

ASP.NET 2.0

Relatore:

Massimo Piubelli

massimo@piubelli.net
www.piubelli.net

<i>Introduzione ad ASP.Net 2.0</i>	<i>1</i>
<i>I Controlli standard</i>	<i>1</i>
<i>I Controlli di convalida</i>	<i>4</i>
<i>Nuovi controlli: TreeView, MultiView e View</i>	<i>6</i>
<i>Le Master Page</i>	<i>6</i>
<i>Le Web Part</i>	<i>7</i>
<i>ComboBox e ListBox</i>	<i>9</i>
<i>Controlli utente (.ascx)</i>	<i>10</i>
<i>Bind dei dati... meno codice da scrivere</i>	<i>10</i>
<i>Controllo AccessDataSource</i>	<i>11</i>
<i>Oggetto GridView</i>	<i>13</i>
<i>Oggetto FormView</i>	<i>16</i>
<i>Oggetto DetailView</i>	<i>19</i>
<i>Oggetto ListView</i>	<i>19</i>
<i>Filtrare i dati</i>	<i>19</i>

Introduzione ad ASP.Net 2.0

Ha mantenuto il nome e una parte dell'estensione dei file, ma per il resto di asp è rimasto gran poco.

Le novità principali del framework 2.0 hanno infatti rivoluzionato in maniera abbastanza consistente lo sviluppo delle applicazioni web con tecnologia asp.

Infatti il passaggio dalla versione 1 alla 2 di asp.net ha introdotto diversi miglioramenti e ha velocizzato moltissimo le procedure che venivano più spesso svolte dai programmatori web.

Altra grossa novità è un server integrato per provare subito l'aspetto del sito web senza necessità di dover intervenire nella configurazione di IIS.

Ma tralasciando la parte puramente teorica, partiamo con la pratica, iniziando a vedere subito come si utilizzano i controlli standard.

I Controlli standard

Per inserire un controllo standard è sufficiente trascinarlo all'interno della pagina. Una delle proprietà più importanti da impostare dei vari controlli è il corrispondente della proprietà "name" ovvero la proprietà "ID". Questa proprietà ci consentirà poi di gestire il controllo durante l'esecuzione della pagina.

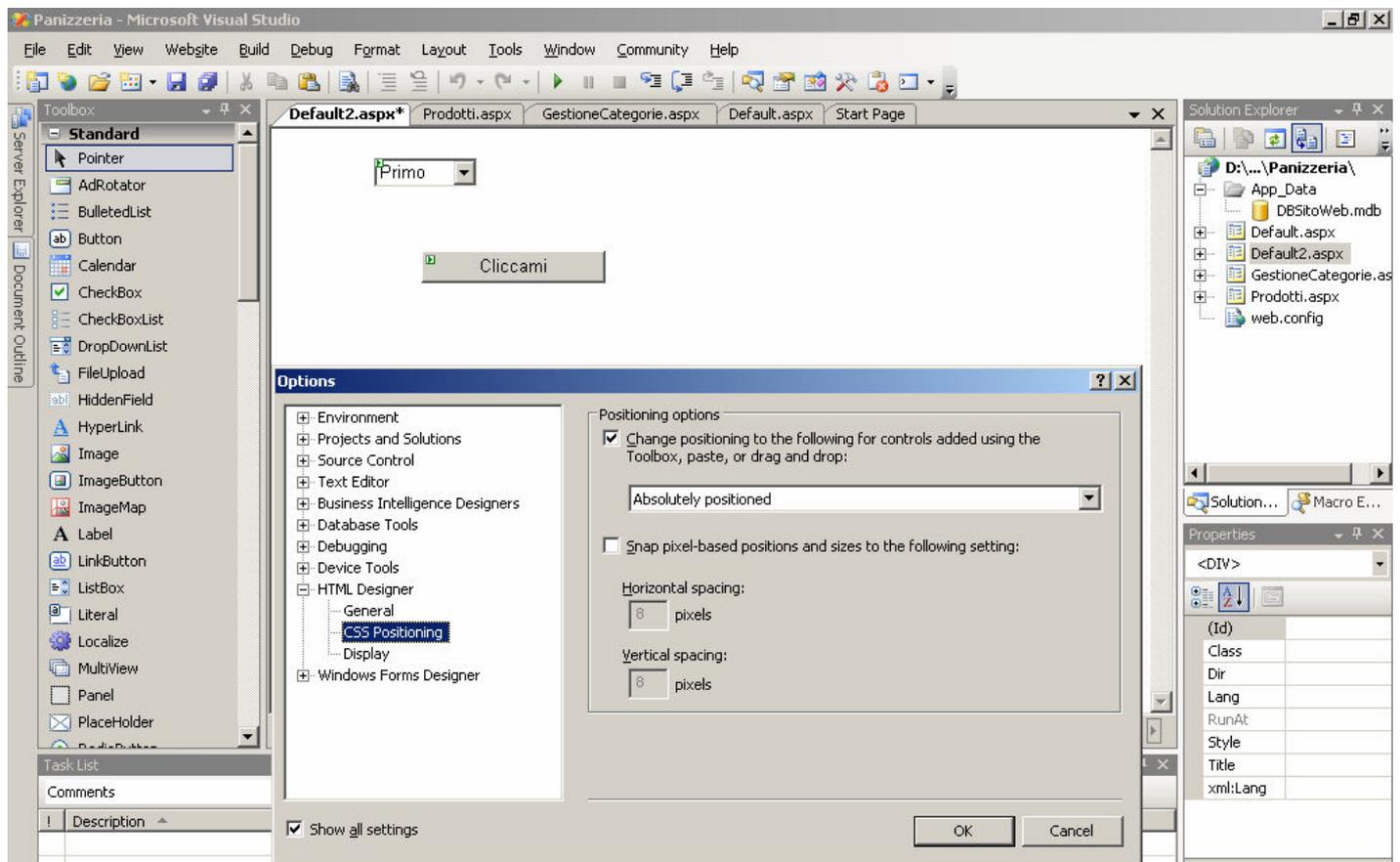
E' interessante notare come ASP va a scrivere le proprietà del controllo all'interno della pagina AspX

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default2.aspx.vb" Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <asp:Button ID="BtnTest" runat="server" Text="Cliccami"
Width="136px" />
      <asp:DropDownList ID="CmbProva" runat="server">
        <asp:ListItem>Primo</asp:ListItem>
        <asp:ListItem>Secondo</asp:ListItem>
        <asp:ListItem>Terzo</asp:ListItem>
      </asp:DropDownList></div>
    </form>
  </body>
</html>
```

Lo sviluppatore windows è solito trascinare l'oggetto direttamente dove deve essere inserito. Tuttavia, il comportamento predefinito di Visual Studio 2005, è quello di fare un posizionamento non assoluto. Per cambiare tale opzione è necessario eseguire questo passaggio da menù
Layout -> Position -> Auto-position Options



E dopo aver selezionato il posizionamento assoluto appare la seguente situazione notiamo che nei controlli viene inserita la posizione assoluta del controllo

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="Default2.aspx.vb" Inherits="Default2" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
  <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
```

```

        &nbsp;
        <asp:DropDownList ID="CmbProva" runat="server" style="z-
index: 100; left: 76px; position: absolute; top: 22px">
            <asp:ListItem>Primo</asp:ListItem>
            <asp:ListItem>Secondo</asp:ListItem>
            <asp:ListItem>Terzo</asp:ListItem>
        </asp:DropDownList>
        <asp:Button ID="BtnTest" runat="server" Text="Cliccami"
Width="136px" style="z-index: 102; left: 111px; position: absolute;
top: 91px" />
    </div>
</form>
</body>
</html>

