Application.
- È parte integrante del *Microsoft .NET Framework*.
- J2EE : .NET Framework = servlet/JSP : ASP.net
- Non è un semplice upgrade della precedente tecnologia ASP, in quanto si basa su un *nuovo modello di sviluppo* per le applicazioni orientate al Web.

Sommario

- Cenni al Microsoft .NET Framework
- Pagine ASP.NET
  - Una nuova architettura server-side
  - Programmazione ad eventi
  - Code Inline vs. Code Behind
- Accesso alle basi di dati
  - ADO.NET
  - Esempio di utilizzo della classe DataGrid

La programmazione lato server secondo Microsoft prima di ASP.NET



Windows DNA (I)

- Nel 1996/97 Microsoft comincia a fondare le proprie strategie scommettendo sul successo di Internet.
- Viene rilasciato un modello di programmazione (per lo sviluppo di applicazioni n-tier) noto come Windows *Distributed interNet applications Architecture* (Windows DNA).

Windows DNA (II)

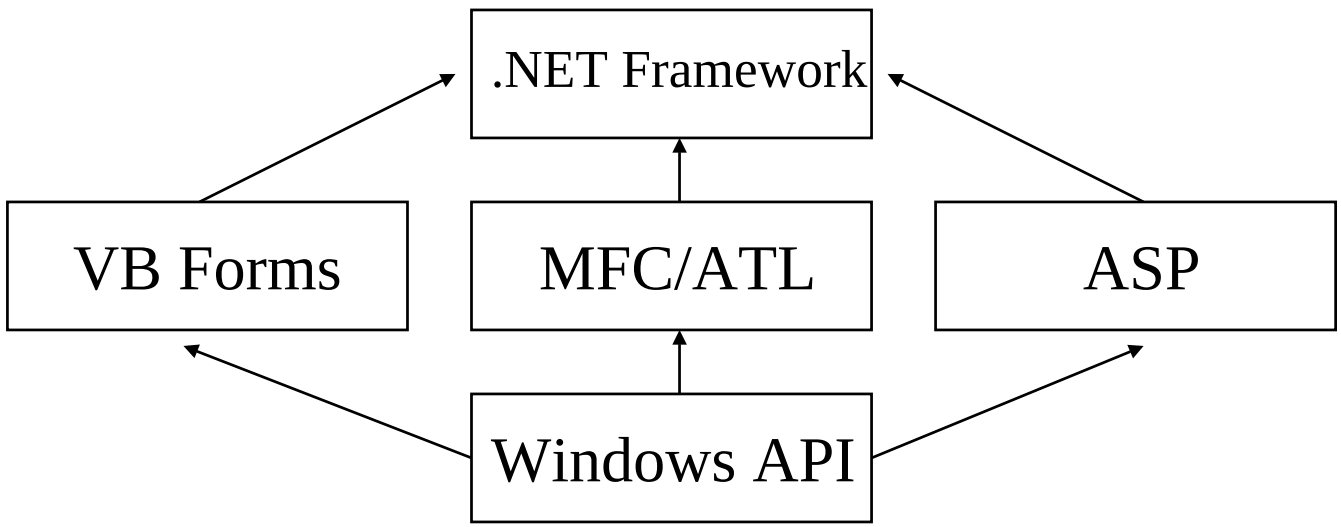
- Windows DNA si basa essenzialmente sulle seguenti tecnologie:
  - Microsoft Component Object Model (COM);
  - Win 32 API;
  - un insieme di linguaggi e protocolli proprietari per la comunicazione fra applicazioni.
- Svantaggi:
  - necessità di arrestare un servizio per installare un nuovo componente;
  - impossibilità di far coesistere versioni differenti di un componente;
  - limitazioni dei linguaggi di scripting per le pagine ASP;
  - problemi di dipendenze delle DLL.

Microsoft .NET Framework

- Il nuovo framework permette di superare i problemi di Windows DNA fornendo le seguenti soluzioni:
  - *shadow copy* (rende possibile l'aggiornamento di componenti senza interrompere l'erogazione dei servizi);
  - *side-by-side execution* (coesistenza di versioni diverse dello stesso componente);
  - *assembly* ("libreria" + manifest: informazioni sulle dipendenze da altri componenti, sulla versione, sui permessi ecc.).

