



جامعة أهلاً عرب

كلية فحول بيسك

قسم الأبحاث والتطوير

درس:

تعلم معنا استخدام

ADO

الجزء الأول

منقول من منتدى VB4ARAB

نسخة منقحة

إعداد:

الأستاذ / هيثم القلاف

جمع وتنسيق: أبوبكر شرف الدين سويدان

بسم الله الرحمن الرحيم

مكتبة ADO:

مكتبة ADO تعتبر الأسلوب الجديد (قبل الأداة الجديدة ADO.NET) للتعامل مع قاعد البيانات بعيداً عن البرنامج الأم الذي أنشئت منها قاعدة البيانات ... فمثلاً يمكننا الاتصال وقراءة محتويات قاعدة بيانات أكسس Access دون الحاجة لوجود Access فهذه المكتبة توفر خدمة الاتصال والإجراءات المختلفة للتعامل معها.

توفير مكتبة ADO في البرنامج:

1. افتح (Project / References) ثم اختر الخيار (Microsoft ActiveX Data Objects 2.X)

(Library) حيث x تمثل رقم إصدار المكتبة.

2. انقر فوق (ok).

كيفية إجراء اتصال بقاعدة البيانات:

لإجراء اتصال بقاعدة البيانات نتبع الخطوات التالية:

- اكتب البرمجة التالية في قسم المتغيرات العامة:

كود برمجي

```
Dim db As New ADODB.Connection
```

- ثم اكتب البرمجة التالية في إجراء حدث التحميل للنافذة:

كود برمجي

```
Private Sub Form_Load()  
db.Provider = "Microsoft.JET.OLEDB.4.0;"  
db.Open App.Path & "\db1.mdb"  
MsgBox "تم الاتصال بقاعدة البيانات بنجاح"  
End Sub
```

كيفية فتح الجداول:

تعرفنا على هوية الأكواد التي نستطيع من خلالها فتح اتصال مع قواعد البيانات والآن ننقل إلى الخطوة التالية ، وهي كيفية فتح الجداول الموجودة في قاعدة البيانات.

ولكي نفتح جدول في قاعدة البيانات ، يجب أولاً تأمين اتصال مع قاعدة البيانات الموجود بها الجدول

المراد فتحه ، يعني نستخدم الكود السابق لفتح قاعدة البيانات مع زيادة طفيفة وهي كالتالي:

كود برمجي

```
Dim db As New ADODB.Connection  
Dim rs As New ADODB.Recordset  
Private Sub Form_Load()  
db.Provider = "Microsoft.JET.OLEDB.4.0;"  
db.Open App.Path & "\db1.mdb"  
rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly  
End Sub
```

انظر للمتغير rs حيث أسندنا إليه خصائص Recordset وهو الذي يُعنى بكل العمليات التي تجري على الجداول من فتح وحذف وتعديل والخ.

وأيضا لاحظ السطر:

rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly

حيث الجدول هنا هو Table1 و db تحدد الجدول الموجود في قاعدة البيانات.

وبنفس الخطوات نستطيع فتح جدول في قواعد بيانات من النوع SQL.

طريقة عرض السجلات:

تعرفنا آنفا كيفية إجراء اتصال مع قواعد البيانات ، ومن ثم تعرضنا لطريقة فتح الجداول ، والآن بمشيئة الله سنتعلم كيفية عرض السجلات ، والإبحار فيما بينها . وكما نعرف إن كل خطوة تبني عليها خطوات ، بمعنى لا يمكن مثلا فتح جدول بدون تأمين اتصال مع قاعدة البيانات ، وكذلك أيضا لا يمكن عرض السجلات وقراءة بياناتها بدون فتح الجدول ، فكل عملية مطلوبة وحتمية ولا يمكن تجاهلها ، لأنها بمثابة سلسلة متكاملة. وهنا سوف نحتاج إلى إنشاء بعض الأدوات وتعريفها بالشكل المناسب كصناديق النصوص واللافتات وبعض أزرار المفاتيح. وقبل كل شيء يجب إنشاء قاعدة بيانات في برنامج ACCESS ، وأسم قاعدة البيانات هو DB1.MDB وتحتوي على جدول واحد ويحمل اسم Table1 ويوجد في الجدول ثلاثة سجلات:

- ❖ السجل الأول ID: ويحوي بيانات رقمية (المفتاح الأساسي).

- ❖ السجل الثاني CATEGORYNAME.

- ❖ السجل الثالث .DISCRIPTION.

من الملاحظ إن أسم الحقول وضعت باللغة الإنجليزية وفي الحقيقة لا يهم وضعها بحروف إنجليزية أو عربية وهنا اخترتها إنجليزية لكي تكون مميزة وملاحظة.

تنبيه: في حالة استخدامك للحروف العربية أو الأسماء الإنجليزية المتفرقة مثل CATEGORY NAME فدائما ضع اسم الحقل ما بين الأقواس المربعة وليست الأقواس المعتادة إنما استخدم هذه الأقواس [] وضع الاسم وسطها مثل [CATEGORY NAME].

