

إهداء للأخ محمد خليل أبو صلاح وكل أعضاء المنتدى وأشكره على مجهوده الكبير في هذا الدرس

التعامل مع أداة الـ **DataList** و **DataTable**
كل ما يخص هذه الأدوات تقريبا

أداة الـ **DataList** من الأدوات المهمة جدا في تطبيقات الـ ASP.NET والتي تقوم بعرض البيانات للزائر بعدة أشكال ممكنة من خلال إمكاناتها المتاحة سواء في التصميم أو في دعمها للأحداث و إحتواء أدوات أخرى. الدرس سيكون طويلا جدا لما يحتويه من تفرعات و تشعبات ولكن سأحاول توصيله بأسهل طريقة ممكنة كي يبقى عالقا في الأذهان لأنه درس كما قلت مهم جدا. في البداية الـ **DataList** تدعم إستخدام القوالب أو ما يعرف في الويب بالـ **Templates** و تقريبا تشبه ما ذكرناه في موضوع الـ **Repeater** و لكن بالأصح هي أقرب لما تدعمه أداة الـ **DataGrid** حيث تملك نفس القوالب.

القوالب الموجودة داخل الـ **DataList** :

١. **:HeaderTemplate**

تحدد كيف سيظهر رأس الـ **DataList** أي المنطقة العلوية للعرض.

٢. **:ItemTemplate**

و هي المنطقة الأكثر حيوية حيث يتم تنسيق البيانات التي ستعرض فيها.

٣. **:AlternateItemTemplate**

يتم فيها تنسيق العناصر التي تظهر مرة بعد مرة خلال عرض سجلات البيانات.

٤. **:SeparatorTemplate**

و هي المنطقة الفاصلة بين السجلات المعروضة و قلنا بالعادة نستخدم فيها الـ **
**.

٥. **:FooterTemplate**

و فيها يتم تنسيق ذيل الـ **Data List** و إعداده إن استلزم الأمر.

٦. **:SelectedItemTemplate**

تتحكم في تنسيق العنصر الذي يتم إختياره.

٧. **:EditItemTemplate**

تقوم بتنسيق العنصر الذي تم إختياره لإجراء التعديل عليه بشكل معين.

ربط الـ **DataList** بالبيانات :

نفس طريقة الـ **DataBinding** التي ذكرناها في الـ **Repeater** تستخدم هنا سواء إستخرجنا البيانات عن طريق **DataReader** أو **DataSet** و هذا مثال يوضح ذلك :

كود

```
<%@ Page Language="C#" Debug="True" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">
void Page_Load()
{
string prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + Server.MapPath("db.mdb");
string SQL="SELECT * FROM Categories";
OleDbConnection cn = new OleDbConnection(prov);
OleDbCommand cmd = new OleDbCommand(SQL,cn);
```

```

cn.Open();
OleDbDataReader reader = cmd.ExecuteReader();

dl.DataSource= reader;
dl.DataBind();

cn.Close();

}

</script>
<html>
<head>
</head>
<body>
<form runat="server">
<asp: DataList id="dl" runat="server">
<ItemTemplate>
<%# DataBinder.Eval(Container.DataItem,"category")%>
</ItemTemplate>
</asp: DataList>
</form>
</body>
</html>

```

لننظر بداية إلى جزء ال HTML حيث أدرجنا أداة ال **DataList** :

كود

```

<asp: DataList id="dl" runat="server">
</asp: DataList>

```

عن طريق الأداة Categories و أضفنا أكواد من أجل الإتصال بقاعدة البيانات و إستخراج السجلات من جدول Page_Load ثم ذهبنا للحدث DataReader.

بالبيانات من خلال التالي dl: و التي أسميتها DataList بعد ذلك قمنا بربط الأداة

كود

```

dl.DataSource= reader;
dl.DataBind();

```

و من أجل عرض أسماء التصنيفات من الجدول نحتاج لكتابة التالي :

كود

```

<%# DataBinder.Eval(Container.DataItem,"category")%>

```

و طبعا يجب أن يكون داخل ال ItemTemplate لنصل على النتيجة المطلوبة و هي إظهار جميع أسماء التصنيفات الموجودة داخل الجدول

لنفرض أننا نود تغيير ترتيب الظهور و إستخدام دالة نمرر لها قيم معينة و نضيف وصلة تشعبية HyperLink فماذا نفعل ؟

دعونا نرى هذا المثال :

كود

```
<%@ Page Language="C#" Debug="True" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

public object IsActive(object vl)
{
string ac="";
if(vl.ToString()=="True")
ac="Yes";
else
ac="No";
return ac;
}

void Page_Load()
{
string prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + Server.MapPath("db.mdb");
string SQL="SELECT * FROM Categories";
OleDbConnection cn = new OleDbConnection(prov);
OleDbCommand cmd = new OleDbCommand(SQL,cn);

cn.Open();
OleDbDataReader reader = cmd.ExecuteReader();
dl.DataSource= reader;
dl.DataBind();
cn.Close();
}

</script>
<html>
<head>
</head>
<body>
<form runat="server">
```

```

<asp: DataList id="dl" runat="server" RepeatDirection="Horizontal" repeaterlayout="table"
borderwidth="3" gridlines="both" width="100%">
<ItemTemplate>
<table border="0" width="100%">
<tr>
<td>
<%# DataBinder.Eval(Container.DataItem,"category")%> </td>
</tr>
<tr>
<td>
Active(Yes/No) : <%# IsActive(DataBinder.Eval(Container.DataItem,"active"))%>
<br /> </td> </tr>
<tr>
<td>
<a href='./more.aspx?id=<%# DataBinder.Eval(Container.DataItem,"id")%>'>more</a> </td> </tr>
</table>
</ItemTemplate>
</asp: DataList>
</form>
</body>
</html>