```

Andando poi a generare l'evento del click del pulsante, il codice viene separato in un file a parte. Questa è la sub di gestione dell'evento e il codice che si scrive è normalissimo codice Visual Basic.Net (con alcune ovvie modifiche, una su tutte di esempio, l'assenza di MessageBox)

```

Partial Class Default2
    Inherits System.Web.UI.Page

    Protected Sub BtnTest_Click(ByVal sender As Object, ByVal e _ As
System.EventArgs) Handles BtnTest.Click

Response.Write("Hai scelto " & _ Me.CmbProva.SelectedItem.ToString)

    End Sub
End Class

```

I Controlli di convalida

Un'operazione molto frequente nello sviluppo di siti web consiste nel far compilare moduli per inserimento di dati che potrebbero richiedere la presenza di alcune informazioni obbligatorie oppure costruite in un certo modo.

Anche se non è possibile verificare al 100% la correttezza delle informazioni inserite, ASP.Net prevede una serie di controlli per la validazione dell'input da parte dell'utente.

I controlli di convalida funzionano sempre con un doppio controllo, lato client (attraverso JavaScript) e lato server. Questo per garantire due cose, se lato client i JavaScript sono funzionanti e supportati, l'utente eviterà di inviare una pagina incompleta/errata al server; mentre se lato client questi controlli non fossero possibili, la pagina viene validata dal server e quindi sono comunque certo della correttezza dei dati inseriti.

Questi sono i controlli di convalida (Validation Controls):

Sotto ogni controllo vengono specificate le proprietà da impostare per ottenere il controllo.

RequiredFieldValidator: Assicura che un campo venga valorizzato
ControlToValidate: Indica il nome del controllo da validare (obbligatorio)
ErrorMessage: Indica il testo da riportare nel caso il controllo sia vuoto

RangeValidator: Assicura che, se viene inserito un valore, il valore sia all'interno di un certo intervallo
ControlToValidate: Indica il nome del controllo da validare (obbligatorio)
ErrorMessage: Indica il testo da riportare nel caso il controllo sia vuoto
MaximumValue: Indica il valore massimo che il controllo deve assumere
MinimumValue: Indica il valore minimo che il controllo deve assumere
Type: Indica il tipo di dato da utilizzare per il confronto (numerico, testo, data ecc.)

RegularExpressionValidator: Verifica che i dati all'interno di un controllo, se inseriti, siano conformi ad una particolare specifica (es: indirizzo e-mail)
ControlToValidate: Indica il nome del controllo da validare (obbligatorio)
ErrorMessage: Indica il testo da riportare nel caso il controllo sia vuoto
ValidationExpression: Indica il tipo di controllo che deve essere effettuato, cliccando i "...” viene eseguito un wizard che consente velocemente di impostare alcuni tipi di controlli frequenti.

Questa stringa ad esempio valida un indirizzo IP:

```
\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b
```

Un sito molto utile in questo senso è <http://www.regular-expressions.info>

CompareValidator: Consente di confrontare i dati di un controllo con i dati di un altro controllo (Comodo ad esempio per la digitazione della conferma password)

ControlToValidate: Indica il nome del controllo da validare (obbligatorio)
ErrorMessage: Indica il testo da riportare nel caso il controllo sia vuoto
ControlToCompare: Indica il nome del controllo con cui deve essere effettuato il controllo
Operator: Indica l'operatore di confronto che deve essere utilizzato (maggiore, minore, uguale ecc)
Type: Indica il tipo di dato da utilizzare per il confronto (numerico, testo, data ecc.)

CustomValidator: Posso definire una mia specifica regola di controllo, sia lato server che lato client

Questo controllo viene utilizzato quando il tipo di controllo da effettuare è troppo complesso da implementare con i controlli standard (es: correttezza formale del codice fiscale)

In questo caso posso specificare una funzione di controllo (solitamente in JavaScript) da usare sul client e una da usare lato server (in questo caso scrivendola direttamente in VB.Net piuttosto che C# ecc)

ValidationSummary: Visualizza il riepilogo di tutti gli errori che si sono verificati nella pagina.

Nessuna proprietà particolare, se non la possibilità di far apparire il messaggio sotto forma di MessageBox (visto che manca di default sul codice ^__^)

Il controllo TreeView

Il controllo TreeView è sicuramente molto noto nell'ambiente Windows (basta pensare ad esplora risorse) ed è molto flessibile il suo utilizzo in Asp.Net.

Una volta inserito (è nella serie di controlli "Navigation") possiamo infatti tramite task list caricare un file XML oppure un site map per visualizzare la struttura del sito.

In alternativa, possiamo aggiungere i nodi manualmente tramite la lista dei task con un semplice wizard.

Il controllo MultiView e View

Questi due controlli sono un aggiornamento dei controlli precedentemente forniti (ma non supportati) TabStrip, PageView e la versione precedente di MultiView.

MultiView funziona come contenitore di controlli View, e fornisce il supporto per scorrere i diversi View in esso contenuti visualizzandone sempre uno alla volta.

La proprietà da impostare nell'oggetto MultiView per scorrere avanti e indietro è la proprietà ActiveViewIndex

Le Master Page

Con asp, per cercare di mantenere il layout del sito uguale, si poteva utilizzare la tecnica degli includes, che tuttavia non risultava molto semplice in quanto era abbastanza difficile con numerosi editor visualizzare l'aspetto grafico del sito.

Ora, per definire il layout generale della pagina posso usare appunto le master page, in cui vado a definire le tabelle, e la grafica della mia pagina.

Per dire ad una pagina qual è la sua master si usa la direttiva

```
"@Page MasterPageFile=NomePagina.master"
```

Posso quindi personalizzare le varie pagine inserendo dei "PlaceHolder" all'interno della pagina master che verrà poi ridefinito in ogni pagina secondaria.

Nelle pagine secondarie poi andremo a modificare il Content Placeholder inserendo il contenuto della pagina. Modificando quindi la pagina master tutte le pagine crete sul modello subiranno la modifica di tutte le zone a parte di quanto inserito nel content Placeholder.

In questo modo risulta abbastanza semplice mantenere la grafica di tutto il sito web.

Le Web Part

Le Web Part sono una interessante novità per lo sviluppo Web. Esse infatti sono simili ai controlli personalizzati, esse infatti consentono di modificare il codice HTML senza dover modificare all'interno del codice l'output della pagina.

Derivano come componente da SharePoint, e semplifica molto il processo di creazione e di modifica delle pagine web da parte degli utenti.

Posso con Asp.Net sia utilizzare le Web Part (utilizzando come vedremo un controllo che si chiama WebPartManager), ma posso anche costruire delle mie Web Part, che di fatto sono dei controlli lato server (simili ai controlli personalizzati)

Il WebPartManager è il controllo principale delle web Part, in quanto la sua presenza è richiesta in ogni pagina che utilizza anche solo una Web Part.