على أي حال ، نشرع في تصميم الفورم وكتابة الأكواد المناسبة.

أولاً وكما نعرف نختار مشروعاً جديداً New Project ثم نختار النوع القياسي Standard. بعد ظهور الفورم ندرج ثلاثة عناصر تسمية Labels فيه ، بالنسبة لترتيب الـ Labels فلا يهم رتبها بالشكل الذي تراه مناسباً. وعنوانها بالشكل التالي ، طبعا سنستخدم خاصية Caption لعنونة الـ Labels.

الأول : ID.

الثاني Category Name .

الثالث Description .

وبعد ذلك أيضاً نضيف ثلاثة صناديق نصوص Text Box ونضعها أمام كل واحد مرتبة حسب وضع الـ Label فمثلاً الـ Label رقم واحد يكون أمامه الـ Textbox رقم واحد . وهكذا بالنسبة للبقية. بعد ذلك نضيف أربعة Command إلى الفورم وسوف نستخدمها في عملية الإبحار ما بين السجلات.

عناوين الـ Command تكون بالشكل التالي ومتسلسلة الأرقام:

الزر الأول يحمل العبارة First Record.

الزر الثاني يحمل العبارة Previous Record.

الزر الثالث يحمل العبارة Last Record.

الزر الرابع يحمل العبارة View Record.

ولا تنسى إن تحفظ المشروع ولنسميه مثلاً ViewRecord.

والآن وصلنا إلى دور كتابة الكود . ومثل ما عرفنا سابقاً إن لإجراء إي اتصال مع إي قاعدة بيانات يجب تهيئة مكتبة ADO وقد شرحنا سابقاً طريقة تهيئتها وبشكل مبسط.
وهنا سنكمل الكود الذي تم فيه الاتصال مع الجدول / تذكير بالكود:

كود برمجي
<pre>Dim db As New ADODB.Connection Dim rs As New ADODB.Recordset ' Private Sub Form_Load() ' db.Provider = "Microsoft.JET.OLEDB.4.0;" db.Open App.Path & "\db1.mdb" rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly End Sub</pre>

أولاً نقوم بإنشاء Sub وسنسميه ViewRecord :

كود برمجي
<pre>Sub ViewRecord() ' End Sub</pre>

والهدف منه هو عرض السجل الحالي الذي يتم استدعائه من قبل الضغط على الـ Command الذي تم إنشاؤه سابقاً. وستوضح الرؤية بشكل أفضل بعد إتمام الكود. فلنتابع...
وبالمناسبة توجد عدة طرق لعرض السجلات وأقصد بعرض السجلات هو إظهار البيانات في صناديق النصوص TextBox وليس الانتقال ما بينها. وسأستعرض طريقتين الأولى المفضلة لي والثانية لا أحبذ استخدامها ولكن للمعرفة.

سأدرج الكود أولاً ثم سأضع التعليقات حول طريقة اشتغال الكود:

كود برمجي

```
Sub ViewRecord()  
If Not rs.RecordCount = 0 Then  
    Text1.Text = ""  
    Text2.Text = ""  
    Text3.Text = ""  
    If Not IsNull(rs![ID]) Then Text1 = rs![ID]  
    If Not IsNull(rs![CategoryName]) Then  
        Text2 = rs![CategoryName]  
    End If  
    If Not IsNull(rs![Description]) Then  
        Text3 = rs![Description]  
    End If  
End If  
End Sub
```

الجملة الشرطية والتي هي **If Not rs.RecordCount = 0 Then** :

قبل أن أشرح ما فائدتها دعني أوضح فائدة الخاصية **RecordCount** وهذه الخاصية في الحقيقة ترجع عدد السجلات الموجودة في الجدول ، فمثلاً يوجد عندنا 10 سجلات مخزنة في الجدول فإن هذه الخاصية ترجع لنا العدد 10 ، فبها نستطيع معرفة ما يحتويه الجدول من عدد السجلات وأيضاً نستطيع من خلالها التأكد من أن الجدول فارغ ولا يحتوي على بيانات . أتوقع بدأنا نفهم سبب استخدامها هنا.

وسبب استخدامها في الجملة الشرطية هو التحقق من عدم خلو الجدول من البيانات فإذا كان الجدول فارغ فلا يتم إرجاع قيم الحقول ، وإلا سوف نقع في دائرة الأخطاء البرمجية.

أما السطور الثلاثة التي تلي الجملة الشرطية فهو ثغني بإفراغ محتويات الـ **Textbox** لكي يتم تجهيزها لاحتضان بيانات جديدة.
أما هذا الكود:

كود برمجي

```
If Not IsNull(rs![ID]) Then Text1 = rs![ID]  
If Not IsNull(rs![CategoryName]) Then Text2 = rs![CategoryName]  
If Not IsNull(rs![Description]) Then Text3 = rs![Description]
```

إذ جئنا لترجم الكود السابق لنص مقروء فالنتائج يكون:

إذا كان الحقل الذي اسمهم **ID** فارغ ولا يحتوي على بيانات حينئذ تجاهل إسناد القيمة الفارغة إلى صندوق النصوص . وهذه هي الترجمة الحقيقية للكود.

