سلسلة دروس VBA

الدرس الأول : الحقات التكرارية (الدورانية) Looping:

في كثير من الأحيان يحتاج المبرمج إلى تكرار تنفيذ مجموعة من التعليمات عدداً من المرات، ويوجد طريقتين أساسيتين للتكرار في حالة كان عدد مرات التكرار معروفة مسبقا حيث يجب استخدام **For ... Next** ، وإذا كان عدد مرات التكرار غير محددة مسبقاً ومبنية على تحقق شرط معين (شرط إنتهاء الدوران أو البقاء به) فهنا نستخدم. **Do ... Loop**

يوجد أربع أشكال لحقات الدوران التكرارية المبنية على شرط وبجميعها يكون هناك شرط يحدد استمرار التكرار أو توقفه. وقيمة الشرط ستكون إما **True** أو **False** ، أو تعبير يرُجع قيمة رقمية حيث تعتبر القيمة التي لا تساوي صفراً **True** وإلا **False** .

أشكال البنى التكرارية الأربعة

١. **تنفيذ التعليمات statements هنا سيستمر حتى يتحقق الشرط condition.**

CODE

```
Do Until condition
Statements
Loop
```

وعندما يصل أكسيس لتفيذ هذا الدوران ، فإنه سيبدأ وقبل كل شيء بفحص قيمة الشرط **condition** ، فإذا كان **True** ، سيتم الإنتقال فوراً إلى ما بعد التكرار أي أنه لن يتم الدخول للحلقة. أما إذا كان **False** ، فسيتم تنفيذ التعليمات **statements**،ثم يرجع للسطر **DO Until** ، للقيام بفحص الشرط ثانية، وهكذا...

إذاً سيتم تكرار الحلقة عدداً من المرات ، طالما بقي الشرط condition قيمته False ولن ينفذ ولو مرة واحدة إذا كان الشرط.True

ويجب في الحقات التكرارية المبني إنتهائها على شرط أن نلاحظ أنها لن تنتهي ما لم يتحقق الشرط ويجب العمل على تحقيقه في داخل الحلقة في لحظة ما وإلا سيستمر الدوران إلى ما لا نهاية ويعلق الجهاز.

ولنأخذ مثالاً حتى نفهم كيف سيتم التعامل مع هذا الشكل من أشكال الحلقات الدورانية:

نريد أن نجمع مجموعة من الأرقام المدخلة إلى الحاسوب ونطبع نتيجة جمعها . عندما ندخل العدد ٩٩٩ نتوقف عن الإدخال.

CODE

```
sum = 0
x= InputBox ("أدخل الرقم ")
do until x=999
sum = sum+x
x= InputBox ("أدخل الرقم ")
loop
Msgbox sum
```

مع ملاحظة ما يلي :

١. أننا قمنا أولا بتحديد قيمة المجموع وهي بالبداية صفراً
٢. ثم قمنا بإدخال القيمة الإبتدائية قبل الدخول للحلقة وهي X ويتم إدخالها من قبل المستخدم عند تشغيل البرنامج.
٣. بناء على قيمة x فإذا كانت ٩٩٩ يكون الشرط **true** ولا ندخل للحلقة وإلا يكون **false** وندخل في داخل الحلقة نقوم بزيادة قيمة sum االمجموع بمقدار القيمة المدخلة وهي. x
٤. ثم تأتي عملية الإدخال من جديد داخل الحلقة والتي بدونها لن ينتهي التكرار حيث من خلال هذه الجملة سيتم وقف التكرار وذلك عند إدخال القيمة ٩٩٩.
٥. وفي نهاية البرنامج نطبع القيمة المطوبة وهي مجموع الأرقام التي تم إدخالها.

٢. الشكل الثاني من أشكال الحلقات سيقوم بتنفيذ الدوران على الأقل مرة واحدة حيث يتم الدخول للحلقة أولا ثم في نهاية الحلقة يتم فحص الشرط:

CODE

```
Do
Statements
Loop Until condition
```

عندما يصل أكسيس لهذه الحلقة يتم الدخول فيها فورا وتنفيذ الجمل **statements** وبعد تنفيذها يتم فحص الشرط **condition** فإذا كان **True**يخرج من الدوران وإلا يبقى وينفذ مرة أخرى وهكذا...

ولنأخذ المثال التالي:

نريد أن نحسب مساحة مجموعة من الدوائر وعندما ندخل أن نصف القطر للدائرة صفراً تتوقف الحقلة (الدوران).

CODE

```
Private Sub ex2_Click()
Do
r = InputBox ( "أدخل نصف القطر " )
a = r ^ 2 * 22 / 7
MsgBox "المساحة = " & a
Loop Until r = 0
End Sub
```

وأرجو ملاحظة ما يلي:

١. تم الدخول للحلقة مباشرة دون شرط أو قيد ولهذا سيتم تنفيذ الحلقة على الأقل مرة واحدة
٢. تم إدخال نصف القطر r حيث سليزم لحساب مساحة الدائرة والتي تساوي مربع نصف القطر مضروبا في النسبة التقريبية **ط** والتي تساوي ٧/٢٢.
٣. يتم احتساب مساحة الدائة وطباعة النتيجة وذلك داخل الحلقة ايضاً.
٤. في نهاية الحقلة يتم السؤال عن الشرط فإذا كانت قيمة نصف القطر r = ٠ فسنخرج من الحلقة والا ننتقل لإدخال نصف قطر جديد وحساب مساحة جديد وهكذا..

٣ . في الشكل الثالث من أشكال الدوران ، تنفذ التعليمات طالما أن قيمة الشرط **condition** صحيحة **true** وإلا يتم التوقف عن الدوران في الحلقة التكرارية.

CODE

```
do while condition
statements
loop
```

عندما يصل الأكسيس إلى هذه الحلقة يفحص قيمة الشرط أولا وقبل الدخل للدوران فإذا كان صحيحاً يستمر في الدوران وإلا يخرج من الدوران.

ولنأخذ المثال التالي:

نريد أن نقرأ مجموعة من الأسماء ونطبعها وإذا كان الحرف الأول في الكلمة ع (حرف العين) نتوقف.