```

في منطقة الأكواد نرى أننا أضفنا دالة أسميناها IsActive من نوع كائن Object نمرر لها باراميتر يقوم بفحصه فإن كانت قيمته True أظهر الناتج Yes و إلا سيظهر لنا كلمة No
بعد ذلك تأتي أكواد الإتصال في منطقة حدث تحميل الصفحة Page_Load و ربط ال DataList و نلاحظ أننا غيرنا خصائص ال DataList كالتالي :

كود

```

<asp: DataList id="dl" runat="server" RepeatDirection="Horizontal" repeaterlayout="table"
borderwidth="3" gridlines="both" width="100%">

```

أيضا أعود للدالة و نرى كيف تم إستدعائها مع تمرير باراميتر :

كود

```

<%# IsActive(DataBinder.Eval(Container.DataItem,"active"))%>

```

نرى دائما عند محاولة الكتابة للعناصر التي تخضع لعملية ال Data Binding في منطقة ال HTML نستخدم معها علامة الموجه Directive و هي الإشارة # ثم نقوم بكتابة البيانات أو إستدعاء أي دالة.

من أهم الخصائص كانت ال RepeatDirection جعلناها Horizontal بدل Vertical التي تكون Default

إستخدام الأنماط Styles مع ال DataList :

BackColor

يتحكم في لون خلفية الكلام الذي سيظهر داخل العنصر.

BorderColor

يظهر لون الحدود الخاصة بالعناصر التي ستظهر.

BorderStyle

نمط أو شكل الحدود التي ستظهر به و هناك عدة قيم بالعادة يتم إستخدامها و هي :

(Solid Dashed-Dotted-Double-Groove-Inset-None-NotSet-OutSet-Ridge and)

BorderWidth

تحدد حجم الحدود التي سنعرض بها العناصر.

CssClass

تحدد نوع الفئة التي سيرتبط بها العنصر من خلال ملف CSS يحوي عدة أنماط على شكل فئات.

Font

تحديد نوع الخط.

ForeColor

لون الخط الذي سنكتب به.

Height

إرتفاع العنصر الذي سيتم عرضه.

HorizontalAlign

تتحكم في الإرتفاع الأفقي للعنصر و القيم المعتاد كتابتها هي :

(Right Center-Justify-Left-NotSet and)

VerticalAlign

تحديد المحاذاة الأفقية للعنصر و القيم المعتاد إستخدامها هي :

(Bottom-Middle-NotSet and Top)

Width

عرض العنصر الذي سيعرض.

كانت هذه معظم الأنماط المستخدمة مع ال DataList.

لنأخذ مثالين صغيرين يقومان بعمل تنسيقات معينة للعناصر:

كود

```
<asp: DataList
ItemStyle-Font-Italic="True"
ItemStyle-Font-Bold="True"
ItemStyle-Font-Name="Tahoma"
Runat="server"
>
or
<asp: DataList
AlternatingItemStyle-ForeColor="Blue"
AlternatingItemStyle-Font-Name="Arial"
AlternatingItemStyle-Font-Size="14pt"
Runat="server"
>
```

و هكذا رأينا كيف نضيف الأنماط Styles إلى ال DataList كي تظهر لنا البيانات بالشكل الذي نرغب به.

أحداث ال DataList :

١.CancelCommand:

من أجل تفعيل هذا الحدث نقوم بإضافة الخاصية CommandName للزر و نجعل قيمته cancel.

٢.DeleteCommand:

نقوم بجعل الخاصية CommandName تساوي delete.

٣.EditCommand:

نجعل قيمة ال CommandName هي edit.

٤.UpdateCommand:

نجعل قيمة ال CommandName هي update.

٥.ItemCommand:

يتم تفعيل هذه الخاصية عند وجود حدث غير ما ذكرته بالأعلى من حذف و تعديل و إلغاء و إدخال.

و لكن ما هي ال CommandName أو بشكل آخر أين نكتب؟؟

الجواب سهل جدا و هي خاصية تضاف لأحد الأدوات التالية :

Button

LinkButton

ImageButton

نضيف لهم الخاصية CommandName بحيث عند الضغط على أحدهم يتم تنفيذ الحدث المعين مع العلم أنه يجب إضافة الحدث لل DataList و هذا سيتضح لنا من خلال الأمثلة القادمة.