IsNull(rs![ID])

تتحقق من إن حقل **ID** لا يحوي قيمة فارغة.

Text1 = rs![ID]

هنا نرجع قيمة الحقل إلى الـ **Textbox** ، وهكذا بالنسبة للبقية.

والآن وبعد الانتهاء من كتابة الكود السابق يكون بمقدورنا استعراض البيانات . ولكن قبل تجربة هذه المرحلة يجب وضع كود في قسم التحميل **Form_Load** وهذا الكود يقوم باستدعاء الـ **Sub** الذي هو **ViewRecord** فلذلك نكتب الكود التالي :

كود برمجي
Call ViewRecord

ويكون مكانه أسفل الكود الخاص بفتح الجدول . والآن تستطيع تشغيل البرنامج لترى ما تم إنجاز . تلك كانت الطريقة الأولى والتي أفضّلها أما الثانية فيه قريبة من الطريقة الأولى ولكن عوضا عن استخدام اسم الحقل سوف نستخدم رقم الحقل أنظر الكود :

كود برمجي
<pre>Sub ViewRecord() If Not rs.RecordCount = 0 Then Text1.Text = "" Text2.Text = "" Text3.Text = "" If Not IsNull(rs.Fields(0).Value) Then Text1 = rs.Fields(0).Value If Not IsNull(rs.Fields(1).Value) Then Text2 = rs.Fields(1).Value If Not IsNull(rs.Fields(2).Value) Then Text3 = rs.Fields(2).Value End If End Sub</pre>

وأيضا توجد طريقة أخرى سأذكرها من باب العمل بالشيء . وفي هذه الطريقة نضع الكود في قسم التحميل:

كود برمجي
<pre>Private Sub Form_Load() ' db.Provider = "Microsoft.JET.OLEDB.4.0;" db.Open App.Path & "\db1.mdb" rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly ' Set Text1.DataSource = rs Set Text2.DataSource = rs Set Text3.DataSource = rs Text1.DataField = "ID" Text2.DataField = "CategoryName" Text3.DataField = "Description" End Sub</pre>

وهكذا باب البرمجة والإبداع موجود ولكل شخص الطريقة التي يفضل استخدامها بغض النظر عن طول الكود أو قصره .

والآن وبعد أن تعرفنا على بعض الطرق المختلفة لعرض البيانات سوف نقوم بكتابة كود لتصفح السجلات أو البيانات والانتقال ما بينها.

نبدأ بكتابة كود الانتقال إلى السجل الأول **Record First** وسنضع الكود في **Command1**

الكود:

كود برمجي

```
If Not rs.BOF Then rs.MoveFirst
If rs.BOF And rs.RecordCount > 0 Then
    Beep
    Exit Sub
End If
Call ViewRecord
```

السطر الأول:

If Not rs.BOF Then rs.MoveFirst

ومعنى الكود السابق لفظياً هو إذا كان موقع السجل ليس بالأول فانتقل إلى السجل الأول. والخاصية BOF هي المسنولة عن إرجاع القيمة التي تشير إلى موقع السجل المعروف ، و MoveFirst هي المسنولة عن تحقيق الانتقال و في حالة عدم تحديد الجملة الشرطية سنقع في خطئ برمجي فادح. أما باقي الكود:

كود برمجي

```
If rs.BOF And rs.RecordCount > 0 Then
    Beep
    Exit Sub
End If
```

وفي الحالة الأخرى وإذا كانت نقطة موقع العرض موجودة على السجل الأول وكانت RecordCount أكبر من الصفر فإنها تخرج من طور التنفيذ ولا يتم إتمام الكود . فبإتمام الكود سنحصل على خطأ . بالنسبة للأكواد المتبقية فيها تقريباً بنفس الفكرة ولكن سيختلف المضمون فعوضاً عن استخدام MoveFirst سنستخدم مثلاً MovePrevious مع إضافات بسيطة . أما آخر سطر من الكود هو Call ViewRecord فالغاية منه هو إظهار ما تم تنفيذه في الكود الذي سبقه . ولأن سأعرض باقي الأكواد ، واترك لكم فهم الكود ، كود التراجع أو الانتقال إلى السابق MovePrevious ونضعه في Command2.

كود برمجي

```
Private Sub Command2_Click()
If Not rs.BOF Then rs.MovePrevious
If rs.BOF And rs.RecordCount > 0 Then
    Beep
    rs.MoveFirst
    Exit Sub
End If
Call ViewRecord
End Sub
```

لعرض السجل التالي MoveNext نستخدم الكود الذي يلي ونضعه في Command3 .