Microsoft .NET Framework

- In .NET c'è un unico modo (object-oriented) di accedere alle funzionalità del framework, indipendentemente dal linguaggio di programmazione scelto (C#, VB.NET, Jscript.NET, MC++, J#).



Caratteristiche di .NET

- Supporto di diversi linguaggi di programmazione (al momento: C#, VB.NET, Jscript.NET, MC++, J#) senza grosse differenze di prestazioni.
- Supporto per diverse architetture tramite il *Common Language Specification* (CLS), progetto portato avanti con la collaborazione di HP, Intel, IBM, …
- Prestazioni: tramite il *Common Language Runtime* (CLR) il codice di un linguaggio .NET-compliant viene tradotto in *Intermediate Language* (IL) e compilato *Just-In-Time* in codice macchina al momento dell'esecuzione (metodo per metodo) o al momento dell'installazione di un'applicazione.

Servizi offerti dal CLR

- Il CLR fornisce dei servizi di base alle applicazioni .NET:
  - ***Garbage Collection*** (gestione automatica della memoria)
  - ***Code Verification*** (type safety, gestione delle eccezioni)
  - ***Code Access Security*** (gestione dei permessi accordati al codice in base alla sua provenienza, alla configurazione dell'host ecc.)

Pagine ASP.NET

- ASP.NET non è una nuova versione della tecnologia ASP, ma un nuovo paradigma di programmazione per le applicazioni Web.
- Ogni pagina diventa un oggetto completamente programmabile con compilazione Just-In-Time, caching dinamico e gestione degli eventi.

Modalità di sviluppo

- Per quanto riguarda la modalità di progettazione e sviluppo delle applicazioni web basate su ASP.NET vi sono due alternative sostanziali:
  - ***code inline*** (i frammenti di codice del linguaggio di programmazione scelto sono “iniettati” nel template HTML della pagina unitamente ai nuovi tag);
  - ***code behind*** (codice e template HTML, arricchito con nuovi tag, risiedono in file distinti): separazione della logica dalla presentazione.

Una pagina in ASP.NET (Code Inline) – Parte 1 (esempio1.aspx)

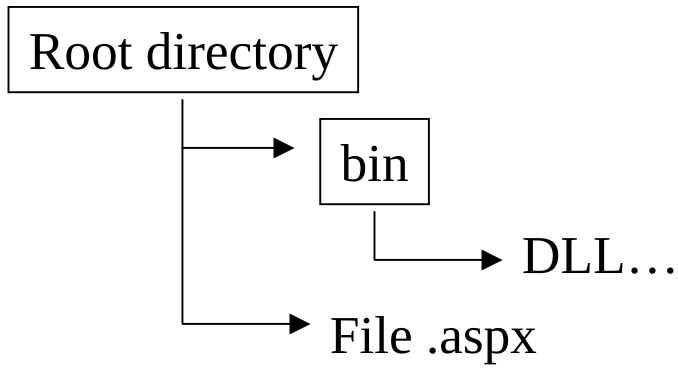
```
<html>
  <head>
    <title>
      ASP.NET Page - Code Inline
    </title>
  </head>
  <form runat="server">
    <body>
      <ASP:Label id="Etichetta" runat="server" />
      <ASP:TextBox id="CasellaTesto" runat="server" Columns="30" />
      <br><br>
      <ASP:Button id="InviaTesto" runat="server"
   OnClick="VisualizzaTesto" />
      <br><br>
      <ASP:Label id="Testo" runat="server" />
    </body>
  </form>
</html>
```

Una pagina in ASP.NET (Code Inline) – Parte 2 (esempio1.aspx)