CODE

```
Private Sub ex3_Click()
na = InputBox("أدخل أسم")
Do While Left (na, 1) <> "ع"
MsgBox na
na = InputBox ( "أدخل أسم " )
Loop
End Sub
```

ومن المهم ملاحظة ما يلي:

١ . يتم إدخال اسم ابتدائي قبل الدخول للحلقة حيث سيتم فحص الشرط مباشرة
٢ . لاحظ أن الشرط هو بالسؤال عن الحرف الأول من اليسار Left للاسم وإذا لم يكن يساوي حرف العين نستمر في الدوران وعندما ندخل اسم يكون أول حرف فيه عين يتم التوقف والخروج من الدوران.
٣ . في داخل الدوران سيتم طباعة النتيجة المطلوبة وهي الأسم ثم سيتم السؤال عن الأسم التالي تمهيدا لتنفيذ الدورة الثانية وهكذا...

٤ . الشكر الرابع والأخير من أشكال الحلقات الدورانية المشروطة يتم فيه تنفيذ التعليمات مرة واحدة على الأقل وهو كما يلي:

CODE

```
Do
statements
Loop While condition
```

وهنا عندما يثل الأكسيس إلى الحلقة يتم الدخول مباشرة لها وتنفيذ الجمل التي بها **statements** ثم يتم السؤال عن صحة الشرط **condition** حيث إذا كان **true** يستمر في الدوران وإلا يتم الخروج منه.

ولنأخذ المثال التالي وهو حساب مجموع مربعات (تربيع) الأعداد المدخلة . نتوقف عن الحساب ونطبع النتيجة المجموع عندما ندخل العدد ٠.

CODE

```
Private Sub ex5_Click()
sum = 0
Do
x = InputBox ( "أدخل رقم " )
sum = sum + x ^ 2
Loop While Not x = 0
MsgBox "مجموع مربعات الأعداد المدخلة هو " & sum
End Sub
```

لاحظ أن شرط الإنتهاء هو ٠ = not x وهو مكافىء للشرط ٠ <>x حيث الرمز <> يعني لا يساوي.

لاحظ أيضا أنه في جميع الأشكال الأربعة الساقة لم يحدد في الأمثلة عدد مرات التنفيذ وهكذا تستخدم الحلقات الدورانية في هذه الأشكال عندما لا يعرف عدد مرات التنفيذ) الدوران) مسبقاً.

أما إذا كنا نعلم عدد مرات التنفيذ (الدوران) فإنه من غير المفضل نهائياً استخدام الأشكال السابقة ويجب استخدام الشكل الخاص بالتكرار المروف مرات الدوران مسبقا وهو **For .. do** التالي:

CODE

```
For counter = start to end [Step increment]
statements
Next [Counter]
```

حيث أن:

counter : متغير (عداد الحلقة التكرارية)
start : القيمة الإبتدائية للدوران
end : القيمة النهائية للدوران
increment : مقدار الزيادة في كل مرة من مرات الدوران وإذا لم يوضع تكون الزيادة بشكل معياري هي ١

ويمكن أن يكون **increment** موجب أو سالب فإذا كان سالب يجب أن تكون القيمة النهائية أضغر من الإبتدائية.

وإليكم كيفية تعامل أكسيس مع هذا النوع من الحلقات:

١. عند الدخول للدوران **for** يتم وضع قيمة **counter** تساوي القيمة الإبتدائية **start**
٢. يتم ومباشرة وقبل تنفيذ الجمل التي تلو for إذا كانت قيمة العداد counter أكبر من القيمة النهائية end فإن كانت كذلك يتم إنهاء الدوران والخروج منه.
أما إذا كانت قيمة **Increment** سالبة فإن الفحص يتم بطريقة عكسية حيث يتم التأكد فيما إذا كانت قيمة العداد **counter** أقل من القيمة الإبتدائية **start** وإذا كانت كذلك يتم الخروج من الحلقة.
٣. يتم تنفيذ الجمل التي في الحلقة
٤. يتم زيادة قيمة العداد بمقدا واحد أو **increment** إذا كان محدداً.
٥. وهكذا يتم تكرار الخطوات من ٢ إلى ٤ في كل مرة.

ولنأخذ المثال التالي نفرض أننا نريد جمع الأعداد الزوجية من ١ - ١٠٠ وطباعة النتيجة:

CODE

```
Private Sub ex6_Click()
sum = 0
For i = 1 To 100 Step 2
sum = sum + i
Next i
MsgBox "المجموع هو " & sum
End Sub
```

وفي مثالنا الثاني نريد أن نقرأ اسم نموذج فإذا كان مفتوح نطبع أن النموذج مفتوح وإلا نطبع غير مفتوح

CODE

```
Private Sub ex7_Click()
n = InputBox("أدخل اسم النموذج")
IsOpen = False
For i = 0 To Forms.Count - 1
If Forms(i).FormName = n Then
 IsOpen = True
End If
Next i
If IsOpen Then
MsgBox " مفتوح النموذج"
Else
MsgBox "النموذج غير مفتوح"
End If
End Sub
```

لاحظ أن للنماذج المفتوحة رقم يبدأ من صفر **forms.count**
ولاحظ اسم النموذج **forms(i). formname**

أما باقي الأمور في الحلقة فأعتقد أنها سهلة.

ملحق الدرس الأول:

بالنسبة للحلقات المشروطة لا يوجد طرق أخرى.