كود برمجي
Private Sub Command3_Click() ' If Not rs.EOF Then rs.MoveNext If rs.EOF And rs.RecordCount > 0 Then Beep rs.MoveLast End If ' Call ViewRecord End Sub

ولعرض السجل الأخير MoveLast والكود يسكون موضعه في Command4 .

كود برمجي
Private Sub Command4_Click() ' If Not rs.EOF Then rs.MoveLast If rs.EOF And rs.RecordCount > 0 Then Beep Exit Sub End If ' Call ViewRecord End Sub

فبهذا سنكون قد انتهينا من إعداد مفاتيح الانتقال ما بين السجلات وعرض البيانات في صناديق

النصوص TextBox .

إدخال البيانات ، تحريها ، حذفها :

مررنا سابقا بعدة مراحل كانت الأولى إجراء اتصال مع قواعد البيانات ومن ثم فتح الجدول وكان آخرها هو عرض السجلات وتصفحها . والآن سنأتي على ذكر العمليات أو الإجراءات والتي تعتبر جزء مهم من أجزاء التعامل مع قواعد البيانات والتي تتبلور في عمليات الإضافة وتحري وحذف للبيانات.

وسنبدأ بعملية الإضافة:

قد رأينا في قسم استعراض السجلات إن هناك عدة طرق للوصول إلى هدف معين في البرمجة ولكل مبرمج طريقته أو أسلوبه المفضل. والبرمجة في الحقيقة هي طريق للإبداع .

طبعاً للوصول إلى هذه النقطة نقطة إضافة البيانات يجب توفير:

- اتصال مع قواعد البيانات.
- فتح الجدول بالطريقة المعتادة ولكن مع إعطاء صلاحية الإضافة أو صلاحيات المعالجة.

والآن لتأمين اتصال يتيح لنا معالجة البيانات يجب إجراء تغيير طفيف في كود الاتصال مع الجدول.
أنظر إلى الكود السابق:

كود برمجي
Dim db As New ADODB.Connection Dim rs As New ADODB.Recordset Private Sub Form_Load() db.Provider = "Microsoft.JET.OLEDB.4.0;" db.Open App.Path & "\db1.mdb" rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly End Sub

الكود المعدل:

كود برمجي
Dim db As New ADODB.Connection Dim rs As New ADODB.Recordset ' Private Sub Form_Load() db.Provider = "Microsoft.JET.OLEDB.4.0;" db.Open App.Path & "\db1.mdb" rs.Open "[Table1]", db, adOpenStatic, adLockPessimistic End Sub

هل لاحظت الفرق ما بين الكود الذي استخدمناه في السابق والكود الجديد ، نعم يوجد فرق في السطر:
السطر من الكود السابق:

كود برمجي
rs.Open "[Table1]", db, adOpenStatic, adLockReadOnly

السطر من الكود الجديد:

كود برمجي
rs.Open "[Table1]", db, adOpenStatic, adLockPessimistic

في الكود السابق قد أمنا اتصال مع الجدول في وضعية القراءة فقط ، ولكن عندما نريد إجراء عمليات
المعالجة يجب تأمين اتصال في وضعية الصلاحيات (Pessimistic) ، لهذا استبدلنا خاصية
adLockReadOnly بالخاصية adLockPessimistic .

نكمل . بعد إن تعرفنا على المسار الجديد والتغيير المحدث في فتح الجداول ، سنقوم بإضافة خمسة

:Commands

Command5: يحمل الاسم Add.

Command6: يحمل الاسم Edit.

Command7: يحمل الاسم Delete.

Command8: يحمل الاسم Save.

Command9: يحمل الاسم Cancel.

بعد ذلك سنقوم بإضافة Sub والغرض منه عدم إتاحة للخاصية الأخرى فمثلا عندما تضغط على مفتاح Add فمن المفروض إن المفاتيح الأخرى تكون غير متاحة أو ممكنة حيث لا يمكن إجراء عمليتين مع بعضهما البعض في نفس الوقت. على العموم سوف نتضح فكرة الـ Sub عند إكماله.

أسم الـ SUB : فليكن EnabledUnEnabled.

كود برمجي
Sub EnabledUnEnabled(n As Boolean)
End Sub

والكود الذي سيحويه الـ SUB هو:

كود برمجي
Sub EnabledUnEnabled(n As Boolean)
Text1.Locked = True
Text2.Locked = n
Text3.Locked = n
Command1.Enabled = n
Command2.Enabled = n
Command3.Enabled = n
Command4.Enabled = n
Command5.Enabled = n
Command6.Enabled = n
Command7.Enabled = n
Command8.Enabled = Not n
Command9.Enabled = Not n
If rs.RecordCount = 0 Then
Command6.Enabled = False
Command7.Enabled = False
End If
End Sub

قد تلاحظ الحرف n الذي أضفناه إلى SUB من Boolean و الـ Boolean هو متغير يرجع قيمة

True أو False وسوف نستفيد منه في عملية الاستدعاء Calling .