لنأخذ مثالا نستخدم فيه أغلب الأحداث بشكل بسيط بحيث نحتاج إلى DataList و Label تقوم بكتابة جملة تدل على الحدث الذي اخترناه :

كود

```
<%@ Page Language="C#" Debug="True" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">

void dataBind(){
string prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + Server.MapPath("db.mdb");
string SQL="SELECT * FROM Categories";
OleDbConnection cn = new OleDbConnection(prov);
OleDbCommand cmd = new OleDbCommand(SQL,cn);

cn.Open();
OleDbDataReader reader = cmd.ExecuteReader();
dl.DataSource= reader;
dl.DataBind();
```

```

cn.Close();
}
void Page_Load()
{
if(!Page.IsPostBack) {
dataBind();}
}

void dl_ItemCommand(Object source, DataListCommandEventArgs e) {
msg.Text="<li>Item Command!";}
void dl_EditCommand(Object source, DataListCommandEventArgs e) {
msg.Text="<li>Editing Item!";}
void dl_DeleteCommand(Object source, DataListCommandEventArgs e) {
msg.Text="<li>Deleting Item!";}

void dl_UpdateCommand(Object source, DataListCommandEventArgs e) {
msg.Text="<li>Updating Item!";}

</script>
<html>
<head>
</head>
<body>
<form runat="server">
<p>
<asp : DataList id="dl" runat="server" OnUpdateCommand="dl_UpdateCommand"
OnDeleteCommand="dl_DeleteCommand" OnEditCommand="dl_EditCommand"
OnItemCommand="dl_ItemCommand" repeatColumns="1" repeaterlayout="table" ItemStyle-Font-
Name="tahoma" borderwidth="1" gridlines="both" width="33%">
<ItemStyle font-names="tahoma"></ItemStyle>
<ItemTemplate>
<table border="0" width="100%">
<tr>
<td>
<%# DataBinder.Eval(Container.DataItem,"category")%> <br />
<asp:LinkButton runat="server" Text="Edit!" CommandName="edit" />
<asp:LinkButton Text="Delete!" CommandName="delete" runat="server" />
<asp:LinkButton Text="Update!" CommandName="Update" runat="server"
/></td></tr></table></ItemTemplate>
</asp : DataList>

```

</p>

<asp:Label id="msg" runat="server" text=""></asp:Label></form></body></html>

نبدأ في تحليل المكتوب في المثال..

في الأعلى نرى أكواد الإتصال بقاعدة البيانات و هذه مألوفة لنا و بعدها هناك بعد الأحداث نعود لها بعد قليل.
نذهب لجزء ال HTML حيث الأدوات و قلنا أننا نحتاج ل Label نسميه msg و DataList نسميها dl كالتالي :

كود

```
<asp : DataList id="dl" runat="server" OnUpdateCommand="dl_UpdateCommand"
OnDeleteCommand="dl_DeleteCommand" OnEditCommand="dl_EditCommand"
OnItemCommand="dl_ItemCommand" repeatColumns="1" repeaterlayout="table" ItemStyle-Font-
Name="tahoma" borderwidth="1" gridlines="both" width="33%">
```

نرى الخصائص التي أضفناها و الأحداث التي عرفناها لتكون متصلة بال DataList مثلا لإضافة الحدث الخاص بالحدث نضيف هذه الخاصية :

كود

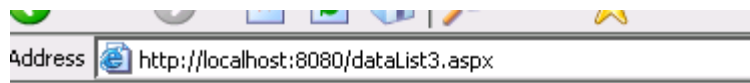
```
OnDeleteCommand="dl_DeleteCommand"
```

قمنا بعد ذلك بتحديد ال ItemStyle من أجل تنسيق النمط الخاص بالمخرجات و أكتفيت فقط بجعل الخط Tahoma و من ثم أضفنا ال ItemTemplate كما هو موضح هنا :

كود

```
<ItemTemplate>
<table border="0" width="100%">
<tr>
<td>
<%# DataBinder.Eval(Container.DataItem,"category")%>
<br />
<asp:LinkButton runat="server" Text="Edit!" CommandName="edit" />
<asp:LinkButton Text="Delete!" CommandName="delete" runat="server" />
<asp:LinkButton Text="Update!" CommandName="Update" runat="server" />
</td>
</tr>
</table>
</ItemTemplate>
```

طبعا أضفنا جدولا من أجل تنسيق المخرجات في كل صف سيظهر لنا إسم التصنيف Category و تحته ٣ أزرار من نوع LinkButton و كل واحد منها أضفنا له الخاصية CommandName و أسندنا لها القيمة المطلوبة سواء حذف أو تعديل أو إدخال كما هو موضح بهذا الشكل :



ASP	Edit! Delete! Update!
VB.NET	Edit! Delete! Update!
C#.NET	Edit! Delete! Update!
ADO.NET	Edit! Delete! Update!
ASP.NET	Edit! Delete! Update!

- Deleting Item!

طبعا سنظهر عدة بيانات أو أقسام على حسب السجلات الموجودة في جدول البيانات و مع كل سجل سنظهر ٣ أزرار كل منها عبارة عن زر يقوم بإستدعاء حدث معين و لنفرض أننا ضغطنا الزر الذي يظهر لنا الكلمة Delete! فماذا سيحدث؟ سيتم إستدعاء الحدث dl_DeleteCommand حسب ما أضفنا لخصائص ال DataList و دعونا نعود لمنطقة الأكواد و لنرى هذا الحدث ماذا يحتوي :

كود

```
void dl_DeleteCommand(Object source, DataListCommandEventArgs e) {
    msg.Text="<li>Deleting Item!";}
```

فقط أضفنا تعليمة تقوم بجعل ال Label يقوم بعرض جملة مكتوب بها Deleting Item! كي نرى ما هو الزر الذي ضغطنا عليه و لكن هي كانت فكرة كي نرى كيف يتم إستدعاء الأحداث و لعل الأمثلة القادمة تزيد الإصارة و التعرف أكثر على الأحداث.