Le "Zone Incorporate", un concetto che provo a spiegare in termini semplici, con le web part io posso consentire all'utente di modificarsi il layout del sito senza conoscere l'html. Per farlo devo inserire dei controlli all'interno di varie Zone.

- WebPartZone: Questa è la zona principale e anche quella di base, essa contiene tutti i controlli lato server che voglio inserire inoltre posso inserire in questa zona dei controlli WebPart
- CatalogZone: La catalog zone mi permette di aggiungere ad esempio dei controlli, di visualizzarli o di nasconderli
- EditorZone: Questa zona permette agli utenti di modificare la pagina in base alle loro preferenze, possono scegliere colore e dimensione come anche altezza e larghezza dei vari controlli
- ConnectionZone: Questa zona consente di collegare tra loro Webpart, facendole comunicare in modo dinamico.

Vi sono poi alcuni controlli disponibili che consentono di effettuare altre operazioni molto comode sulle WebPart.

Il PageCatalogPart: che consente di aprire e chiudere i vari controlli WebPart, questo controllo va inserito necessariamente all'interno di una CatalogZone.

Il DeclarativeCatalogpart: che consente di aggiungere delle parti in modo dinamico, ad esempio è possibile aggiungere ad una pagina web una casella di testo, anche questo controllo va inserito all'interno di una CatalogZone

L' AppearanceEditorPart: che consente di modificare le proprietà dell'aspetto di una web part o di un controllo come le dimensioni, il titolo o il colore; questo controllo va inserito all'interno di un EditorZone

Il BehaviorEditorPart consente di cambiare il comportamento di un controllo, posso ad esempio decidere se consentire l'editing di quel controllo, di consentirne la chiusura ecc. va inserito all'interno di una EditorZone

Il LayOutEditorPart consente di modificare il layout delle mie webPart ad esempio cambiarne l'ordinamento, va inserito all'interno di una EditorZone

Le Web Part supportano 5 modalità di visualizzazione

BrowserDisplayMode: Modalità standard, in questa modalità non sono consentite personalizzazioni alle web part

DesignDisplayMode: In questa modalità è possibile modificare il layout mediante drag-drop

EditDisplayMode: In questa modalità è possibile eliminare i componenti che sono stati aggiunti in maniera dinamica. Apparirà infatti una voce "delete" sulle web part eliminabili

ConnectDisplayMode: Un utente può in questa modalità connettere

CatalogDisplayMode: In questa modalità un utente può aggiungere delle Web Part in un controllo WebPartZone in fase di esecuzione.

Non autenticando gli utenti, ogni modifica apportata alle web part, sarà visibile a tutti gli altri utenti.

L'argomento delle Web Part è purtroppo molto vasto e non può essere trattato in così poco tempo, sono molto comode soprattutto in applicazioni di tipo portale vista la loro possibilità di migliorare le possibilità di personalizzazione di un sito mediante asp.Net 2.0

ComboBox e ListBox

Molto spesso nel web si incontrano controlli come combobox e listbox per visualizzare dati come stato, provincia oppure il comune.

In questi casi è comodo sapere che questi controlli possono essere popolati in due modi, caricando i dati con delle istruzioni VB oppure associando un origine dati.

Nel primo caso scriveremo un qualcosa di questo tipo

Che consente di caricare in una lista un oggetto di tipo ArrayList (di fatto andiamo a costruire una classe per poi scegliere quale parte della classe visualizzare e quale parte usare

```
Partial Class TestIList
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.Load
        'la lista è basata su oggetti di tipo IList
        Dim ListaValori As IList = CreaLista()

        'se la pagina viene ricaricata per effetto di un post-
        If Not Me.IsPostBack Then
            'imposto il datasource
            Me.ListBox1.DataSource = ListaValori
            'scelgo di vedere la descrizione
            Me.ListBox1.DataTextField = "Descrizione"
            'scelgo che il nome è il valore che selezionerà il controllo
            Me.ListBox1.DataValueField = "Nome"
            'effettuo il bind del controllo
            Me.DataBind()
        End If
    End Sub

    Protected Function CreaLista() As IList
        Dim MioArray As New ArrayList 'variabile di tipo lista di elementi
        Dim MiaClasse As ClDescrizioneComponente 'variabile di tipo classe che ho
definito

        'creo un'istanza della classe per ogni elemento che aggiungo
        MiaClasse = New ClDescrizioneComponente
        MiaClasse.ImpostaDescrizione("12345", "Massimo Piubelli")
        MioArray.Add(MiaClasse)

        MiaClasse = New ClDescrizioneComponente
        MiaClasse.ImpostaDescrizione("6789", "Luca Rossi")
        MioArray.Add(MiaClasse)

        'ritorno la mia variabile
        Return MioArray
    End Function

    Protected Sub ListBox1_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ListBox1.SelectedIndexChanged
        'si noti come il selectedvalue è non il nome che appare
        Me.LblLista.Text = "Hai selezionato " & Me.ListBox1.SelectedValue

    End Sub
End Class
```

```

'Classe che definisce le caratteristiche del mio nuovo oggetto
Public Class ClDescrizioneComponente
    Private M_Nome As String
    Private M_Descrizione As String

    Public Property Nome() As String
        Get
            Return M_Nome
        End Get
        Set(ByVal value As String)
            M_Nome = value
        End Set
    End Property

    Public Property Descrizione() As String
        Get
            Return M_Descrizione
        End Get
        Set(ByVal value As String)
            M_Descrizione = value
        End Set
    End Property

    Public Sub ImpostaDescrizione(ByVal StrNome As String, ByVal StrDescrizione As
String)
        M_Descrizione = StrDescrizione
        M_Nome = StrNome
    End Sub
End Class

```

Controlli utente (.ascx)

Con questa tecnologia è possibile costruire dei controlli utente in maniera molto semplice perché verrà costruito con la finestra di progettazione di Visual Studio. Possiamo immaginare il controllo utente come una piccola WebForm, da inserire poi all'interno di altre form, anche in questo caso per scrivere meno codice e per mantenere una struttura del sito molto simile.

Un esempio potrebbe essere rappresentato dalla finestra di login, che magari è utilizzata più volte nel sito. Ipotizzando che non mi piaccia l'aspetto del controllo standard, fornito già con Visual Studio, potrei creare il mio controllo e trascinarlo poi dentro nelle varie WebForm.

Bind dei dati... meno codice da scrivere

Nel secondo caso i dati verranno agganciati in maniera del tutto visuale. Impostando la proprietà DataSource possiamo scegliere da dove il controllo preleva i dati, in questo caso l'associazione con i dati è molto più semplice in quanto questi controlli nascondono la complessità delle operazioni sopra descritte e velocizzano sensibilmente lo sviluppo dell'interfaccia, consentendo anche di scegliere il layout della pagina web

Controllo AccessDataSource

Andremo a fare il bind dei controlli tramite questo oggetti, in particolare andremo a collegarci con un database di tipo Microsoft Access, in particolare, il famoso Northwind.