فمثلاً عندما نستدعي الـ SUB السابق بهذا الشكل:

كود برمجي
Call EnabledUnEnabled(True)

فسيكون الناتج هو إتاحة جميع المفاتيح ما عدا مفتاح Save والـ Cancel أما البقية ستكون متاحة والعكس صحيح. أما السطور الثلاثة الأولى فمهمتها قفل الـ Textbox ولا يمكن الكتابة فيه إلا بعد الضغط على مفتاح إضافة سجل جديد Add ، وأما ما بعدها فهي مفهومة الغرض.

أما الجزء الأخير من الكود وهو:

كود برمجي

```
If rs.RecordCount = 0 Then  
    Command6.Enabled = False  
    Command7.Enabled = False  
End If
```

فهذا يتكفل بعملية إغلاق مفتاح Edit و Delete وهذا الإغلاق مشروط وهو إذا كان الجدول فارغاً ولا يحتوي على بيانات. أما إذا كان الجدول يحتوي على البيانات فلا يتم تنفيذ الكود السابق . طبعاً بدون هذه الخاصية ستنتج أخطاء برمجية إذا ما تم النقر على المفاتيح السابقين وفي حالة عدم وجود أي بيانات حيث كيف يمكن تعديل أو تحرير بيانات في جدول معين والجدول في الأصل لا يحتوي على بيانات؟. وبهذا قد انتهينا من الـ SUB وأتمنى إن تكون فكرة الـ SUB مفهومة لدى الجميع.

والآن سننتقل إلى مفتاح الإضافة Add وكل الذي سيحويه هذا المفتاح هو التالي:

كود برمجي

```
Private Sub Command5_Click()  
Text1.Text = ""  
Text2.Text = ""  
Text3.Text = ""  
Call EnabledUnEnabled(False)  
    Text2.SetFocus  
End Sub
```

والآن سوف ننتقل إلى مفتاح الحفظ . Save وقبل إن اشرع في كتابة الكود أحببت انوه على إن عملية الحفظ هي العملية التي سوف يتم تخزين البيانات المدخلة فيها . فيجب إن نحرص على وجود بيانات ، إذا يجب وضع كود يقوم بتحقق من وجود البيانات قبل عملية الحفظ بخطوات .

وهذا الكود الذي سنضعه في مفتاح الحفظ . Save وبعدها سنتناول الكود خطوة خطوة:

كود برمجي

```
Private Sub Command8_Click()  
If Trim(Text2.Text) = "" Then  
    MsgBox "Cnnot contain a Null Value .", vbCritical, "Message"  
    Text2.SetFocus  
    Exit Sub  
End If  
With rs  
    .AddNew  
        ![CategoryName] = Trim(Text2.Text)  
        If Not Trim(Text3.Text) = "" Then ![Description] = Trim(Text3.Text) .Update  
    End With  
    rs.Requery  
    rs.MoveLast  
Call EnabledUnEnabled(True)  
Call ViewRecord  
End Sub
```

أولاً:

كود برمجي

```
If Trim(Text2.Text) = "" Then  
    MsgBox "Cnnot contain a Null Value .", vbCritical, "Message"  
    Text2.SetFocus  
    Exit Sub  
End If
```

ونستفيد منه في عملية التحقق المذكورة سابقاً وفي حالة لم يتم العثور على نص في Text Box رقم 2 فإنه سيقوم بأخبارك إنه لا يمكن إسناد قيمة فارغة. وسيعيدك إلى الـ Text Box الذي من المفترض تعبئته بالبيانات ، ومن ثم يخرج من الـ SUB ولا يتم تنفيذ ما بعده وهذا النوع من الأكواد أنا شخصياً أطلق عليه اسم (صائد الأخطاء).
أما الكود الذي يليه:

كود برمجي

```
With rs  
    .AddNew  
        ![CategoryName] = Trim(Text2.Text)  
        If Not Trim(Text3.Text) = "" Then ![Description] = Trim(Text3.Text)  
    .Update  
End With
```

فهنا سنقوم بإضافة السجل الجديد أو البيانات الجديدة والأمر AddNew هو المسئول عن الإضافة. وعلامة التعجب ! تفيد إن ما بعدها هو حقل وطبعاً بعد تحديد الحقل سوف نسند له البيانات التي تم إدخالها. وبالنسبة للحقول الضرورية أو المطلوبة فنكتفي بوضعها بهذا الشكل :

كود برمجي

```
![CategoryName] = Trim(Text2.Text)
```

أما الحقول غير ضرورية والمتروكة كخيار للمستخدم يقوم بتعبئتها أو يتركها فارغة . فهذه الحقول يجب وضع لها كود التحقق ويكون كالتالي :

كود برمجي

```
If Not Trim(Text3.Text) = "" Then ![Description] =Trim(Text3.Text)
```

والكود السابق يقوم بفحص الـ Text Box وإذا كان فارغ فهو يتجاهل الحقل أما إذا كان الـ Text Box يحوي على بيانات فإنه سيسند القيمة إلى الحقل الصحيح. أما الخاصية Update فهي المسئولة عن حفظ القيم في السجل وبدونها سيكون ما سبق عقيماً.
وسنأخذ السطور المتبقية كل على حده:

كود برمجي

```
rs.Requery
```

الـ Requery هي بمثابة Refresh في DAO حيث تقوم بتحديث للبيانات.