وقاعدة كتابة هذه الجمل بشكلها المتكامل هو كما يلي:

CODE

```
Syntax

Do [{While | Until} condition]
[statements]
[Exit Do]
[statements]

Loop

Or, you can use this syntax:

Do
[statements]
[Exit Do]
[statements]

Loop [{While | Until} condition]
```

بالنسبة للحلقات غير المشروطة على غرار **next** ... for يوجد أيضا تركيب أخر مهم لا نستطيع تجاهله وهو **for each ... next** وكمثال عليه أقدم لكم ما يلي:

هذا الموضوع يتماشى مع Access ٢٠٠٢ وAccess ٢٠٠٣

يستفاد من هذا الموضوع معرفة طريقة إضافة الطابعات المحملة (المعرفة) لجهاز الحاسوب إلى مربع تحرير وسرد (combo box) باستخدام البرمجة بلغة. (VBA)

خطوات الطريقة هي:

١. ندخل إلى خصائص مربع التحرير والسرد ونضغط لسان التبويب الكل.(All tab)
٢. نغير قيم الخصائص كما يلي:

نضع في خاصية الاسم (Name) الاسم dPrinter

نختار في خاصية نوع مصدر الصف (Row Source Type) قائمة الحقول (Value List)

نضع في حدث عند التحميل للنموذج (On Load) الكود التالي:

CODE

```
Private Sub Form_Load()
Dim prt As Printer
For Each prt In Application.Printers
Me!dPrinter.AddItem prt.DeviceName
Next
End Sub
```

ملاحظة (١) : عندما نضيف (نحمل) عناصر لمربع التحرير والسرد بالبرمجة (كما في المثال السابق) فإن العناصر لا تترتب أبجديا وإنما حسب ترتيب التحميل طبعا هناك حل لهذا الأمر أيضا، أرجو من الإخوة الأعضاء التفكير في الحل.

ملاحظة (٢) : انتبه إلى طريقة استخدام**.(AddItem)**

الخلاصة : تم إضافة الطابعات المعرفة إلى مربع التحرير والسرد برمجيا وهذا مجرد مثال لأفكار كثيرة يمكن تطبيقها بنفس التقنية والشيء الممتع في هذه التقنية أن نتيجتها متواصلة التغير **(Dynamic)** أي كلما تغيرت تعريفات الطابعات تتغير محتويات مربع التحرير والسرد.

الملحق ٢ : نهاية درس الحلقات التكرارية.

١. يجدر الإشارة إلى تركيب أخر دوراني يندر استعماله حيث أن كفاءة الدورانات السابقة أكثر منه ولأنه لا يحتوي على طريقة للخروج إلا بتحقق شرطه وهو التركيب التالي:

CODE

```
While condition
[statements]
Wend
```

ويعمل طالما تحقق الشرط **condition** وبقي محققا يعني true فسيستمر تنفيذ التعليمات **statements** وعندما يصبح الشرط **false** يتنهي الدوران

ولنأخذ مثالا عليه:
لو أردنا أن نقوم بقراءة مجموعة كلمات وطباعة هذه الكلمات في النهاية على شكل جملة بحيث تنتهي القراءة عند إدخال كلمة توقف فيكون المثال كما يلي:

CODE

```
Private Sub PrintStatement_Click()
Dim W, Stat As String
Stat = ""
W = ""
While Not W = "توقف"
W = InputBox("أدخل كلمة")
Stat = Stat & " " & W
Wend
MsgBox Left(Stat, Len(Stat) - 4)
End Sub
```

لاحظوا أننا طبعنا في النهاية قيمة state المتغير النصي الذي سيحتوي على كل الكلمات ولكن - ٤ أي بدون الأربعة الحروف الأخيرة لأنها ستكون الكلمة توقف.

٢. الخروج من الحلقات الدورانية:
بشكل عام وكما لاحظنا في الدرس الأول فإن الخروج من الحلقة المشروطة يتم عندما يتحقق الشرط المطلوب للخروج وكذلك الحال بالنسبة للحلقات غير المشروطة **for ... next** فإن الخروج يتم عند اتمام عدد المرات للدوران.
يمكن في كل الأحول ما عدا في **(while ... wend)** استخدام الخروج المباشر **exit for** أو **exit do** ومن أي مكان داخل الحلقة ، ويتم الخروج إلى الجملة التي تلي الحلقة مباشرة.
فمثلا في مثال تحديد فيما إذا كان نموذج ما (معروف اسمه) مفتوحا أو لا سيترتب علينا عمل دورة ندور بها في كل النماذج ونفحص بها إذا

كان النموذج مفتوحا أو لا ، والسؤال المفيد الذي يطرح نفسه هنا لماذ نستمر في الدوران إذا وجدنا النموذج المطلوب مفتوحاً، بكل تأكيد أن الجواب هو لا داعي للإستمرار في الدوران لأن ذلك سيكون مضيعة للوقت ، ويجب الخروج الفوري من الدوران . وللنظر إلى الإقتران التالي

CODE

```
Function IsOpen(ByVal MyFormName As String) As Boolean
Dim i
IsOpen = False
For i = 0 To Forms.Count - 1
If Forms(i).FormName = MyFormName Then
 IsOpen = True
 Exit For
End If
Next i
End Function
```

وطريقة طلب الإقتران ستكون كما يلي:

CODE

```
Private Sub FormCmd_Click()
MsgBox IsOpen("form١")
End Sub
```

لاحظ أيضا أن هذا المثال يبين أنه يمكن استخدام exit من داخل if ، وكذلك يمكن استعمالها داخل جملة select إذا كان داخل دوران.

٣. التكرار (الدوران) المتداخل**Nested Loops**

مما لا شك فيه أن الدوران يمكن أن يتداخل ولنأخذ مثال على ذلك : نريد أن نقرأ من المستخدم كلمة فإذا كانت واحد يطلب منا إدخال رقم وإذا كانت أثنين يطلب منا إدخال رقمين وهكذا حتى خمسة وإذا طبعنا أي شيء غير واحد إلى خمسة يتوقف العمل وفي النهاية يطبع لنا مجموع جميع الأرقام التي تم إدخالها

CODE

```
Private Sub AddAll_Click()
Dim w As String
Dim out As Boolean
sum = ٠
Do
w = InputBox ( " الرجاء ادخال عدد الارقام التي تريد جمعها واحد الى خمسة او اي شئ آخر " )
Select Case w
Case  " واحد ": last = ١
Case  " اثنين ": last = ٢
Case  " ثلاثة ": last = ٣
Case  " اربعة ": last = ٤
Case  " خمسة ": last = ٥
Case Else: last = ٠
          out = True
End Select
 For i = ١ To last
sum = sum + InputBox ("ادخل رقم")
 Next i
Loop Until out
MsgBox "المجموع هو " & sum
End Sub
```

لاحظوا أنه في case else
١. تم تعيين قيمة **out = true** وذلك للخروج من الحلقة الخارجية المسؤولة عن إدخال الكلمات حيث ان هذه الحالة case يتم الدخول إليها في حال أدخلنا كلمة غير من واحد إلى خمسة.
٢. تم تعيين قيمة **last =** ٠ حتى لا يتم تنفيذ الدوران **for .. next** لأنه يبدأ من ١.