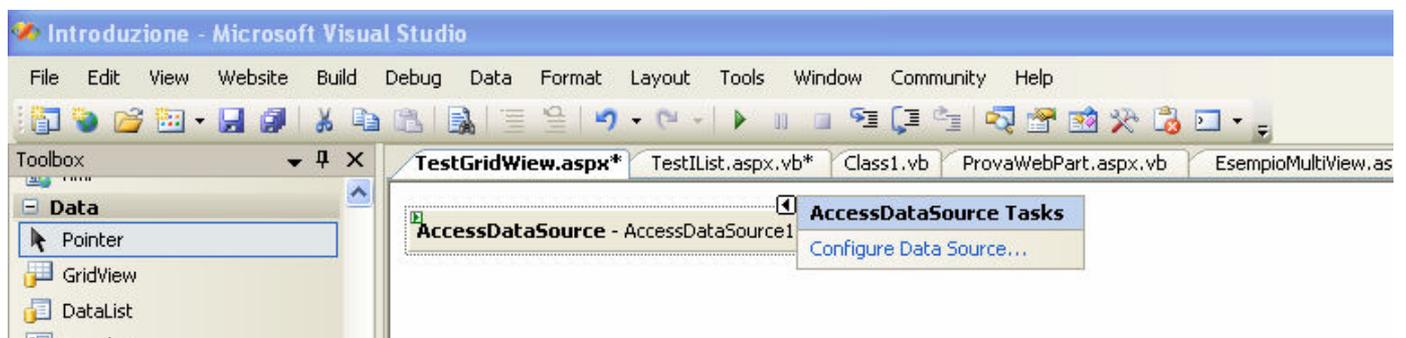
Il bind con altri database è molto simile, quello che potrebbe cambiare è ad esempio l'indicazione di altre informazioni necessarie per altri motori di database (Sql Server ad esempio necessita di sapere se utilizzare l'autenticazione integrata o quella di sql, qual è il database da utilizzare, il nome del server ecc)

Prima di tutto va aggiunto il Database al nostro sito, dal menù "WebSite" scegliere "Add Existing Item", quindi selezionare il file.

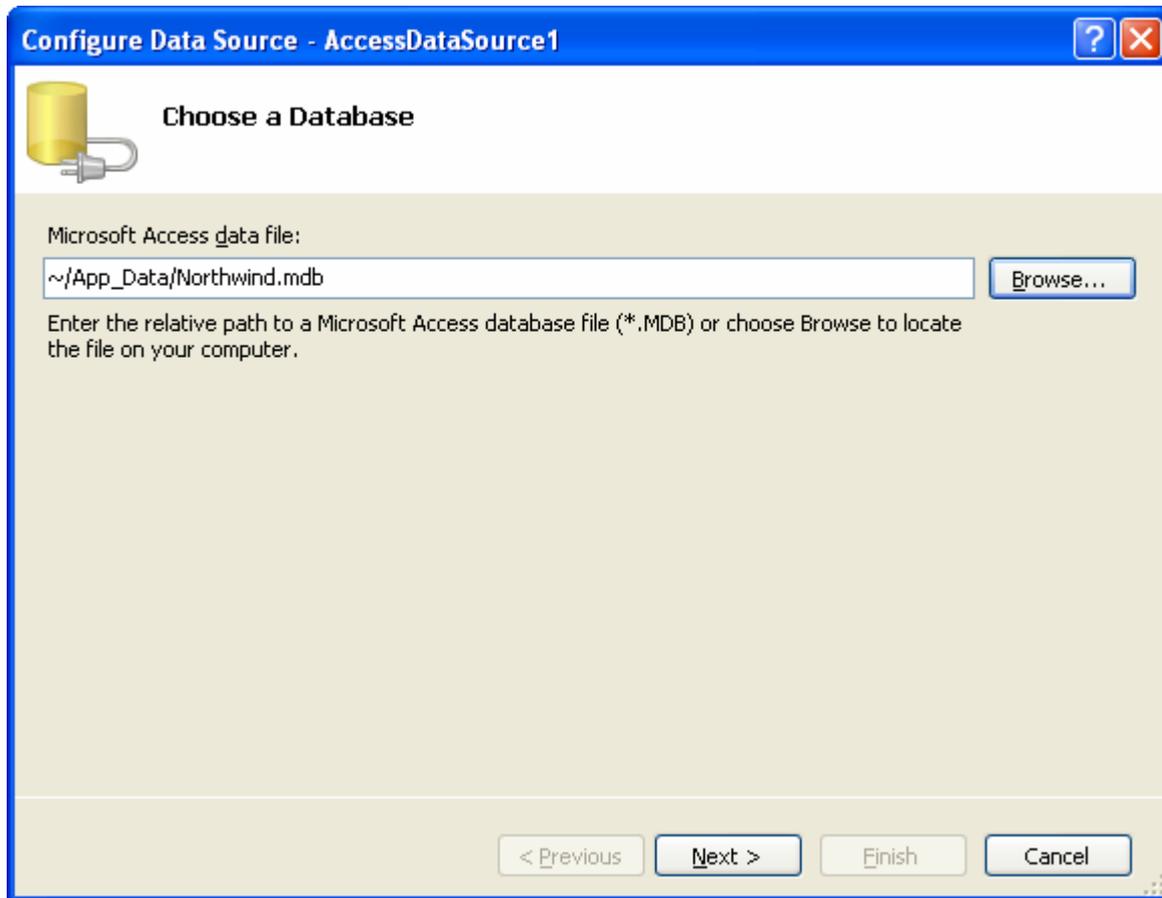
Una volta che il file si trova all'interno del progetto, spostarlo all'interno della cartella App_Data che contiene appunto i database del sito (tra cui quelli di SQL Server)

A questo punto possiamo iniziare. Trascinando un controllo AccessDataSource (che si trova dentro Data) sulla nostra pagina.

Nella finestra AccessDataSource Task clicckiamo su "Configure Data Source"



A questo punto partirà il wizard per la configurazione del nostro data source, come primo passaggio, selezioniamo con il pulsante Browse il file di northwind che si trova all'interno di App_Data