كود برمجي

```
rs.MoveLast
```

والـ MoveLast الانتقال إلى السجل الأخير .. ولماذا نستخدمها هنا؟ .. ببساطة لأنه السجل الجديد سيكون آخر سجل تم إضافته.

كود برمجي

```
Call EnabledUnEnabled(True)
```

ومثل ما عرفنا سابقا إن EnabledUnEnabled يقوم بعملية قفل أو فتح الـ Text Box والـ Commands وهنا يجب أفعالها فعندما ننتهي من أي إجراء يجب قفل ما تم فتحه أو إتاحتها.

كود برمجي

```
Call ViewRecord
```

والـ ViewRecord يقوم بعرض السجل الجديد . أنتهي.

بالنسبة لمفتاح إلغاء Cancel قم فقط بإضافة الكود التالي والكود مفهوم ولا يحتاج إلى توضيح:

كود برمجي

```
Private Sub Command9_Click()  
Call EnabledUnEnabled(True)  
Call ViewRecord  
End Sub
```

تحرير أو تعديل البيانات :

تحرير البيانات هي العملية التي يتم فيها استبدال البيانات السابقة ببيانات جديدة والفرق الأساسي ما بين عملية إضافة سجل جديد وتعديله تكمن في خاصية AddNew. حيث هذه الخاصية نستخدمها عند إضافة سجل جديد إما في العملية الأخرى وهي التحرير فلا نستخدمها فالسطور القادمة بإذن الله تبين ما سبق .
نبدأ بإضافة المتغير الذي سوف يأتي ذكره والهدف منه هو تعريف البرنامج إن المستخدم الآن يقوم بعملية إضافة أو تحرير وسيفيدنا هذا المتغير في عدم تكرار كتابة الأكواد حيث بدونها سوف نضطر إلى كتابة كود آخر وتخصيص مفتاح ليعمل على حفظ البيانات المعدلة.
المتغير:

كود برمجي

```
Dim EditORAdd As Boolean
```

ونضع المتغير في. General

ثم نضيف المتغير إلى مفتاح إضافة سجل جديد Add وجعل قيمته True . ليكون الكود كاملاً بالشكل التالي:

كود برمجي
Private Sub Command5_Click() ' Text1.Text = "" Text2.Text = "" Text3.Text = "" ' Call EnabledUnEnabled(False) Text2.SetFocus ' EditORAdd = True End Sub

والآن كلما نقرنا على مفتاح Add ستكون قيمة المتغير EditORAdd دائماً True ثم نضيف الكود التالي إلى مفتاح Edit .

الكود:

كود برمجي
Private Sub Command6_Click() ' Call EnabledUnEnabled(False) Text2.SetFocus EditORAdd = False End Sub

وهنا نضع قيمة المتغير دائماً تساوي False ، إذاً مما سبق يتبين لنا إن المتغير إذا كانت قيمته تساوي False فمعنا ذلك إن المستخدم يريد تحرير البيانات إما إذا كانت قيمة المتغير تساوي True فذلك يقودنا إلى إن المستخدم بصدد إدخال بيانات جديدة.

والآن سنضيف جملة شرطية إلى مفتاح الحفظ Save وهي ستحل محل السطر التالي من الكود الموجود في مفتاح الحفظ Save .

الكود المستبدل:

كود برمجي
.AddNew

الكود الجديد:

كود برمجي
If EditORAdd = True Then .AddNew

هل يمكنك ترجمة الكود السابق؟؟؟! هو ببساطة إذا كانت قيمة المتغير EditORAdd تساوي True فسيتم تنفيذ الخاصية AddNew إما إذا كانت قيمة المتغير EditORAdd لا تساوي True فإنه يتجاهل تنفيذها.

وأيضا سنضيف نفس الجملة الشرطية في آخر الكود من المفتاح Save انظر الكود بعد آخر تعديل.

كود برمجي

```
Private Sub Command8_Click()  
If Trim(Text2.Text) = "" Then  
    MsgBox "Cannot contain a Null Value .", vbCritical, "Message"  
    Text2.SetFocus  
    Exit Sub  
End If  
With rs  
    If EditORAdd = True Then .AddNew  
    ![CategoryName] = Trim(Text2.Text)  
    If Not Trim(Text3.Text) = "" Then ![Description] = Trim(Text3.Text)  
    .Update  
End With  
If EditORAdd = True Then  
    rs.Requery  
    rs.MoveLast  
End If  
Call EnabledUnEnabled(True)  
Call ViewRecord  
End Sub
```