لنأخذ مثالا نوضح فيه تعديل البيانات في ال DataList و أتمنى أن تنتبهوا له جيدا لأنه مثال ممتاز نوعا ما. لنبدأ في تحليل الفكرة:

أولا سنظهر التصنيفات الموجودة في جدول categories.

ثانيا نضيف زر للتعديل مع كل تصنيف يقوم بحدث التعديل.

هذا الشكل يوضح أول مرحلتين :

ASP
Edit!
VB.NET
Edit!
C#.NET
Edit!
ADO.NET
Edit!
ASP.NET
Edit!

ثالثا عند الضغط على زر نقوم بإظهار إسم التصنيف داخل مربع نص Text Box و بجانبه زر لتأكيد التعديل.

ASP
Edit!
VB.NET
Edit!
C#.NET Update!
ADO.NET
Edit!
ASP.NET
Edit!

رابعا بعد الضغط على زر التعديل تنتهي المهمة و تعود الصفحة للعرض كما كانت بعد حفظ التعديلات.
لتنفيذ أول خطوة نحتاج لـ DataList تعرض البيانات عن طريق الـ ItemTemplate كما عرفنا سابقا و نستخدم أيضا زر من نوع LinkButton و نعين الـ CommandName إلى Edit و هذا يتضمن ثاني مرحلة أيضا.

أما ثالث مرحلة فسنحتاج لإضافة الوسم EditItemTemplate كي تظهر بدل الـ ItemTemplate عند السجل الذي إختارناه و هي تحوي مربع نص Text Box و زر لتأكيد التعديل من نوع Link Button..

و المرحلة الرابعة و الأخيرة هي مرحلة إستدعاء الحدث الخاص بالتعديل و عمل بعض التعليمات كي نقوم بحفظ التعديلات في قاعدة البيانات و إعادة ربط الأداة بالبيانات بعد التعديل عليها و ظهورها بالشكل الطبيعي أي قبل ضغط زر التعديل أو التحرير بالأصح.

سوف أكتب الكود هنا ثم أبدأ في شرحه جزءا جزءا كي نستوعب الأمور بشكل أوضح و على فكرة إستخدام هذه الأدوات يعطينا مرونة و سرعة أكبر من إستخدام الطرق الشبيهة بالـ ASP الكلاسيكية القديمة و التي للأسف يقوم العديد من المبرمجين بإستخدامها لليوم مع هذه التقنية متجاهلين إمكانيات الأدوات الجديدة و التي توفر العديد من الجهد و الوقت و كل ما نحتاجه فقط القليل من التدريب و التواصل معها كي نصل لمرحلة نستطيع تقدير الخدمات التي توفرها لنا هذه الأدوات الرائعة والان نعود للكود :

```
<%@ Page Language="vb" Debug="true" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.OleDb" %>
<script runat="server">
```

```
Sub DataBind()
Dim prov As String
Dim SQL As String
Dim cn As OleDbConnection
Dim reader As OleDbDataReader
Dim cmd As OleDbCommand

prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" & Server.MapPath("db.mdb")
SQL="SELECT * FROM Categories"
cn = new OleDbConnection(prov)
cmd = new OleDbCommand(SQL,cn)

cn.Open()
reader = cmd.ExecuteReader()
dl.DataSource=reader
dl.DataBind()
reader.Close()
cn.Close
End Sub

Sub Page_load
If Not Page.IsPostBack Then
DataBind()
End If
End Sub

Sub dl_EditCommand(source As Object, e As DataListCommandEventArgs)
dl.EditItemIndex=e.Item.ItemIndex
DataBind()
End Sub
```

```

Sub dl_UpdateCommand(source As Object, e As DataListCommandEventArgs)
Dim newCatName As TextBox
Dim catID As String
Dim prov As String
Dim SQL As String
Dim cn As OleDbConnection
Dim reader As OleDbDataReader
Dim cmd As OleDbCommand

prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" & Server.MapPath("db.mdb")
SQL="Update Categories set Category=@newCatName where id=@catID"

cn = new OleDbConnection(prov)
cmd = new OleDbCommand(SQL,cn)

newCatName=e.Item.FindControl("newCatName")
catID=dl.DataKeys(e.Item.ItemIndex)

cmd.Parameters.Add("@newCatName",newCatName.Text)
cmd.Parameters.Add("@catID",catID)

cn.Open()
cmd.ExecuteNonQuery()
cn.Close()

dl.EditItemIndex=-1
DataBind()
End Sub

```

```

</script><html><head></head>
<body><form runat="server">
<asp : DataList id="dl" runat="server" DataKeyField="id" OnUpdateCommand="dl_UpdateCommand"
width="33%" OnEditCommand="dl_EditCommand" border="2">
<ItemTemplate>
<%# DataBinder.Eval(Container.DataItem,"category") %>
<br />
<asp:LinkButton id="edit" commandName="edit" text="Edit!" runat="server" />
</ItemTemplate>
<EditItemTemplate>

```

```

<p>
<asp:TextBox id="newCatName" runat="Server" Text='<%#
DataBinder.Eval(Container.DataItem,"category") %>' />

<asp:LinkButton id="update" commandName="update" text="Update!" runat="server" />
</p>
</EditItemTemplate>
</asp : DataList>
</form>
</body>
</html>