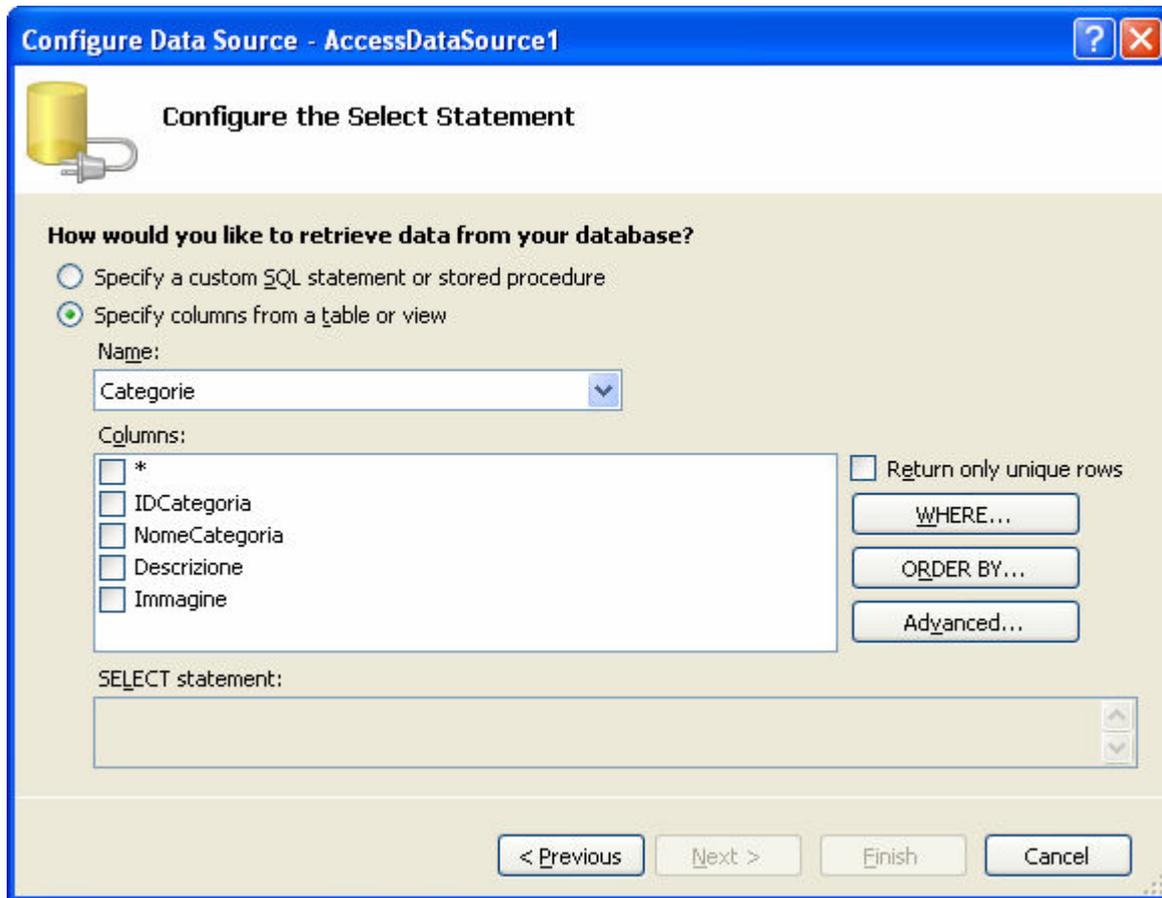


Quindi passiamo alla fase successiva.

In questa fase abbiamo la possibilità di scegliere se caricare i nostri dati da una tabella oppure se costruire un'istruzione sql, per semplicità ci agganceremo alla tabella clienti e ci faremo costruire le operazioni di insert, update e delete direttamente dal Wizard cliccando su "Advanced".

Interessante poi la possibilità di ordinare già i dati in base a tre ordinamenti. Nel nostro caso imposteremo, per nomesocietà e per città.

Il pulsante Where verrà analizzato successivamente.



Nella schermata successiva (che non riporto) possiamo testare la query per vedere un'anteprima dei dati e chiudere quindi il wizard.

Abbiamo pronta la nostra origine dati. Ora dovremo scegliere quale controllo utilizzare per vedere i nostri clienti.

Oggetto GridView

Il primo modo di rappresentazione è il GridView.

Questo controllo è l'evoluzione della vecchia "DataGrid" di asp 1.0. Ma come funzionalità è veramente innovativo. Possiamo infatti vedere come all'interno della griglia lui possa velocemente visualizzare i nostri record, ci consenta di dare una grafica velocemente al nostro controllo.

Ma la parte interessante è che questo controllo gestisce nativamente il paging e l'ordinamento dei record.

Altra funzione molto interessante di questo oggetto è la possibilità di abilitare la modifica dei dati, quindi operazioni di inserimento, modifica e cancellazione di record e tutto SENZA SCRIVERE UNA RIGA DI CODICE!!!! (o quasi)

Andiamo a vedere come in modalità visuale possiamo inserire questo controllo all'interno della nostra pagina.

Una volta trascinato, andiamo a vedere le possibilità che ci vengono offerte dai task di questo controllo.

Come prima cosa si dovrà impostare il DataSource (selezionando l'AccessDataSource1 che avevamo costruito in precedenza)

Quindi, con AutoFormat possiamo velocemente dare il layout al nostro controllo, ma la parte interessante è quando si decide che cosa abilitare.

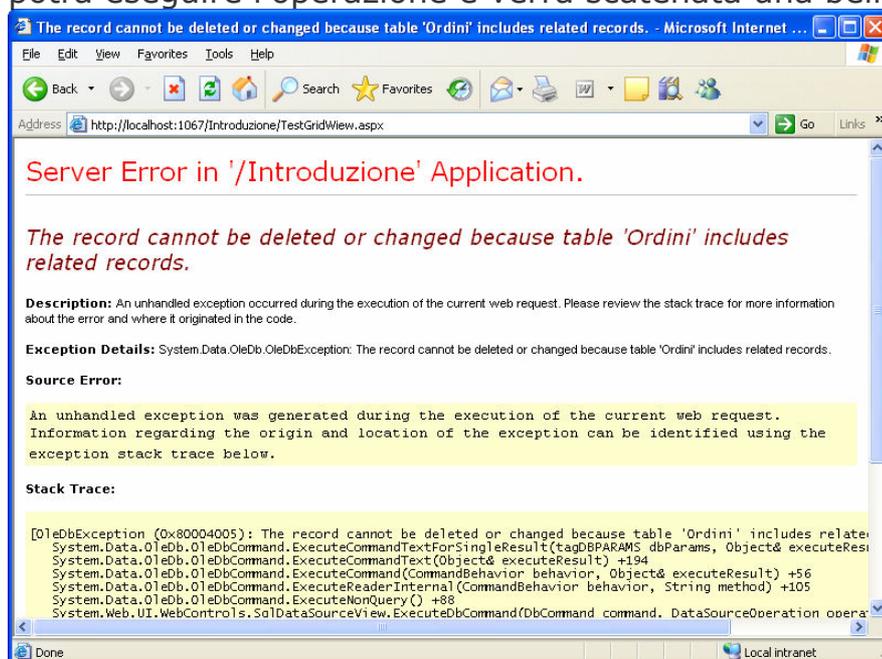
In questo primo esempio, abilitiamo il Paging (lui metterà la divisione di ogni gruppo di n elementi – per modificarla la proprietà è PageSize)

Il Sorting (ogni etichetta dei nomi dei campi diventerà cliccabile e produrrà l'ordinamento crescente/decescente in base al campo cliccato)

L'edit ed il delete pertanto lui inserirà due "link" per consentire la modifica/cancellazione in corrispondenza di ogni record.

Il problema ce lo potremmo avere in questo caso: che succede se si sta cancellando un cliente che ha record di ordini collegati?

Ovviamente, visto il vincolo di integrità referenziale applicato, il database non potrà eseguire l'operazione e verrà scatenata una bella eccezione sulla pagina



La descrizione dell'eccezione è molto esauriente, il record non può essere cancellato perché vi sono ordini collegati. Dobbiamo quindi intervenire, e la soluzione in questo caso è di fare i "programmatori" ovvero scrivere un pochino di codice per gestire l'evento RowDeleting della nostra griglia (evento che succede appunto prima che la riga sia cancellata, e nel quale possiamo annullare l'operazione se il cliente abbia ordini in corso.