وبهذا قد انتهينا من عملية تعديل البيانات فمثل ما جاء آنفا إن الفرق الأساسي بين العمليتين هي خاصية AddNew فحين تنفيذها سيتم إضافة سجل جديد أما حين تجاهلها فسيتم تعديل البيانات . وأيضا قد استخدمنا المتغير لكي يتجاهل خاصية الإنعاش Requery وكذلك خاصية الانتقال إلى السجل الأخير .MoveLast

حذف السجل :

الخاصية Delete هي التي تقوم بحذف السجل المفرد أو السجل الحالي وطريقة بنائها بالشكل التالي

الكود كاملاً :

كود برمجي

```
If MsgBox("Do you want to Delete a current record  ?"), _  
    vbQuestion + vbYesNoCancel, "Message") = vbYes Then  
    rs.Delete adAffectCurrent  
    Text1.Text = ""  
    Text2.Text = ""  
    Text3.Text = ""  
    Call EnabledUnEnabled(True)  
    Call Command3_Click  
    Call ViewRecord  
End If
```

وكود الحذف هو rs.Delete adAffectCurrent أما باقي الأكواد فيه مسألة تنظيمية ونستغلها أيضا في الحفاظ على استقرار البرنامج ومنع أي خطأ قد يسبب مشكلة في عملية الحذف فمثلا عندما يتم حذف جميع السجلات من الجدول يتم منع المستخدم من الضغط على مفتاح الحذف وبهذا نضمن عدم تنفيذ الكود حينما يكون الجدول فارغ من السجلات.

البحث (Search) :

ويمكننا البحث عن البيانات بطريقتين الأولى بالبحث بدلالة الكلمة والثاني البحث بدلالة حرف من كلمة أو جزء من نص.

نبدأ:

يمكننا استخدام الإجراء Find للبحث عن (نص ، رقم ، تاريخ) أو إي بيان على اعتبار إن هذا البيان قد تم تحديده وإسناده مسبقاً إلى الإجراء . Find وقد تكون هذه البيانات موجودة في السجل الحالي أو في السجل التالي أو السابق أو في السجل الأول أو الأخير ولهذا يجب علينا وضع بعض المعايير لكي يتم تنفيذ البحث بطريقة سليمة وصحيحة حيث لا يمكننا البحث عن بيان موجود في السجل الأول ونحن في وسط السجلات بدون وضع معيار معين يتم بواسطة الرجوع إلى السجل الأول ومن ثم تنفيذ عملية البحث . وطبعاً يمكننا البحث عن السجلات في الجدول الحالي فقط أو الجدول النشط يعني لا يمكن البحث في جدول غير نشط .

وطريقة بناء الإجراء Find بالشكل التالي :

كود برمجي
Find (Criteria, SkipRows, SearchDirection, Start)

Criteria

وتتكون هذه (الوسيطة) من اسم الحقل مع البيان الذي يجب إيجاده في الجدول ، ولكي نبني معيار صحيح يستلزم معرفة تصنيف البيان مع معرفة نوع البيانات التي يحتضنها الحقل من حيث إنها بيانات (نصية Text ، رقمية Numeric ، تاريخ/Time/Date) ، وهذه في الحقيقة من الشروط التي يجب توفرها عند بناء جملة البحث أو إجراء البحث وهنا سأضع مثال يوضح طريقة تكوين معيار صحيح مرتبط بنوع البيانات. في حالة البحث عن بيانات نصية فتكون الطريقة بالشكل التالي على اعتبار المتغير rs قد أسند له

RecordSet : خصائص

كود برمجي
rs.Find "[CategoryName] = 'Produce' "

حيث CategoryName هو اسم الحقل و Produce هو البيان المطلوب البحث عنه وإيجاده. وفي حالة البحث عن قيم رقمية أو بيانات رقمية نكون الإجراء بالشكل التالي:

كود برمجي
Rs.Find = " ID = 7 "

حيث ID هو اسم الحقل و 7 هي القيمة المطلوب إيجاده ، والآن أخي العزيز هل لاحظت الفرق ما بين كلا الحالتين. بالطبع الفرق ما بينها هي الفاصلة حيث نستخدم الفاصلة للبحث عن القيم الحرفية أما عن البيانات الرقمية فإننا لا نستخدم لها الفاصلة.

وفي حالة البحث عن بيانات ذات قيم تاريخية فإننا نكون الإجراء بالشكل التالي:

كود برمجي
Rs.Find = " FieldDate = #02/01/2004# "

وهكذا يمكننا بناء الإجراء معتمدين على نوع البيانات فإذا كانت بيانات رقمية لا نستخدم الفاصلة أو إذا كانت بيانات ذات صيغ تاريخية سنستخدم علامة # .

:SkipRows

وتستخدم لتحديد نقطة البداية لعملية البحث . والرقم صفر وهو الرقم الافتراضي لهذه الوسيطة ، وبدون تحديدها ستكون نقطة الانطلاقة لعملية البحث من الصف الحالي المتوقعين عنده (position) .