```

يوجد عندنا أحداث و إجراءات و هي :

كود

```

DataBind
Page_Load
dl_EditCommand
dl_UpdateCommand

```

و هذه في منطقة الكود و الأولى أنا وضعت فيها الكود الخاص بربط الأداة DataList مع جدول التصنيفات و الحدث Page_Load هو الحدث الذي سينفذ عند إستدعاء الصفحة حيث يقوم بإستدعاء الإجراءية الأولى كي يتم ربط الأداة بالبيانات و تظهر لنا على الواجهة التطبيقية. و نرى الحدث dl_EditCommand و هو يحدد أي عنصر سيتم تفعيل ال EditTemplate عنده كي تظهر الأدوات اللازمة لمرحلة التعديل و نقوم بإعادة ربط البيانات. أما الحدث الأخير فهو يقوم بتعديل البيانات في الجدول عند الضغط على زر التعديل حيث يأخذ البيانات من ال TextBox و يقوم بإستبدالها بما هو موجود في الجدول قبل التعديل ثم تقوم بإعادة ربط الأداة بالبيانات كي تظهر لنا البيانات الجديدة بعد التعديل و يجب أن لا نتجاهل هذه النقطة.

الآن نذهب لجزء ال HTML و نرى أداة ال DataList التي أضفناها كما يلي :

كود

```

<asp : DataList id="dl" runat="server" DataKeyField="id" OnUpdateCommand="dl_UpdateCommand"
width="33%" OnEditCommand="dl_EditCommand" border="2">
<ItemTemplate>
<%# DataBinder.Eval(Container.DataItem,"category") %>
<br />
<asp:LinkButton id="edit" commandName="edit" text="Edit!" runat="server" />
</ItemTemplate>
<EditItemTemplate>
<p>
<asp:TextBox id="newCatName" runat="Server" Text='<%#
DataBinder.Eval(Container.DataItem,"category") %>' />

<asp:LinkButton id="update" commandName="update" text="Update!" runat="server" />

```

```
</p>
</EditItemTemplate>
</asp : DataList>
```

نرى أننا أضفنا الخاصية DataKeyField وهي مهمة للغاية و سأشرح ما هي و ما الفائدة منها.
 الآن عند الضغط على أي عنصر بيانات سوف نقوم بإستدعاء حدث معين و يلزمنا تمرير قيمة نستطيع من خلالها تحديد السجل الذي سنجري عليه التعديل و بالعادة نمرر قيمة ال Primary Key في الجدول كي نستخدمها كشرط في الإستعلام SQL أو في ال DataView كي نصل للسجل المطلوب و هنا مررنا قيمة ال id الخاص بالسجل و طبعاً سيمرر حسب ال Template الذي سنضغط عليه.
 في منطقة ال Item Template قمنا بكتابة تعليمة تظهر لنا إسم التصنيف و تظهر أسفله زر مكتوب عليه كلمة Edit! كما هو موضح هنا :

كود

```
<ItemTemplate>
<%# DataBinder.Eval(Container.DataItem,"category") %>
<br />
<asp:LinkButton id="edit" commandName="edit" text="Edit!" runat="server" />
</ItemTemplate>
```

أهم نقطة أننا حددنا ال CommandName في الزر إلى edit .

بعد ذلك نرى الجزء الخاص بال EditItemTemplate حيث أدرجنا فيه TextBox تظهر إسم القسم كمحتوى لها أو كقيمة إفتراضية و بجانبها زر من أجل تعديل البيانات كما يلي :

كود

```
<EditItemTemplate>
<p>
<asp:TextBox id="newCatName" runat="Server" Text='<%#
DataBinder.Eval(Container.DataItem,"category") %>' />

<asp:LinkButton id="update" commandName="update" text="Update!" runat="server" />
</p>
</EditItemTemplate>
```

و نرى أننا كتبنا update كقيمة لل commandName كي يتم إستدعاء حدث التعديل.

و الآن نعود للحدث dl_EditCommand :

كود

```
Sub dl_EditCommand(source As Object, e As DataListCommandEventArgs)
dl.EditItemIndex=e.Item.ItemIndex
```

DataBind()

End Sub

dl_EditItemIndex تأخذ قيمة رقمية و هي تحدد رقم العنصر الذي سنعدله أي الذي سنظهر له ال EditItemTemplate و القيمة هي عبارة عن قيمة ممررة مع المتغير e الذي تم تمريره كباراميتر للحدث و من أجل الوصول لرقم العنصر الذي قام بتنفيذ الحدث نكتب e.Item.ItemIndex ثم بعد ذلك قمنا بربط البيانات.