الدرس الثاني : التعامل مع مجموعات التسجيلات

قبل أن نخوض في تفاصيل وطرق التعامل مع مجموعة التسجيلات يجب أن نعرف ما هو المقصود بمجموعة التسجيلات والتي يسميها البعض أيضا بمجموعة السجلات ، **والتي هي عبارة عن مجموعة السجلات محور الإهتمام في نموذج أو تقرير أو عنصر تحكم في مربع نص أو عنصر تحكم مربع تحرير وسرد أو متغير في وحدة نمطية** ، (والحقيقة لا أجد لها أماكن استخدام غير الخمسة المذكورة) وهذه السجلات في شكلها البسيط ممكن أن تكون **جدول** واحد مؤلف من عدة حقول ، وممكن أن تكون مبنية من **عدة جداول** وبناء على شروط قد تكون معقدة، وتعتبر جميع نتائج **الاستعلامات** أو تعليمات **SQL** مجموعات تسجيلات.

مجموعة التسجيلات Recordset هي كائن ، **يعتبر من أكثر وأهم الكائنات استعمالا في VBA، ولذا أعزائي الأعضاء ، أحببت أن يكون شرحه في مقدمة دروسنا في.VBA**

ودرسنا اليوم تحديداً عن الكائن **Recordset** في**. DAO : Data Access Object.**

يوجد في هذا الكائن المهم أكثر من ٣٠ خاصية و ١٨طريقة (أمر) وتقريباً كل قوة محرك بيانات أكسيس Jet ، في عمليات الإضافة والحذف والتعديل والبحث والفرز والتصفية على البيانات تكمن في الكائن.Recordset

وقبل البدء بشرح كيفية التعامل مع مجموعة التسجيلات أرجو منكم أن تحفظوا الجملة التالية التي لن نفهمها تماماً إلا بعد اتمام الدرس كاملا والذي هو طويل نوعاً ما وتجدر الإشارة إلى هذه الجملة المهمة الأن وقبل البداية:

"إن سلوك القراءة والكتابة في مجموعة التسجيلات يعتمد على نوع الكائن (ADO أو DAO) ويعتمد كذلك على نوع البيانات (SQL او Jet) المعرفة بواسطة الكائن Recordset"

وسأراجع معكم هذه الجملة في أخر الدرس التي بإذن الله عندها ستكون من الخبراء في استعمال مجموعات التسجيلات.

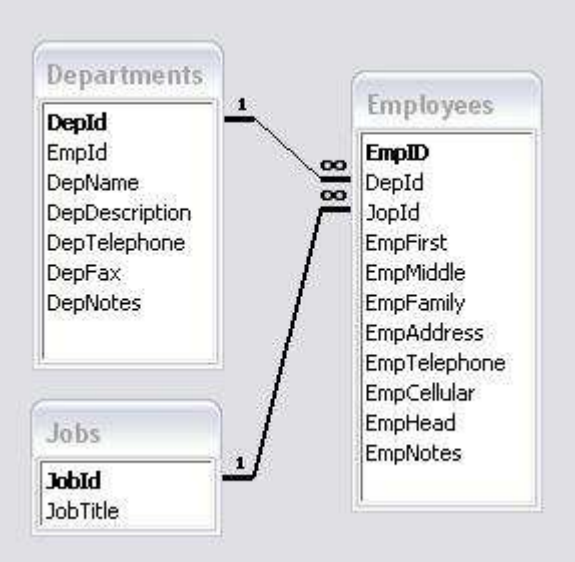
وبكل تأكيد أن التعامل مع مجموعة التسجيلات بتقنية **ADO** هي الأحدث ولن نغفل ذلك نهائيا وسأحاول (حيثما أمكن) وضع الأمثلة بطريقة **ADO**وكذلك **DAO** وما هذه إلا البداية والله الموفق ، علما أنني بذلك أحاول ربط القديم بالجديد وفي ذلك فوائد عديدة منها إلقاء الضوء على الطريقتين وتشجيع البعض ممن لم يتعامل مع أحدى التقنيات أن يستعملها أو على الأقل يستفيد منها في فهم الأمثلة التي تملىء المنتديات والتي فيها مزيجا من هذه التقنيات ولا أخفيكم أن هذه الفكرة قد تبادرت إلى ذهني بعد التفكير ملياً بأحسن الوسائل لطرح الموضوع بما فيه مصلحة الأعضاء العامة وبتشجيع من إخواني الخبراء في جامعة أهلا عرب وبعد استشارة أخي أبا هادي من أوفيسنا جزاهم الله خيراً.

يوجد أربعة أنواع من مجموعات التسجيلات فيDAO

١. النوع :**Table** ويستخدم لتحديد جدول في قاعدة البيانات الحالية فقط ، ولا يمكن استخدامه لتحديد استعلام أو جدول مرتبط من قاعدة أخرى ، ويمكن أن يستخدم هذا النوع للإضافة، الحذف ، التعديل ، أو البحث في الجدول، وعندما نستخدم هذا النوع فإن التغيرات والتعامل تكون مع الجدول مباشرة ، ومن مميزاته الحسنة أنه يسمح باستخدام أسرع طريقة للبحث عن التسجلات بالأمر seek الذي يبحث في الفهارس المعرفة على الجدول.

إخواني وأخواتي ، لا أريد أن أطيل عليكم بالتفاصيل النظرية عن جميع الأنواع والنظريات المتعلقة بالموضوع لأنه طويل جدا وستملون الموضوع بك تأكيد ، لذا سأنتقل مباشرة إلى التطبيق العملي حتى نفهم ما توصلنا إليه حتى الأن ونستوعبه ونطبقه عملياً.

بشاشة العلاقات بالاسفل ٣ جداول فقط الموظفين (**employees**) والأقسام (**departments**) والوظائف (**jobs**) بحيث أن لكل قسم مسؤولا (موظفاً) واحدا فقط ولكل موظف مسئول مباشر عنه وكل موظف يتبع لقسم ما ولكل موظف وظيفة محددة.