Da notare in questo caso che le righe che andremo a scriver dovranno spostare in basso la nostra pagina

```

Protected Sub GridView1_RowDeleting(ByVal sender As Object, ByVal e As
System.Web.UI.WebControls.GridViewDeleteEventArgs) Handles GridView1.RowDeleting

    Dim CodiceCliente As String 'recuperare l'id del cliente
    Dim Cnn As New System.Data.OleDb.OleDbConnection 'variabile connection
    Dim contarighe As Integer 'numero di righe trovate
    Dim cmdcontarighe As New OleDbCommand 'oggetto command per contare le
righe
    Dim cmdordini As New OleDbCommand 'oggetto command per caricare gli ordini
del cliente
    Dim drcaricaordini As OleDbDataReader 'oggetto data reader per leggere gli
ordini del cliente

    'recupero l'id del cliente dalla mia gridview (e.RowIndex corrisponde
all'indice della riga su cui si è cliccato)
    CodiceCliente = GridView1.DataKeys(e.RowIndex).Value

    'imposto una connessione via codice con il database
    'interessante il server.MapPath che consente di recuperare il percorso
assoluto del file "C:\documents and settings\max\Desktop\Cisa 2\..."
    'una volta pubblicato non dovrò cambiare l'istruzione
    Cnn.ConnectionString = "provider = microsoft.jet.oledb.4.0; data source =
" & Server.MapPath("app_data\Northwind.mdb")

    'apro la connessione
    Cnn.Open()

    'imposto le proprietà connection e commandtext all'oggetto command che
conta il numero di ordini
    cmdcontarighe.Connection = Cnn
    cmdcontarighe.CommandText = "select count(*) from ordini where idcliente =
'" & CodiceCliente & "'"

    'il metodo execute scalar ritorna la prima riga e la prima colonna della
mia select
    'va particolarmente bene quando uso, come in questo caso, funzioni di
aggregazione
    contarighe = cmdcontarighe.ExecuteScalar

    'se trovo delle righe
    If contarighe > 0 Then
        'prima cosa, scrivo nella pagina corrente che non può essere
cancellato
        Response.Write("Il cliente non può essere cancellato perchè ha " &
contarighe & " ordini in corso")

        'imposto il nuovo command che mi carica i dati degli ordini
        cmdordini.Connection = Cnn
        cmdordini.CommandText = "select idordine, dataordine from ordini where
idcliente = '" & CodiceCliente & "'"

        'imposto il datareader, in questo caso il metodo executereader ritorna
esattamente un datareader
        'oggetto di cui tra l'altro non avevo (e non potevo) crearne l'istanza
        drcaricaordini = cmdordini.ExecuteReader

        'ciclo su tutte le righe
        'a differenza dei "cursors" di ado/dao, in questo tipo di cursore non
uso il movenext
        'in quanto lo stesso metodo read legge e fa uscire dal ciclo quando
non trova più righe

```

```

Do While drcaricaordini.Read
    Response.Write("<br/>")
    Response.Write("ordine numero : " &
drcaricaordini.Item("idordine"))
    Response.Write(" del : " & drcaricaordini.Item("dataordine"))
Loop

'questa è la parte principale di tutta la cosa.
'Gli dico di annullare l'operazione che ha fatto l'utente (click sul
pulsante elimina)
e.Cancel = True

'una parte molto importante è quella di chiudere il datareader, in
quanto altrimenti potrebbe non consentire ulteriori
operazioni sulla connessione.
drcaricaordini.Close()

'distruggo poi i vari oggetti
drcaricaordini = Nothing
cmdordini = Nothing
End If

cmdcontarighe = Nothing
Cnn = Nothing

End Sub

```

Oggetto FormView

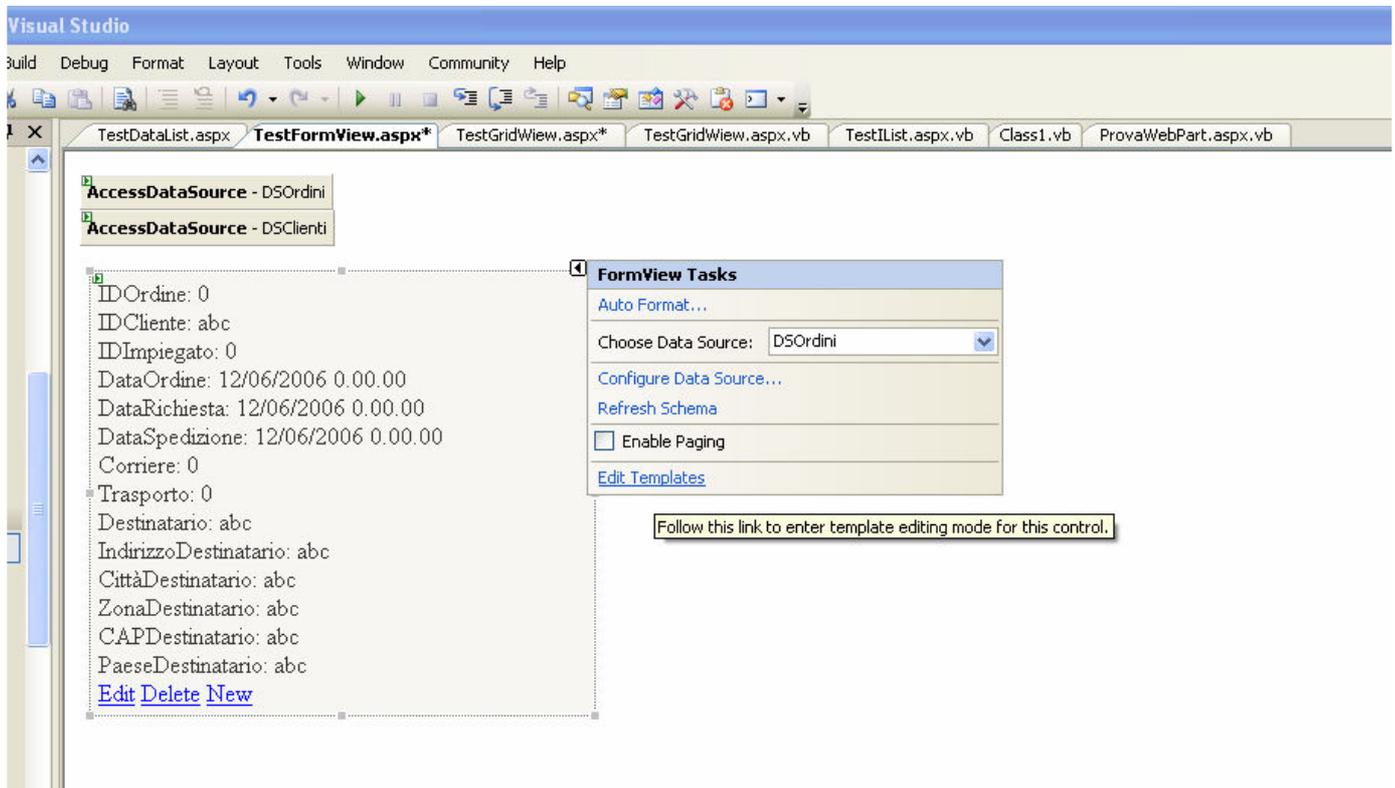
Molto simile all'oggetto DataGrid con una differenza che i controlli sono liberi, ovvero posso editare i vari "templates" per scegliere con che oggetti verranno modificati i dati (ad esempio posso scegliere di usare una textbox piuttosto che una comboBox)

I passaggi sono molto simili, anche in questo caso trasciniamo il controllo e lo agganciamo al nostro DataSource.