أما الجزء الأخير فهي الحدث الخاص بالتعديل على قاعدة البيانات و دعونا نكتب الكود هنا ثم سأقوم بشرحه بعدها:

كود

```
Sub dl_UpdateCommand(source As Object, e As DataListCommandEventArgs)
```

```
Dim newCatName As TextBox
```

```
Dim catID As String
```

```
Dim prov As String
```

```
Dim SQL As String
```

```
Dim cn As OleDbConnection
```

```
Dim reader As OleDbDataReader
```

```
Dim cmd As OleDbCommand
```

```
prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" & Server.MapPath("db.mdb")
```

```
SQL="Update Categories set Category=@newCatName where id=@catID"
```

```
cn = new OleDbConnection(prov)
```

```
cmd = new OleDbCommand(SQL,cn)
```

```
newCatName=e.Item.FindControl("newCatName")
```

```
catID=dl.DataKeys(e.Item.ItemIndex)
```

```
cmd.Parameters.Add("@newCatName",newCatName.Text)
```

```
cmd.Parameters.Add("@catID",catID)
```

```
cn.Open()
```

```
cmd.ExecuteNonQuery()
```

```
cn.Close()
```

```
dl.EditItemIndex=-1
```

```
DataBind()
```

```
End Sub
```

في البداية قمت بتعريف بعد المتغيرات و الأدوات كالتالي :

كود

```
Dim newCatName As TextBox
Dim catID As String
Dim prov As String
Dim SQL As String
Dim cn As OleDbConnection
Dim reader As OleDbDataReader
Dim cmd As OleDbCommand
```

حيث أن ال newCatName من نوع مربع نص TextBox و رقم ال CatID الخاص بالتصنيف من نوع نص تعريفات تخص الربط بقاعدة البيانات كلنا نعرفها.

و الآن لنحصل على قيمة ال newCatName نحتاج للحصول على الأداة التي أضفناها لل DataList و نحتاج لإستخدام الطريقة FindControl لتقوم بهذه الوظيفة و بالعادة تقوم ال FindControl بإعادة رقم ١- في حال عدم إيجادها للأداة المطلوبة أما ال catID فنستطيع الحصول على قيمة رقم التصنيف من خلال تمرير رقم العنصر التسلسلي حسب ظهوره في الأداة DataList و طبعا يبدأ الترقيم التسلسلي من صفر و هو ليس برقم ال PK في الجدول الخاص بالتصنيفات في قاعدة البيانات لذا نحتاج لتمرير هذا الرقم للطريقة DataKeys و التي تعيد قيمة الحقل المفتاحي و نحن قد حددناه الحقل id عند إدراج الأداة DataList.

كود

```
newCatName=e.Item.FindControl("newCatName")
catID=dl.DataKeys(e.Item.ItemIndex)
```

من الإمكانيات المميزة في الدوت نت و التي سهلت الكثير من العمل علينا و حدثت من الأخطاء التي يقع فيها المبرمجون في كتابة جمل الإستعلام عند تمرير متغيرات لجمله الإستعلامات أنها قدمت لنا طريقة جديدة لكتابة الإستعلام SQL و دعونا نرى هنا :

كود

```
SQL="Update Categories set Category=@newCatName where id=@catID"
```

نرى أننا إستخدمنا الرمز @ و بعده مررنا إسم الباراميتير و كي نضيفه نكتب التالي :

كود

```
cmd.Parameters.Add("@newCatName",newCatName.Text)
cmd.Parameters.Add("@catID",catID)
```

طبعا نحتاج لل cmd و التي هي OleDbCommand كي نضيف إليها قيم المتغيرات حيث أن ال @newCatName كانت قيمتها هي قيمة مربع النص و التي هي newCatName.Text أما ال @catID فقد حصلنا عليها قبل قليل مستفيدين من الباراميتير الممرر للحدث e. بعد ذلك نقوم بفتح الإتصال و نستخدم ال DataReader و أعتقد أنكم قرأتم درسه و عرفتم أننا في حال التعديل و الحذف نحتاج لإستخدام ExecuteNonQuery لأننا لا نريد إرجاع بيانات كما هو الحال في ال ExecuteReader أو ال ExecuteScalar ثم نغلق الإتصال كما هو هنا :

```
cn.Open()
cmd.ExecuteNonQuery()
cn.Close()
```

بعد ذلك كتبنا dl.EditItemIndex=-1 والتي تفيد في حجب ال EditItemTemplate و إعادته لحالته الأولى كما كان في حال ال ItemTemplate و ثم ربطنا البيانات كي نضمن ظهورها بعد عملية التعديل.

هذا الدرس المهم جدا سأشرح عن ال DataTable و قد ذكرناها بشكل مبسط جدا في دروس سابقة من أجل إظهار نتائج من جدول معين مخزن في ال DataSet. طبعا أولا يجب أن نعرف ما هو ال DataTable؟ بكل بساطة هو عبارة عن ناتج بيانات مستخرجة من جدول من قاعدة بيانات و مخزن في الذاكرة في ال dataset التي هي تعتبر كقطعة ذاكرة أو كمخزن لعدد من ال DataTables .

قلنا أنه عند إستدعاء الطريقة Fill لل DataAdapter سوف تقوم بتخزين ال DataTable في ال DataSet و قد يكون عندها عدد منها و لكن هل يمكن عمل DataTable بشكل برمجي ؟

دعونا نذكركم بمثال نقوم بإدخال ال DataTable إلى ال DataSet كما ذكرنا في درس سابق :

```
// VB.NET + SQL Server
Dim ds As DataSet
Dim cn As SqlConnection
Dim da As SqlDataAdapter
ds = new DataSet()
cn = new SqlConnection("Server=localhost; database=db; UID=sa;PWD=123")
da = new SqlDataAdapter("SELECT * From Students",cn)
da.Fill(ds,"std")
```