أرجو منكم تفحص تصميم الجداول وخصائص الحقول جيداً.

ما سبق هو ليس موضوعنا الأساسي ... الآن ننتقل إلى الموضوع:

في مثالنا الأول نريد أن نبحث عن رقم موظف وإذا كان موجودا في جدول الموظفين نطبع اسم الموظف وعائلته وإذا لم يكن موجودا نطبع أن الرقم المدخل غير موجود ، ومن هنا نبدأ فهمنا لمجموعة التسجيلات...

CODE

```
Private Sub example١_Click()

Dim MyDb As Database
Dim rstEmployees As Recordset

Set MyDb = CurrentDb
Set rstEmployees = MyDb.OpenRecordset("employees")

Id = InputBox ( " ادخل رقم الموظف " )

rstEmployees.Index = "PrimaryKey"
rstEmployees.Seek "=", id
If rstEmployees.NoMatch Then
MsgBox ( " ادخل رقم الموظف " )
Else
MsgBox rstEmployees!EmpFirst & " " & rstEmployees!empfamily
End If

rstEmployees.Close

End Sub
```

وإليكم الملاحظات المهمة التالية:

١. بناء المتغير من نوع **recordset** هو عن طريق **OpenRecordset** ويوجد لها شكلان واحد لبناء مجموعة التسجيلات من قاعدة البيانات كما في مثالنا والثاني لبناء مجموعة التسجيلات من **TableDef** أو **QueryDef** أو من **Recordset** وأعدكم أنني لن أتشعب كثيرا حيث الموضوع طويل وسأركز حديثي في الغالب حول الأمثلة التي أطرحها ، وبإمكانكم بعد فهم الأساسيات التوسع بالموضوع بكل سهولة.

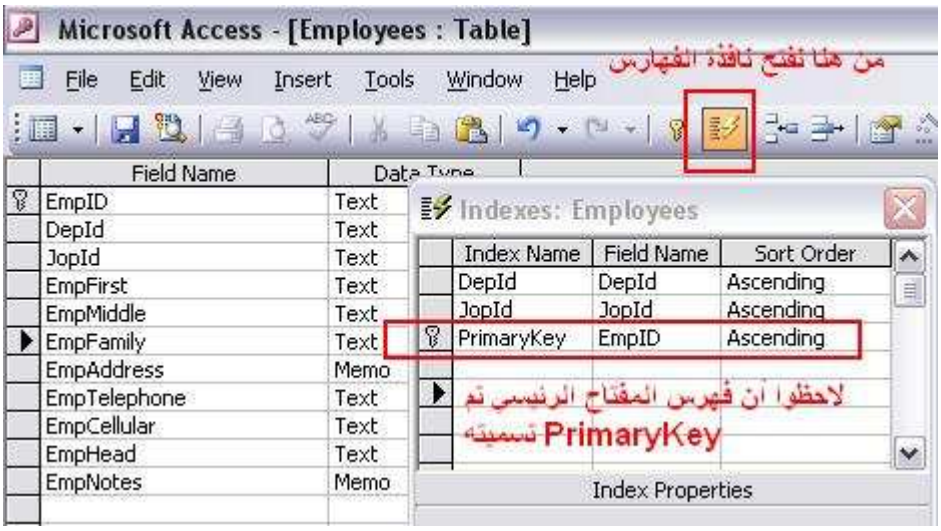
الشكل العام لطريقة بناء المتغير والتي استعملناها في مثالنا هي:

CODE
set rst = database.openrecordset (source,type,options,lockedits)

حيث

١- **rst** اسم المتغير من النوع **recordset**
٢- **database** اسم المتغير من نوع **database**
٣- source متغير نصي ، اسم جدول أو استعلام او أمر **sql** ترجع سجلات. أما في مثالنا حيث النوع هو **table** ، فيجب أن يكون **source**اسم جدول
٤- type وهو ثابت رقمي يحدد نوع المجموعة من (**dbopentable**، **dbopendynaset**،**dpopensnapshot** ، **dbopenforwardonly**)ويوجد نوع خامس يستخدم فقط مع **ODBCDIRECT** وهو**dbopendynamic**
مع ملاحظة أنه إذا فتحنا مجموعة سجلات بدون أن نحدد نوعها كما في مثالنا فإن المحرك **Jet** سينشيء المجموعة بشكل تلقائي من النوع **table** ، إذا كان المصدر جدولا ، أما إذا كان المصدر استعلاما أو جدولا مرتبطا فإن النوع سيصبح.**dynaset**
٥- **options** هي مجموعة من الثوابت الإختيارية التي تحدد خصائص مجموعة التسجيلات في بيئة تعدد المستخدمين.
٦- **lockedits** أيضا ثابت اختياري يحدد قفل التسجيلات أم لا.

٢. تحديد الفهرس تمهيدا لإستخدام الأمر seek وإذكر مرة أخرى هنا أن هذه الطريقة تستخدم للبحث عن تسجيلة في مجموعة التسجيلات من النوع Table ولا يمكن استعمالها مع باقي الأنواع، وتقوم هذه الطريقة بالبحث فقط وفقاً لفهارس الجدول المعرفة أو التي يمكن تعريفها أيضا برمجياً وقبل كل شيء لنلقي نظرة على فهارس جدول الموظفين كما في الصورة:



ولهذا قمنا بوضع اسم الفهرس **PrimaryKey** علما أننا نستطيع تغيير هذا الأسم في وضع تصميم الجدول ومن نافذة الفهارس كما في الصورة.

لاحظوا أيضا أن **rstemployees.Index** تدعى **خاصية** الفهرس

ومرة أخرى أذكر أن للكائن **recordset** ولاي كائن يوجد خصائص (**properties**) و طرق (**Methods**) يعني أوامر.

السؤال الذي يطرح نفسه هنا ماذا يحدث لو لم نقم بضبط خاصية الفهرس قبل استخدام الأمر seek ، سينتج الخطأ التالي:

Operation invalid without a current index

وعلماً أنه يمكن البحث عن تسجيلة وفق فهرس يتألف من أكثر من حقل (حتى ١٣ حقلا) وسنأخذ مثال على ذلك لاحقاً بإذن الله.