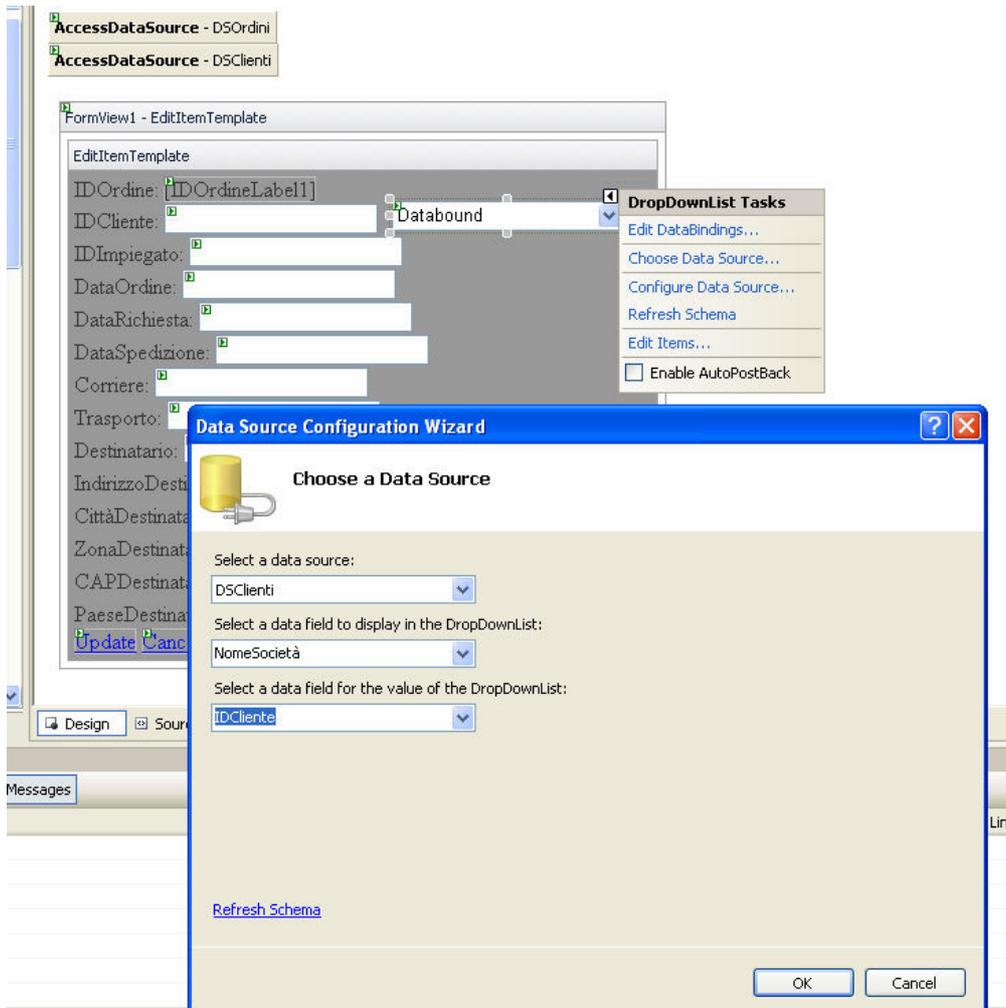
Nell'esempio che propongo farò un bind del controllo ad una combo in "stile Access" prendiamo come base la tabella "Ordini", quindi vorrei scegliere il cliente tramite una comboBox. In access, non facevo altro che costruire una ulteriore query sulla Combo, e scegliere la colonna associata per poi nascondersela.

Qui la procedura è molto simile, solo che mi serve un altro AccessDataSource (che in questo caso non aggiornerà i dati) che caricherà IDCliente e NomeSocietà per la combo.

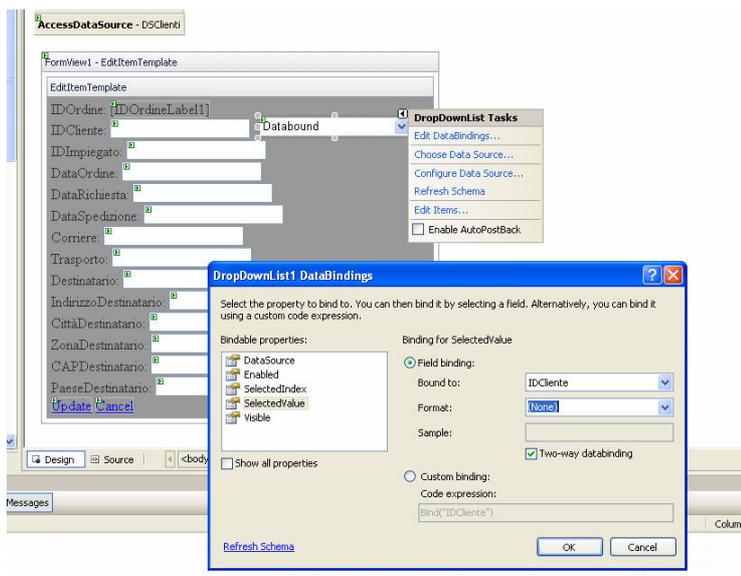
In questo caso è opportuno anche dare dei nomi "intelligenti" ai vari dataSource, li chiamerò DSClienti e DSOrdini.



Una volta agganciato il DataSource, selezioniamo la voce Edi Templates. Quindi sul nuovo task che appare decidiamo di modificare il template di modifica dei dati, per sostituire alla casella di testo la combo, selezioniamo quindi EditItemTemplate. Trasciniamo una DropDownList all'interno della zona della nostra FormView. Andiamo quindi ad effettuare due operazioni, la prima scegliamo il DataSource. Agganciandolo al nostro dataste DSClienti, andiamo poi ad impostare il campo da utilizzare come valore e quello da visualizzare (Se ricordiamo, era la stessa impostazione che avevamo utilizzato prima quando avevamo caricato i dati all'interno della DropDown via codice)



Ci resta un ultimo passaggio, il controllo infatti risulta "Unbound" quindi non associato. Selezioniamo il task Edit DataBindings per collegare la nostra DropDownList al campo idCliente.