هذه هي الطريقة التقليدية و لكن يمكننا عمل جدول DataTable برمجي و إسناد قيم له و يلزمنا إضافة Namespace من أجل التعامل مع ال dataTable و قلنا أنها System.Data و ها هو المثال :

```
Dim tbl As DataTable
Dim myRow As DataRow
Dim myCol As DataColumn
' create the DataTable
tbl = new DataTable("myTable")

' create Columns
myCol = new DataColumn("studentID" , GetType (Int32) )
tbl.Columns.Add(myCol)
```

```
myCol = new DataColumn("studentName" , GetType (String) )
tbl.Columns.Add(myCol)
```

```
myCol = new DataColumn("studentEmail" , GetType (String) )
tbl.Columns.Add(myCol)
```

```
' create Rows
```

```
myRow = tbl.NewRow()
myRow("studentID")=13
myRow("studentName")="Mohd. Khalil Abu Salah"
myRow("studentEmail")="mohd_kh_80@hotmail.com"
tbl.Rows.Add(myRow)
```

```
myRow = tbl.NewRow()
myRow("studentID")=7
myRow("studentName")="Jumana"
myRow("studentEmail")="jumana_777@yahoo.com"
tbl.Rows.Add(myRow)
```

```
' Display DataTable
```

```
For each myRow in tbl.Rows
    For each myCol in tbl.Columns
        Response.Write(myRow(myCol))
    Next
Next
```

الآن نأتي لشرح ما سبق في الكود الذي ذكرناه من أجل إنشاء DataTable برمجيا.

أولا قمنا بتعريف جدول tbl من الفئة DataTable و عرفت صف myRow من المجموعة DataRow Collection و عمود myCol من المجموعة DataColumn Collection .

بعد ذلك قمنا بإنشاء الجدول و أسميته myTable و هي تشبه ما فعلناه عندما قمنا بتعبئة ال DataSet بجدول معين كالتالي:

كود

```
Da.Fill(ds,"myTable")
```

الآن أصبح عندنا في الذاكرة جدول برمجيا إسمه tbl و من أجل إكماله يلزمنا إضافة أعمدة بيانات و صفوف أو سجلات لها و هذا ما قمنا به حيث قمنا بإنشاء ٣ أعمدة الأول يحمل رقم الطالب و الثاني لإسمه و الثالث لبريده الإلكتروني كما هو هنا :

كود

```
' create Columns
myCol = new DataColumn("studentID" , GetType (Int32) )
tbl.Columns.Add(myCol)
```

```
myCol = new DataColumn("studentName" , GetType (String) )
tbl.Columns.Add(myCol)
myCol = new DataColumn("studentEmail" , GetType (String) )
tbl.Columns.Add(myCol)
```

طبعا العمود سيكون عبارة عن كائن اسمه myCol من نوع DataColumn و يحمل العمود الأول الرقم الخاص بالطالب studentID و جعلنا نوعية البيانات المدخلة من النوع Int32 ثم قمنا بإضافة العمود للجدول كما رأينا أو كما هو موضح هنا :

كود

```
tbl.Columns.Add(myCol)
```

و نستمر في ذلك مع تغيير اسم العمود و نوعية البيانات فنلاحظ أنها من نوع String في اسم الطالب و بريده الإلكتروني و هكذا .
بعد ذلك جاء وقت إدخال البيانات التي هي على شكل سجلات أو صفوف للبيانات rows كالتالي :

كود

```
' create Rows
```

```
myRow = tbl.NewRow()
myRow("studentID")=13
myRow("studentName")="Mohd. Khalil Abu Salah"
myRow("studentEmail")="mohd_kh_80@hotmail.com"
tbl.Rows.Add(myRow)
```

نلاحظ أن الصفوف تحوي البيانات مباشرة و هذا سهل للغاية حيث قمنا بعمل صف جديد من خلال الطريقة NewRow التي تقوم بإنشاء صف Row جديد في الجدول tbl ثم نقوم بإسناد القيم له كما هو بالمثال الأعلى و بالآخر نقوم بإضافة السجل أو الصف للجدول tbl .

الآن نحن إنتهينا من عملية تكوين الجدول و تعبئته بالبيانات و يبقى أن نتأكد فقط من أن هناك بيانات مدخلة لذا قمنا بإضافة هذا الجزء :

كود

```
' Display DataTable
```

```
For each myRow in tbl.Rows
For each myCol in tbl.Columns
Response.Write(myRow(myCol) & "<BR>")
Next
Response.Write("<HR>")
Next
```

و هي عبارة عن دوارتين Loop من نوع For Loop تقوم بعرض محتويات الجدول المخزن بالذاكرة و طبعا في الدائرة الأولى قمنا بعمل دورة في كل السجلات على حسب الأعمدة من أجل إستخراج البيانات.

يمكننا معرفة عدد السجلات الموجودة في الجدول من خلال الطريقة التالية :

كود

```
Response.Write(tbl.Rows.Count)
```

حيث أن من الخصائص الموجودة داخل الفئة Rows هي count و التي تعطينا عدد السجلات الموجودة و بالتالي يمكننا معرفة عددها.