٣. الأن الوضع مرتب لإجراء البحث ويتم ذلك من خلال الطريقة (الأمر) method واسمها seek بالطرية التالية

rstEmployees.Seek "=", id

حيث id هي القيمة التي تم قراءتها والأن سنبحث عنها ويمكن البحث بعدة طرق "=" ، ">" ، "<" ، "=>" ، "=<"

٤. بعد الأمر السابق وهو أمر البحث يتم عمل شيئين بشكل تلقائي وفي الخفاء وهما:

- يتم الإنتقال غلى التسجيلة التي وجدت حسب البحث (إن وجدت)
- يتم تعريف قيمة الخاصية **nomatch** بناء على النتيجة إذا وجد السجل يكون فيها **false** وإذا لم يوجد يكون فيها**true**

٥. الأن الأمر اصبح في غاية البساطة ولأننا نبحث في مفتاح رئيسي فلا حاجة للبحث إلا مرة واحدة فإن وجد الرقم فالسجل موجود وإلا فغير موجود.

٦. في نهاية العمل لا تنسوا دائما إغلاق مجموعة التسجيلات بإستخدام الأمر.**close**

في هذا الجزء من الدرس سيتم إلقاء الضوء على تقنيات طرق التعامل مع مجموعات التسجيلات المختلفة في فيجوال بيسك وهي بشكل من الجديد إلى القديم **ADO** و **DAO** و: **RDO**
في فيجول بيسك ، يتوفر هذه الواجهات (تقنيات اتصال) للوصول للبيانات:

١. ActiveX Data Objects (ADO)
٢. Data Access Objects (DAO)
٣. Remote Data Objects (RDO)

وكل من هذه الثلاثة تمثل كائن من كائنات الوصول للبيانات بوجوه مختلفة ، وباستخدام **VBA** بإمكانك برمجيا التحكم بالاتصال ، بنى الجمل ، والبيانات التي سيتم إرجاعها للاستخدام في التطبيق .

السؤال هو ؟ لماذا هناك ٣ طرق للوصول للبيانات في الفيجوال بيسك ؟ من المعروف أن طريق الوصول للبيانات تتطور بشكل متواصل وكل طريقة من هذه الطرق تمثل مرحلة من المراحل الفنية في هذا التطور . الأخيرة (الأحدث) هي **ADO** ، والتي من خصائصها البساطة - ورغم البساطة الأكثر مرونة – من كلا **RDO** أو . **DAO** للمشاريع الجديدة ، ينصح بشدة اعتماد **ADO** كواجهة (تقنية) الوصول للبيانات.
هذه النصيحة ليست من خضر الرجبي ولكن من ميكروسوفت نفسها.

أما بالنسبة لي فكما ذكرت سابقا أن أنصح الجميع بمعرفة القديم والجديد وقد وضعت المبررات لذلك وأنا ماض بتعريفكم على **ADO** و **DAO**بإذن الله كما وعدت سابقا.

لماذا نستخدم ADO ؟

تم تصميم ADO بأبسط مستوى اتصال في ميكروسوفت مع OLE DB ، و OLE DB توفر كفاءة عالية في الوصول إلى أي مصدر بيانات ، سواءً في قواعد البيانات العلائقية أو غير العلائقية (لعلمكم يوجد قواعد بيانات تسمى الشبكية و المهيكلة وهي الجيل السابق من قواعد البيانات) ، وكذلك من مصادر البيانات أنظمة الملفات والبريد الإلكتروني ، والبيانات النصية والرسومية ، وكائنات الأعمال المخصصة ، وغيرها من المصادر .

ADOصممت لملائمة الحد الأدنى من التراسل عندما نتعامل مع بيئة وسيناريوهات الإنترنت ، وأيضا تضع الحد الأدني من الطبقات بين واجهة التطبيق النهائية يعني للمستخدم وبين مصدر البيانات – كل ذلك ضمن واجهة خفيفة ذات كفاءة عالية. وكذلك **ADO** تستعمل طرق محادثة وخصائص مشابهة لتلك في **DAO** و **RDO** ، مع تبسيط منطق هذه الطرق وذلك من أجل التسهيل على المستخدم.

ماذا بشأن DAO و RDO ؟

من أجل التوافقية (compatibility) ولدعم المشاريع السابقة استمرت فيجوال بيسك بدعم DAO و RDO للمشاريع القائمة.

مقارنة ADO مع RDO و DAO :

ADOليست متواقة - الكود تلقائيا مع تطبيقاتك التي تستعمل طرق الوصول للبيانات السابقة . بينما كبسلة (**encapsulation**)تتضمن العمليات المتوفرة في كل من **ADO** و **RDO** ، يجب عليك تحويل الكثير من عناصر اللغة لاستعمال قواعد **ADO**في مشروع كأن يستخدم **DAO** إن أحببت التحويل. في بعض الحالات ، هذا يعني تحويل بعض الإقترانات فقط ، وفي البعض الأخر قد تحتاج لبناء التطبيق كاملا من جديد باستخدام خصائص **ADO** الجديدة.

Data Access Objects (DAO) : هي أول واجهة كائنات - مرئية **object-oriented** تعامل به محرك قاعدة بيانات ميكروسوفت **Jet**والتي مكنت مستخدمي فيجوال بيسك من الوصول إلى الجداول الأكسيس مباشرة وكذلك لقواعد بيانات أخرى من خلال . **ODBC** لا شك أن **DAO** هي الأنسب لتطبيق يعمل على جهاز واحد وكذلك الأنسب لمؤسسة صغير منفردة.

Remote Data Objects (RDO) هو كائن-مرئي يستخدم في التواصل مع **ODBC** بشكل موحد ومستثمرا لسهولة**DAO** ، للكائن **RDO**حدوديات ، ورغم ذلك ، أثبت أفضلية استعماله مع بعض المحركات المشهورة مثل **Oracle** , **Sql Server**.