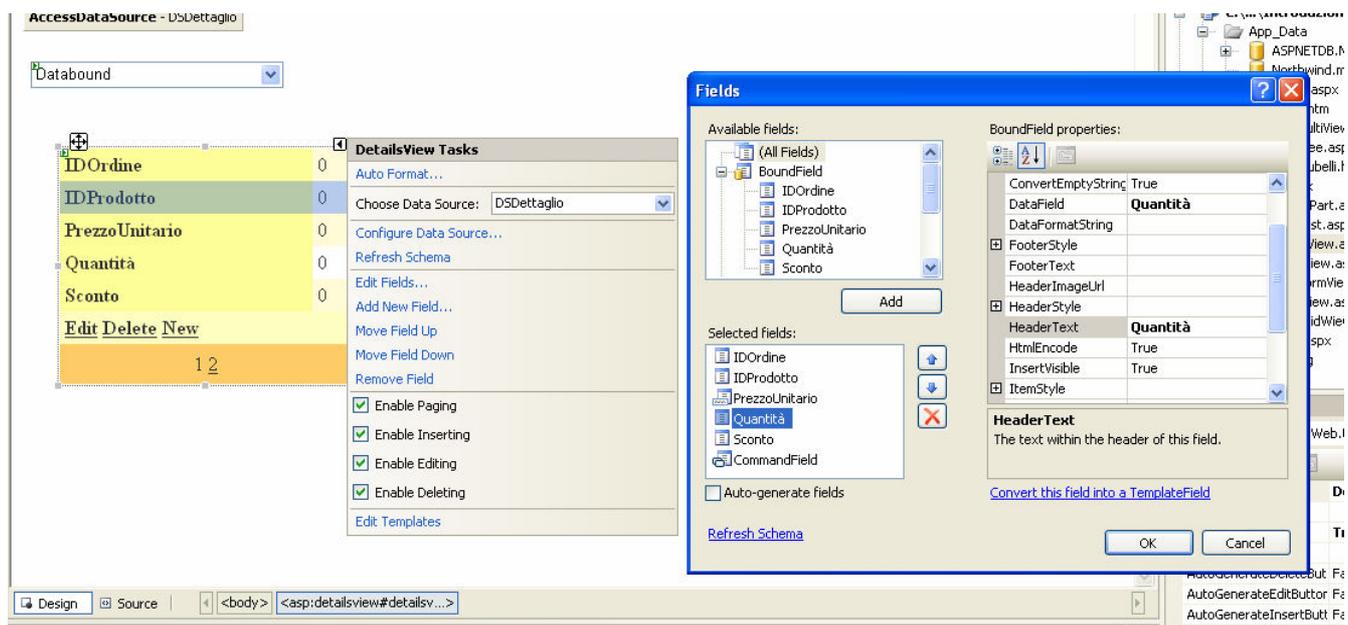
Abbiamo finito, chiudiamo con l'edit del template scegliendo end Template editino dal task principale del nostro formView

Oggetto DetailView

Il DetailView è molto simile al GridView, solo che rappresenta un record alla volta.

In questo caso inoltre è possibile andare a modificare i vari templates, scegliendo il comportamento che dovrà assumere anche in questo caso in fase di modifica o di inserimento, tuttavia il passaggio è un po' più lungo. Si deve infatti passare attraverso la conversione del campo in un TemplateField.

A questo punto posso intervenire sulla modifica di ogni singola parte di campo.



Oggetto ListView

Molto simile come logica alla DataGrid, ma i dati vengono visualizzati in un'unica pagina ripetendo i dati continuando a ripetere uno stesso modello.

In questo controllo la proprietà più interessante a mio avviso è la possibilità di intervenire sul PropertyBuilder per scegliere ad esempio il numero di colonne da utilizzare per la ripetizione dei dati.

Filtrare i dati

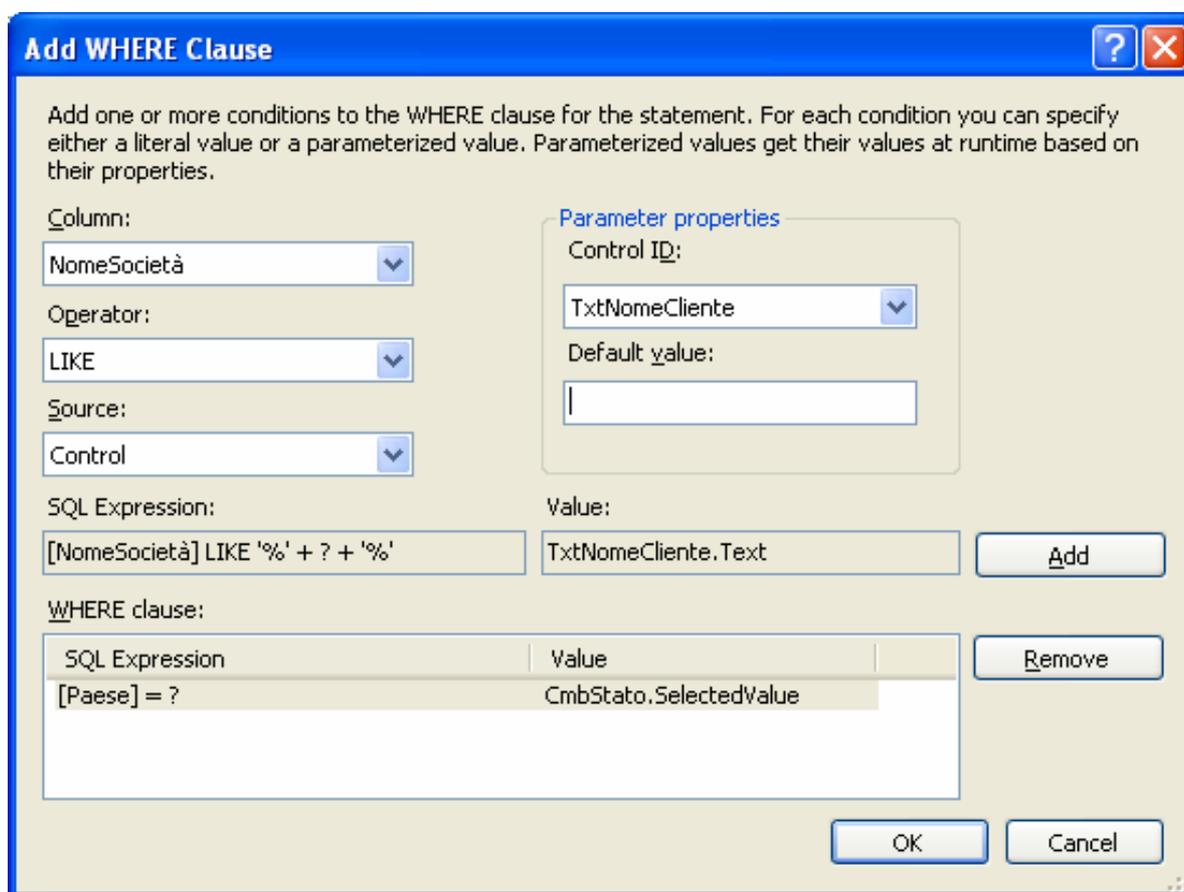
Un'altra cosa che viene fatta molto spesso è una maschera di ricerca, quante volte infatti nelle nostre applicazioni abbiamo cercato di vedere i dati dei clienti che hanno un certo nome e magari sono di una certo stato?

La struttura del NorthWind, non dispone di una tabella degli stati, ma nell'esempio cercherò di fare una ricerca di clienti sulla base di una parte della ragione sociale (nomesocietà) e dello stato. Con la possibilità di filtrare sulla selezione di uno, dell'altro oppure di entrambi.

Quindi inseriamo il primo AccessDataSource (che servirà per caricare gli stati) con una relativa DropDownList. Quindi una casella di testo in cui l'utente scriverà una parte della ragione sociale.

Nel secondo AccessDataSource, andiamo a cliccare ora su Where (quello che prima avevo rimandato)

In questo caso (come si può notare nella parte bassa WHERE clause) ho già inserito il primo criterio e sto impostando il secondo, in cui vado a dire che la colonna nomeSocietà deve essere simile a quanto contenuto nel controllo TxtNomeCliente. Notiamo come lui nello scrivere l'istruzione inserisca già correttamente gli apici e il simbolo % (che corrisponde al * che noi abitualmente utilizziamo in Access)



Confermiamo quindi l'inserimento con il pulsante add e poi confermiamo il tutto con l'OK.

Effettuare quindi il bind come al solito sul controllo.