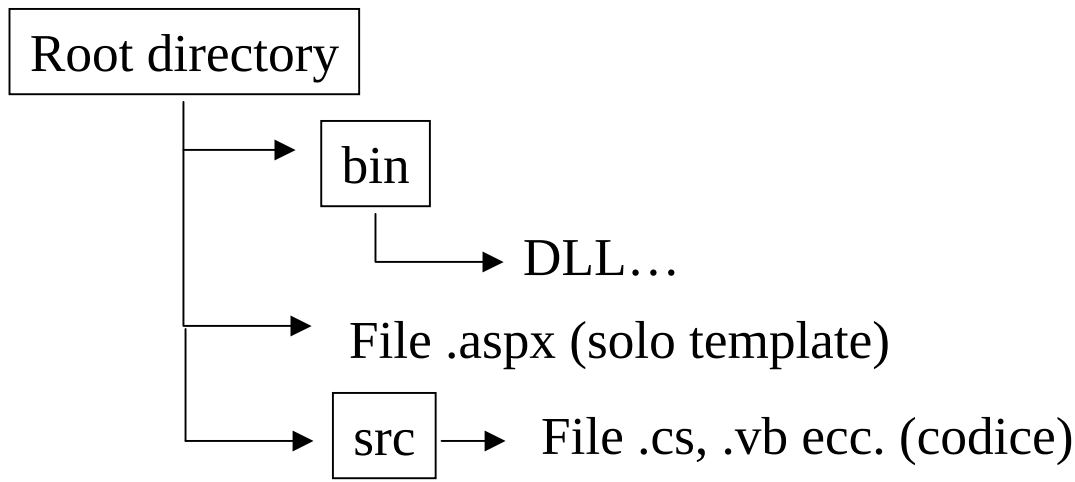
```
<SCRIPT LANGUAGE="C#" RUNAT="server">

void VisualizzaTesto(object sender,EventArgs e) {
     Testo.Text=CasellaTesto.Text.Trim().Length==0 ?
                "Non hai inserito alcun carattere." :
                "I caratteri che hai inserito sono:
   "+CasellaTesto.Text;
}


void Page_Load(object source, EventArgs e) {
     Etichetta.Text="Inserisci dei caratteri:";
     InviaTesto.Text="Invia >>";
}

</SCRIPT>
```

Layout delle directory di una web application in ASP.NET

- Code Inline:



- Code Behind:



Una pagina in ASP.NET (Code Behind) – file template (esempio2.aspx)

```
<%@ Page Inherits="Esempio" Src="Src/esempio2.cs" %>

<html>
  <head>
    <title>
      ASP.NET Page - Code Behind
    </title>
  </head>
  <form runat="server">
    <body>
      <ASP:Label id="Etichetta" runat="server" />
      <ASP:TextBox id="CasellaTesto" runat="server" Columns="30" />
      <br><br>
      <ASP:Button id="InviaTesto" runat="server"
   OnClick="VisualizzaTesto" />
      <br><br>
      <ASP:Label id="Testo" runat="server" />
    </body>
  </form>
</html>
```

Una pagina in ASP.NET (Code Behind) – file sorgente (Src/esempio2.cs)

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;

public class Esempio : Page {
        protected Label Etichetta, Testo;
        protected TextBox CasellaTesto;
        protected Button InviaTesto;

        protected void VisualizzaTesto(object sender,EventArgs e) {
              Testo.Text=CasellaTesto.Text.Trim().Length==0 ?
                        "Non hai inserito alcun carattere." :
                        "I caratteri che hai inserito sono: "+CasellaTesto.Text;
        }

        protected void Page_Load(object source,EventArgs e) {
              Etichetta.Text="Inserisci dei caratteri:";
              InviaTesto.Text="Invia >>";
        }

}
```

Pagine ASP.NET ed eventi

- ASP.NET è “event based”; gli eventi seguono quest’ordine:
  - **`Page_Init`** (la pagina è inizializzata)
  - **`Page_Load`** (la pagina è caricata)
  - **`Control Event`** (evento causato dall’attivazione di un controllo, e.g., la pressione di un pulsante).
  - **`Page_Unload`** (la pagina è scaricata dalla memoria).

Server Round-trip

- Con i controlli messi a disposizione da ASP.NET si possono costruire interfacce web che si comportano in modo simile alle interfacce utente delle comuni applicazioni stand-alone.
- Ogni interazione con il browser (e.g., pressione di un pulsante, selezione di una voce da un menu a tendina) corrisponde ad un evento *gestibile sul server*.
- L'interazione avviene tramite un round-trip, i.e., una richiesta POST inviata al server. Quest'ultimo, dopo aver elaborato l'evento rigenera la pagina e la rispedisce al client.

Il ViewState

- In un modello di programmazione ad eventi, necessariamente ci sono più interazioni con il server (da parte del browser).
- L'HTTP tuttavia è *stateless*, i.e., non conserva informazioni sulla precedente richiesta del client (e.g., valori inseriti nelle caselle di testo).
- Per ricostruire in modo trasparente il contesto della richiesta, .NET mette a disposizione l'oggetto **`System.Web.UI.StateBag`**, che permette di codificare in un campo nascosto (__VIEWSTATE) lo stato corrente della pagina.

Esempio di ViewState