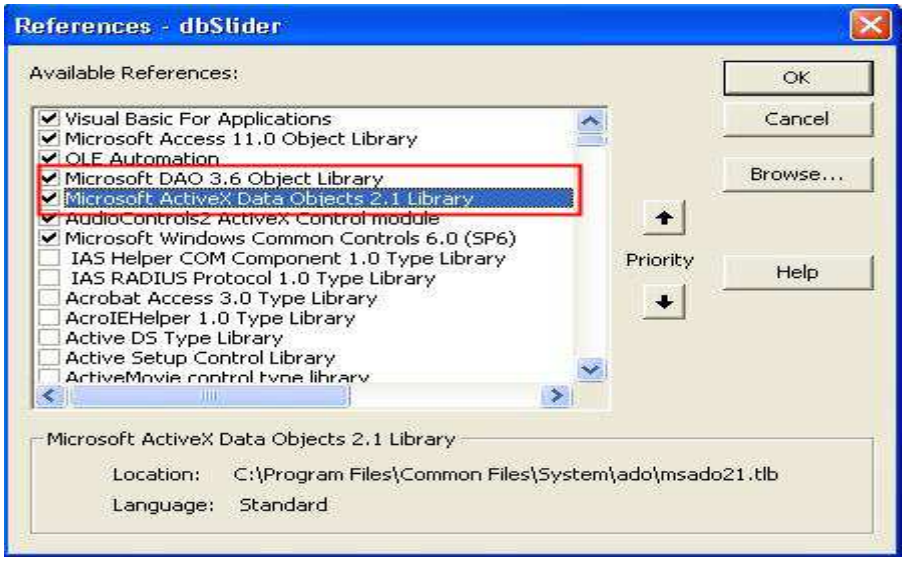
ADOهو الجيل الجديد واللاحق . **DAO/RDO** عمل **ADO** أقرب ما يكون من **RDO** ، وعادة يوجد الكثير من التشابه في طرق الاستعمال. نستطيع القول بأن **ADO** حل محل الكائنين معا . ومن الممكن في البداية أن تظن أن التعامل مع **ADO** صعبا إلا أنك ستكتشف بإن طريقة بنائه المهيكلة ستجعل من السهولة بمكان التعامل معه ببساطة شديدة.

من المهم ملاحظة أيضا أن **ADO** حالياً لا يدعم جميع إقترانات **DAO** ، وهو يحتوي اسلوب الدعم أكثر المتوفر في **RDO** للتفاعل مع مصادر بيانات **OLE DB** بالإضافة لدعم المصادر البعيدة وكذلك تكنولوجية **DHTML** .

بشكل عام ، ربما ما زال مبكرا الرحيل إلى استعمال **ADO** بشكل كامل (بإستثناء أولئك الذين يستعملون **ODBCDirct**) ، حيث أن **ADO** لا تدعم بشكل كامل (**data definition (DDL** ، المستخدمين, المجموعات ، على كل الأحول إذا كنت تستخدم **DAO** فقط في تطبيقات client-serverالخادم - الزبون ولا تعتمد عليه في لغة تعريف البيانات بشكل اساسي ، فربما حان الوقت.

خلاصة القول : وهذا رأيي الشخصي ، أنه قد حان الوقت بالتأكيد لمن لم يعرف عن الاثنين للتعرف على الاثنين معا حتى يواكب القديم والحالي والقادم من التطبيقات.

قبل أن نكمل درسنا في التعمق في طريقة التعامل مع مجموعات التسجيلات وعرض طرق أخرى لتعريف مجموعة التسجيلات **recordset** بأستخدام الكيان **DAO** ، أود أن نلقي الضوء قليلا وعلى نفس المثال السابق وهو الوصول إلى سجل محدد ولكن هذه المرة باستخدام تقنية **ADO**، ولكن قبل عمل ذلك أود أن أشير إلى أهمية الانتباه إلى المراجع حيث مرجعي **ADO** و **DAO** يوجد بهما نفس التسمية للكيانات مثل **recordset** وكيف نتغلب على تشابه التسمية.



من شاشة محرر فيجوال بيسك ... **(ALT+F١١)** ثم ... **tools** ثم **References** يظهر لنا الصندوق الحواري كما في الصورة لاحظوا أننا اخترنا **DAO** وكذلك **ADO** في نفس الوقت وهذا يعني أنه يمكن استخدام جميع الكيانات فيهما الأثنين ومنها الكيان **recordset** فماذا سيعتبر في البرنامج (الوحدة النمطية) هل سيتبع إلى **DAO** أو إلى **ADO** ، في الحقيقة أن أسهل طريقة وحتى لا يحدث لبس هي تعريفه بالشكل التالي **ADODB.Recordset** أو **DAO.Recordset** ام اذا عرفناه بدون تحديد فهو سيتبع للمرجع الذي يظهر أولا وفي هذه الحالة كما في الصورة سيتبع إلى **DAO** حيث هو المعرف قبل.**ADO**

الآن أنتقل بكم مباشرة إلى تطبيق المثال المكافيء للمثال الأول وعلى نفس القاعدة السابقة قاعدة الموظفين التي تم إرفاقها وللحصول على معلومات الموظف مباشرة ولكن بإستخدام تقنية **ADO** .

CODE

```
Private Sub example١_Click()

Dim Cnn As ADODB.Connection
Dim rstEmployees As ADODB.Recordset
Dim strCnn As String

Set Cnn = New ADODB.Connection
strCnn = CurrentProject.Connection
Cnn.Open strCnn

Set rstEmployees = New ADODB.Recordset
rstEmployees.Open "employees", Cnn, ٣, ١, adCmdTableDirect 'read only and allow indexing

id = InputBox  ( " ادخل رقم الموظف " )
rstEmployees.Index = "PrimaryKey"
rstEmployees.Seek id, adSeekFirstEQ

If rstEmployees.EOF Then
MsgBox " الرقم غير موجود "
Else
MsgBox rstEmployees!empfirst & " " & rstEmployees!empfamily
End If

rstEmployees.Close
Set Cnn = Nothing
Set rstEmployees = Nothing

End Sub
```

الملاحظات:

١. تمهيدا للوصول للجدول **employees** الموجود في قاعدة البيانات الحالية وليست في مصادر بيانات خارجية أو قاعدة أخري نعرف أولا ما يسمى نص الإتصال **connection string** وهو مصدر البيانات وهنا الأمر في غاية البساطة فهو في نفس القاعدة ولذا أشرنا له بالنص **currentproject.connection** والمكافئة تماما في تقنية **DAO** للنص**CurrentDb**
ومن الضروري جدا فتح الاتصال وهذا تم في الجملة **Cnn.Open strCnn**