لكن أود أن أوضح نقطة خاصة لمن أراه يطلب حلا لمشكلة و يقوم أحد الأشخاص و أنا أشكره بدلا عن صاحب الطلب لجهدته بالرد بلغة معينة و لنفرض أنها ال C# فيقوم صاحب الطلب للأسف بطلب تحويل الكود إلى ال VB.NET . بصراحة شيء غريب جدا!!! لم لا نحاول أن نتعلم لوحدنا و نتعب قليلا؟؟ أنا مثلا أقوم مرة بالكتابة بال VB.NET و مرة بال C# كي أحاول أن أرضي الطرفين و لكن هذا لا يعني أن نتجاوز الأكواد التي هي باللغة الأخرى لأي حجة كانت فالأكواد سهلة و المهم ليس الكود نفسه و لكن الفكرة. صدقوني أنني أملك كتب كلها بلغة ال VB.NET و أنا أبرمج بلغة السي شارب فقط و أدرس من كتب كلها بلغة ال VB.NET و بصراحة وجدتها أفضل طريقة لأنني أقوم بفهم المطلوب من الكتابة و كأنها لغة إيسيدو كود و أقوم بكتابتها بيدي بالسي شارب و من هنا تعلمت. أتمنى أن تدركوا هذه النقطة.

عموما سأقوم بطرح مثال آخر بلغة السي شارب كي لا يغضب أنصارها مني و سأتناول فكرة جميلة جدا و هي إستخراج بيانات من جدول من قاعدة البيانات و لكن سأضيف لها عمودا يقوم بعرض أرقام تسلسلية AutoNumber كنوع من الترقيم للسجلات.

ها هو المثال :

كود

```
// C#
```

```
DataTable tbl;
```

```
DataColumn myColumn;
```

```
DataSet ds = new DataSet();
```

```
tbl = new DataTable("categories");
```

```
myColumn = new DataColumn("CategoryID", Type.GetType("System.Int32"));
```

```
myColumn.AutoIncrement = true;
```

```
tbl.Columns.Add(myColumn);
```

```
ds.Tables.Add(tbl);
```

```
string prov="Provider=Microsoft.Jet.OleDb.4.0;Data Source=" + Server.MapPath("db.mdb");
```

```
string SQL="SELECT * FROM Categories";
```

```
OleDbConnection cn = new OleDbConnection(prov);
```

```
OleDbDataAdapter da = new OleDbDataAdapter(SQL,cn);
```

```
cn.Open();
```

```
da.Fill(ds,"categories");
```

```
dg.DataSource=ds;
```

```
dg.DataBind();
```

```
cn.Close();
```

قمنا بتعريف DataTable بإسم tbl و عمود بيانات باسم myColumn و الفكرة تقوم على أننا سوف ننشئ DataSet نخزن بها جدول به عمود يزداد بشكل تلقائي و إضافة جدول مستخرج من قاعدة البيانات من أجل إضافته لنفس ال DataSet التي أسميناها ds و ذلك كما درسنا سابقا كيفية إستخراج البيانات عن طريق ال DataAdapter الذي يعمل كوسيط بين مصدر البيانات و قطعة الذاكرة ال DataSet. الآن الخطوة الأولى هل إدراج الجدول والعمود برمجيا كما رأينا في أول الكود و حددنا خاصية العمود ليكون ذو رقم تلقائي كالتالي :

كود

```
myColumn.AutoIncrement = true;
```

بعد ذلك أدرجنا الجدول المستخرج من قاعدة البيانات و لنلاحظ أنه في المرتين أي عند إدخال الجدول مباشرة بالكود كان إسمه categories و هو إسم الجدول في ال DataSet و هو نفسه الذي أدرجناه من خلال إستدعاء الطريقة Fill من ال DataAdapter و بعد ذلك قمنا بعملية ربط البيانات DataBinding بين الجدول الموجود في ال DataSet و أداة ال DataGrid التي أدرجناها من الواجهة الرسومية للتصميم و أسميناها dg أيضا هناك ميزة و هي التعامل مع دوال تقوم بمجموعة عمليات رياضية مثل الجمع Sum و يمكننا إضافة عمود أو صف به مجموع بيانات عمود معين و هذا مثال يضاف للكود السابق و من يجرب الكود سوف يرى النتيجة بنفسه :

كود

```
myColumn = new DataColumn("summation", Type.GetType("System.Single"),"Sum(categoryID)");
```

```
tbl.Columns.Add(myColumn);
```

هناك ملاحظة صغيرة بخصوص إختلاف كبير ما بين ال VB.NET و ال C# من خلال المثالين السابقين نراه هنا :

كود

```
//VB.NET
myCol = new DataColumn("studentID" , GetType (Int32) )

//C#
myCol = new DataColumn("studentID", Type.GetType("System.Int32"));

//VB.NET
Response.Write(myRow(myCol) & "<BR>")

//C#
Response.Write(myRow[ myCol ] + "<BR>");

//VB.NET
myRow = tbl.NewRow()
myRow("studentID")=13
myRow("studentName")="Mohd. Khalil Abu Salah"
myRow("studentEmail")="mohd_kh_80@hotmail.com"
tbl.Rows.Add(myRow)

//C#
myRow = tbl.NewRow();
myRow["studentID"]=13;
myRow["studentName"]="Mohd. Khalil Abu Salah";
myRow["studentEmail"]=mohd_kh_80@hotmail.com;
tbl.Rows.Add(myRow);
```

الدرس من شرح
الأستاذ / محمد خليل أبو صلاح
جمع أخوكم أحمد حموده
منتديات الفريق العربى للبرمجة