```
<html>
  <head>
    <title>
      ASP.NET Page - Code Behind
    </title>
  </head>
  <form name="_ctl0" method="post"
   action="esempio2.aspx" id="_ctl0">
<input type="hidden" name="__VIEWSTATE"
   value="dDw0NTEzMTYzODY7dDw7bDxpPDE+Oz47bDx0PDtsPGk8MT
   47aTw1PjtpPDc+Oz47bDx0PHA8cDxsPFRleHQ7PjtsPEluc2VyaXN
   jaSBkZWkgY2FyYXR0ZXJpOjs+Pjs+Ozs+O3Q8cDxwPGw8VGV4dDs+
   O2w8SW52aWEgXD5cPjs+Pjs+Ozs+O3Q8cDxwPGw8VGV4dDs+O2w8S
   SBjYXJhdHRlcmkgY2hlIGhhaSBpbnNlcml0byBzb25vOiBzZHNkOz
   4+Oz47Oz47Pj47Pj47PvzsJFVQPE7eYg9Hq1xLtjUAVgxW" />
```

Accesso ai data store

- Il framework mette a disposizione una ricca serie di classi per l'accesso ai cosiddetti data store che possono essere di varia natura:
  - basi di dati relazionali
  - repositori di file XML
  - e-mail server
  - documenti "tradizionali" come file Word o Excel
  - ecc.

ADO.NET

- Come ASP.NET non è un semplice upgrade di ASP, ADO.NET ha ben poco in comune con la vecchia tecnologia ADO per l'accesso alle basi di dati.
- ADO.NET è basato sul cosiddetto "*disconnected model*":
  - non è più necessario tenere aperta una connessione per manipolare i dati di un data store;
  - le connessioni hanno luogo solamente quando è necessario: recupero iniziale ed aggiornamento del datastore.

Data binding

- Il *data binding* è un meccanismo che consente di collegare una sorgente di dati (e.g., una tabella di un database) ad un controllo su un form.
- La differenza principale, rispetto alle precedenti tecnologie Microsoft, è che il data binding avviene sul *server* invece che sul client.

Esempio di utilizzo di ADO.NET e della classe DataGrid (Code Inline) – Parte 1

```
<%@ Import Namespace="System.Data.SqlClient" %>

<html>
  <head>
    <title>
      ASP.NET Page - DataGrid
    </title>
  </head>
  <form runat="server">
    <body>
      Stato: <ASP:DropDownList id="State" runat="server" />
      <ASP:Button Text="Mostra autori" OnClick="ShowAuthors"
   runat="server" />
      <p/>
      <ASP:DataGrid id="Autori" runat="server" />
    </body>
  </form>
</html>
```

Esempio di utilizzo di ADO.NET e della classe DataGrid (Code Inline) – Parte 2

```
<script language="VB" runat="server">
  Sub Page_Load(Sender as Object, E as EventArgs)

    ' Se e' la prima volta che la pagina viene caricata
    ' (i.e., non si e' verificato un postback)
    ' il menu a tendina viene popolato con le relative voci
    ' (sigle di stati americani).
    If Not Page.IsPostBack Then
      State.Items.Add("CA")
      State.Items.Add("IN")
      State.Items.Add("KS")
      State.Items.Add("MD")
      State.Items.Add("MI")
      State.Items.Add("OR")
      State.Items.Add("TN")
      State.Items.Add("UT")
    End If

  End Sub
```

Esempio di utilizzo di ADO.NET e della classe DataGrid (Code Inline) – Parte 3