:SearchDirection

وهي تتمثل في اتجاه الانطلاقة لعملية البحث ويوجد اتجاهين :

* الأول adSearchBackward : البحث من النقطة الحالية للسجل إلى نقطة البداية أو إلى السجل الأول.
* الثاني adSearchForward : البحث من النقطة الحالية للسجل إلى آخر نقطة في السجل أو إلى السجل الأخير .

:Start

وهي شبيهة بالأولى حيث نحدد نقطة البداية من إي صف سيكون البحث (starting position for the search). وبالنسبة للثلاث الوسائط فيه اختيارية الاستخدام يعني يمكننا استخدامها ويمكننا تجاهلها .
والآن سنتناول كتابة الكود الخاص بالإجراء . Find وهنا سوف لن نستخدم (الوسائط) سابقة الذكر
الكود :

كود برمجي
Dim gFind As String, gBookMark As Variant gFind = InputBox("Find What : ", "Search") If Trim(gFind) = "" Then Exit Sub gBookMark = rs.Bookmark rs.Find "[CategoryName] = " & Trim(gFind) & "" If rs.EOF Then rs.Bookmark = gBookMark Exit Sub End If Call ViewRecord

وعند تطبيق الكود السابق سيظهر لنا صندوق الحوار ندخل فيه البيان الذي سنبحث عنه ويجب علينا مراعاة إنه يجب وضع البيان كامل يعني نسخة طبق الأصل وبالشكل الذي تم تخزينه في قواعد البيانات .
ولنفرض أننا أدخلنا بياناً يحتل رقم ثلاثة من عدد السجلات في الجدول ونحن قبل تنفيذ الكود كنا عند النقطة الأولى يعني عند السجل الأول فسيكون التحرك من النقطة الأولى (أي السجل الأول) إلى النقطة الثالثة (أي السجل الثالث) يعني سيكون التحرك نحو الإمام .

والآن نريد إن نبحث عن بيان وهذا البيان موجود في السجل الثاني. جرب الكود هل ستجد نتيجة؟
طبعاً لا. لأن الكود يقوم بالبحث عن البيان في النقطة الحالية ثم يمتد إلى النقاط التالية أو إلى السجلات التي تليه وهكذا مستمراً إلى آخر سجل في الجدول. حاول إن تتخيل العملية لأنها ستفيدنا في فهم الكود وستساعدنا في استغلال (الوسائط) التي ممرنا عليها سابقاً . وباختصار الكود السابق يبحث عن البيانات في السجل

الحالي إلى آخر سجل وإذا كنا في السجل رقم 10 فلا يمكن لهذا الكود الرجوع إلى السجلات التي تليه باحثاً فيها عن البيان المطلوب.

والآن دعنا نجري تغييراً طفيفاً على الكود السابق وتحديدًا عند هذا السطر من الكود:

كود برمجي
rs.Find "[CategoryName] = " & Trim(gFind) & ""

ونغير الكود ليصبح بهذا الشكل:

كود برمجي
rs.Find "[CategoryName] = " & Trim(gFind) & "", 0, adSearchForward, 1

انظر للفرق بين الكودين من ناحية الشكل . الأول لم نستخدم (الوسائط) أما الكود الثاني فقد استخدمنا فيه الوسائط ولكن ما الفرق بين الكودين السابقين في عملية التنفيذ.
الفرق بين الكود :

الأول مثل ما قلنا سابقاً هو إنا منهجية البحث تكون بمسار واحد فقط وهو والبحث في السجل الحالي ومن ثم الانتقال إلى السجلات الألاحقه.

والثاني يبدأ عملية البحث من السجل رقم واحد مرور بجميع السجلات بغض النظر عن النقطة (Position) أو السجل الذي نقف عليه أو عنده وعند كل عملية طلب بحث تكرر العملية حيث يبدأ بالسجل الأول وينتهي عند السجل الأخير . أتوقع أن الفرق الأساسي بين الكودين واضح. ولا تنسى أخي العزيز أن تجري بعض الاختبارات حول الإجراء Find وجميع خواصها أو الوسائط التي تتمع هذا الإجراء. الطرق الأخرى للبحث:

1. عند البحث عن رقم سيكون الكود بهذا الشكل:

كود برمجي
rs.Find "[ID] = " & Trim(gFind) & "", 0, adSearchForward, 1

2. البحث بدلالة حرف معين ، وهنا نستخدم الكلمة المحجوزة LIKE مع علامة النسبة المئوية . % حيث أن هذه العلامة تعني جميع الأحرف بعد هذا الحرف أو هذه الكلمة المدخلة.

كود برمجي
rs.Find "[CategoryName] LIKE " & Trim(gFind) & "%", 0, adSearchForward, 1

وهكذا ... وهنا قد انتهينا من الحديث عن الإجراء Find وبمشينة المولى سنتناول في الحديث القادم طرق بناء واستخدام الإجراء Filter ، كما سنتطرق إلى كيفية استخدام الكلمة المحجوزة WHERE .

جمع وتنسيق / أبوبكر شرف الدين سويدان

2004/10/20 ف