```
 ' Subroutine che viene eseguita ogni volta che si preme il pulsante a fianco
 ' della tendina nella pagina.
 Sub ShowAuthors(Sender as Object, E as EventArgs)
  Dim con as New SqlConnection("Data Source=localhost; Initial Catalog=Pubs;" &_
                               " Trusted_Connection=Yes;")
  Dim cmd as SqlCommand
  Dim qry as String
  con.Open()
  ' Query.
  qry="SELECT * FROM Authors WHERE State='" & State.SelectedItem.Text & "'"
  cmd=New SqlCommand(qry, con)
  ' Nelle successive 2 righe viene specificato che la sorgente dei dati per
  ' l'oggetto DataGrid Autori e' il risultato dell'esecuzione del comando cmd
  ' (in base alla connessione con ed alla query qry).
  ' Il comando Autori.DataBind() esegue effettivamente il binding dei dati
  ' alla griglia.
  Autori.DataSource=cmd.ExecuteReader()
  Autori.DataBind()
  con.Close()
 End Sub
</script>
```

Paginazione automatica in un DataGrid

- Le classi disponibili in .NET sono state pensate per facilitare la vita ai programmatori, in modo che questi ultimi possano concentrarsi sulla logica delle applicazioni.
- Pertanto la scrittura di porzioni di codice ripetitivo e “di routine” può essere demandata al framework stesso (generazione automatica).
- Un tipico esempio è il codice necessario per ottenere una visualizzazione per pagine del contenuto di una tabella di un database, utilizzando un DataGrid.

Paginazione (Code Inline) – Parte 1

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<html>
  <head><title>ASP.NET Page - Paginazione</title></head>
  <form runat="server">
    <body>
     Stato: <ASP:DropDownList id="State" runat="server" /><br><br>
     <ASP:RadioButton id="IndiceNumerico" GroupName="Stile" runat="server" />
     Indice numerico
     <ASP:RadioButton id="IndiceTestuale" GroupName="Stile" runat="server" />
     Indice testuale<br><br>
     Numero di righe per pagina: <ASP:TextBox id="NumeroRighe" runat="server" />
     <br><br><ASP:Button Text="Mostra autori" OnClick="ShowAuthors"
   runat="server" />
     <p/>
     <ASP:DataGrid id="Autori" runat="server"
       AllowPaging="True"
       OnPageIndexChanged="ChangeGridPage" />
    </body>
  </form>
</html>
```

Paginazione (Code Inline) – Parte 2

```
<script language="VB" runat="server">
  Sub Page_Load(Sender as Object, E as EventArgs)
    If Not Page.IsPostBack Then
      State.Items.Add("CA")
      State.Items.Add("IN")
      State.Items.Add("KS")
      State.Items.Add("MD")
      State.Items.Add("MI")
      State.Items.Add("OR")
      State.Items.Add("TN")
      State.Items.Add("UT")
      IndiceNumerico.Checked=True
      NumeroRighe.Text=5
    End If
  End Sub

  Sub ShowAuthors(Sender as Object, E as EventArgs)
    BindDataGrid()
  End Sub
```

Paginazione (Code Inline) – Parte 3

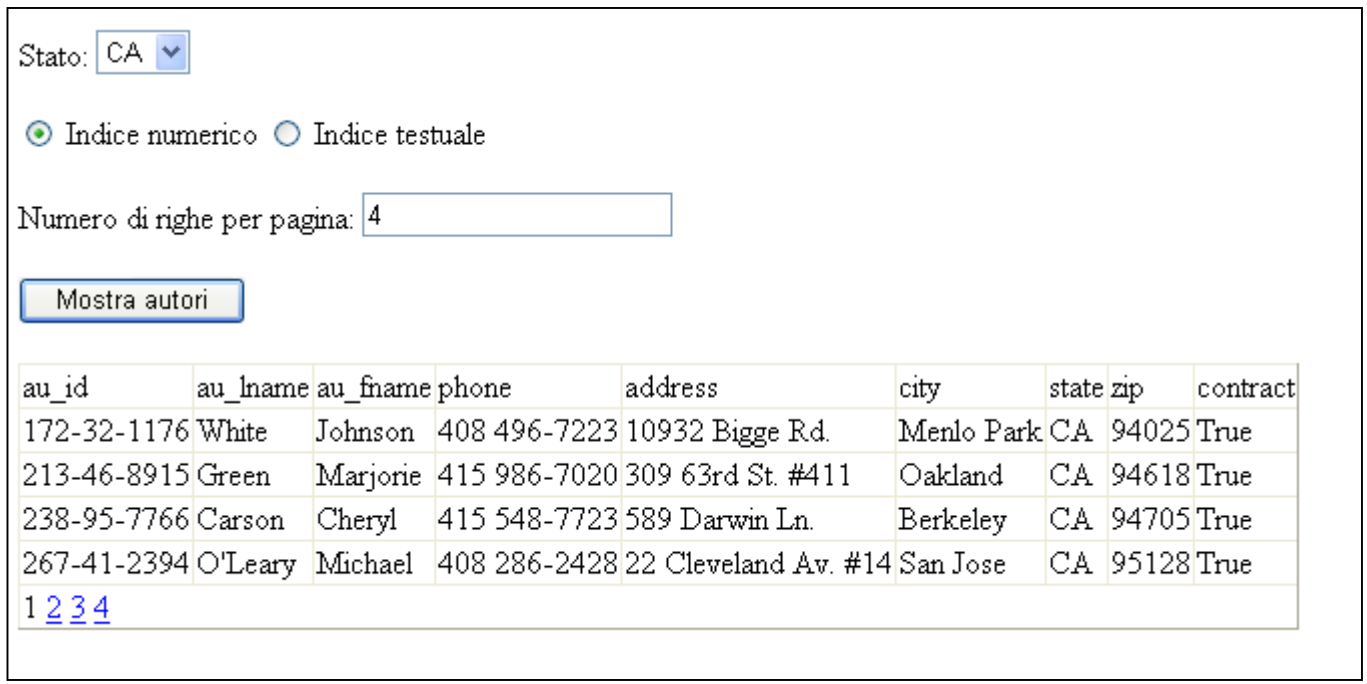
```
Sub BindDataGrid()
 Autori.PageSize=CInt(NumeroRighe.Text)

 If IndiceNumerico.Checked Then
  Autori.PagerStyle.Mode=PagerMode.NumericPages
 Else
  Autori.PagerStyle.Mode=PagerMode.NextPrev
  Autori.PagerStyle.NextPageText="Avanti >>"
  Autori.PagerStyle.PrevPageText="<< Indietro"
 End If

 Dim con as New SqlConnection("Data Source=localhost; Initial Catalog=Pubs;" & _
                              " Trusted_Connection=Yes;")
 Dim cmd as SqlCommand
 Dim qry as String
```

Paginazione (Code Inline) – Parte 4

```
  con.Open()

  qry="SELECT * FROM Authors WHERE State='" & State.SelectedItem.Text & "'"
  cmd=New SqlCommand(qry, con)
  Dim authorsDS as New DataSet()
  ' Il data adapter serve ad estrarre il risultato della query
  Dim authorsDA as New SqlDataAdapter(qry,con)
  ' Il data set viene riempito con il risultato (disconnesso) della query
  authorsDA.Fill(authorsDS,"Autori")
  ' Viene creata una vista sulla tabella disconnessa degli autori
  Dim authorsDV as New DataView(authorsDS.Tables("Autori"))
  Autori.DataSource=authorsDV
  Autori.DataBind()

  con.Close()
End Sub
```

Paginazione (Code Inline) – Parte 5

```
  ' Subroutine responsabile della gestione del cambio pagina
   Sub ChangeGridPage(Sender as Object, E as DataGridPageChangedEventArgs)
     Autori.CurrentPageIndex=E.NewPageIndex
     BindDataGrid()
   End Sub
</script>
```



Cosa è necessario per scrivere pagine ASP.NET

- Microsoft .NET Framework 2.0 (SDK)
- IIS 5.0/6.0 (disponibile in versione Personal con Windows 2000 Professional e Windows XP Professional)
- Un editor di testo
- Un database per cui esista un provider in ADO.NET (Access, SQL Server, MySQL, …), se si vuole accedere a basi di dati dalle pagine ASP.NET.

Link

- msdn.microsoft.com/netframework
- www.gotdotnet.com
- www.asp.net
- www.mono-project.com (Porting del framework .NET in ambiente Unix/Linux/Solaris/Mac OS X).