٢. ثم نقوم بفتح (تعريف) مجموعة التسجيلات والتي تتم من خلال الخاصية **OPEN** كما في **rstEmployees.Open "employees", Cnn, ٣, ١, adCmdTableDirect 'read only and**

والشكل العام لخاصية الفتح هو:

recordset.Open Source, ActiveConnection, CursorType, LockType, Options

علماً أن كلمة **cursor** هنا يعنى بها عنصرا من قاعدة البيانات التي تتحكم في التنقل بين السجلات ، تعديل البيانات ، وظهور التغيرات من المستخدمين للقاعدة.
وطبعا يوجد هنا كما في **DAO** أيضا أربعة طرق مختلفة لفتح مجموعة التسجيلات وهي: **CursorType**
adOpenStatic = ٣، adOpenKeyset = ١، adOpenForwardOnly= ٠٠، adOpenDynamic = ٢٠ ،
والتي سنتعرض لها بالتدريج ، ما تم استخدامه في هذا المثال هو ٣ وهو يسمى **static cursor** وهو عبارة عن نسخة ساكنة عن السجلات للوصول وإيجاد البيانات دونما الأهتمام برؤية الإضافات والتغييرات والحذف الآني من المستخدمين.
LockType : نوع إقفال السجلات وما استخدامناه في مثالنا هو ١ **adLockReadOnly** ويعني أن البيانات للقراءة فقط ولا نستطيع التعديل عليها.

options : الخيارات التي تحدد كيف سيقوم مزود البيانات بمعالجتها وفي مثالنا استخدمنا adCmdTableDirect وبهذا الخيار فقط يمكن استعمال الخاصية seek في ADO وقيمة هذا الثابت ، **يعني adCmdTableDirect هي ٥١٢ . تذكروها جيدا حيث لا يوجد طريقة غيرها.**

٣. الآن البيئة مهيئة تماما لما نريد تحقيقه في المثال نقوم بتحديد الفهرس كما في خاصية الفهرس

CODE
rstEmployees.Index = "PrimaryKey"

ولا تنسوا حكاية اسم الفهرس فهي نفس الحكاية التي تم شرحها أعلاه في. DAO

٤. ثم نقوم بقراءة رقم الموظف وحاولة الوصول إليه من خلال الخاصية
rstEmployees.Seek id, adSeekFirstEQ
٥. لاحظوا هنا الفرق بين هذه الطريقة وطريقة DAO حيث المعاملات تم عكسها علما أن عمليات المقارنة كانت هناك "=" بينما هنا أصبحت كما يلي وحسب مثالنا:
١ **adSeekFirstEQ**البحث عن أول موظف له الرقم المحدد.
٦. بالنسبة **nomatch** لا توجد هذه الخاصية في **ADO** وعوضا عنها نستخدم فكرة أن البحث إذا لم يسفر عن نتيجة فإن مؤشر القراءة الوهمي سيصبح نهاية الملف وهكذا نتصرف ونسأل عن خاصية نهاية الملف **rstEmployees.EOF** إذا كانت true فهذا يعني أننا لم نجد أي سجل يوافق البحث.
٧. هذا كل شيء لا تنسوا إغلاق مجموعة التسجيلات في نهاية العمل كما فعلنا أيضا مع **DAO** .

أليست البرمجة رائعة ؟؟

سنتابع درسنا في Recordsetclone استنساخ مجموعة التسجيلات Recordset

وفي هذا الدرس سنتعلم كيف نتعامل مع مجموعة التسجيلات المرتبطة مع نموذج والتي يمكن تحديدها من خاصية record source مصدر السجل في النموذج في وضع التصميم.

هذه المجموعة من نوع **Dynaset** وهو النوع الثاني في تقنية **DAO** والتي يكافئها في **ADO** خاصية **keyset cursor** وهي مجموعة السجلات التي يمكن الأضافة عليها أو التغيير بها أو الحذف منها والمتعلقة بجدول واحد أو مجموعة جداول.

من المهم ملاحظة أن نوع مجموعة التسجيلات التي يتم تعريفها في وضع التصميم في قاعدة البيانات .**mdb** هي بشكل تلقائي **DAO Recordset**، أما في مشروع أكسيس فتكون.**ADO Recordset**

وربما يتساءل البعض لماذا نحتاج إلى استنساخ مجموعة التسجيلات المرتبطة بنموذج ما ولماذا لا نتعامل معها مباشرة ، الأسباب كثيرة ومتعددة ، دعوني أقدم لكم سببا بسيطا مثلا خاصية **Recordcont** ليست متوفرة للنموذج بينما متوفرة لمجموعة التسجيلات ، والخاصية **find**كذلك ، بالإضافة إلى أن مؤشر مجموعة التسجيلات المستنسخة مستقل ولا علاقة له بمؤشر مجموعة التسجيلات المرتبطة بالنموذج وهذا يمكننا من عمل الأشياء بالخفاء دون ظهور تأثيراتها في النموذج وخصوصا في عملية التنقل بين السجلات التي عادة ما نحتاج لها في معظم الإجراءات والمتطلبات البرمجية لتنفيذ مهمات محددة كما في مثالنا التالي:

في مثالنا لهذا الدرس قمت بتصميم زر أمر الذي بضغطه نقوم بقراءة رقم موظف ونطبع كنتيجة أولا عدد السجلات في جدول الموظفين وكذلك عدد الموظفين الذين تحت أمره (يعني عدد الموظفين الذين هو مسؤول عنهم)

CODE

```
Private Sub example١_Click()

Dim rstEmployees As DAO.Recordset

Set rstEmployees = Me.Form.RecordsetClone

id = InputBox ( " ادخل رقم المسئول " )

C = ٠

MsgBox "عدد الموظفين" &  rstEmployees.RecordCount

rstEmployees.MoveFirst

Do While Not rstEmployees.EOF
If rstEmployees!EmpHead = id Then C = C + ١
rstEmployees.MoveNext
Loop

If C = ٠ Then
MsgBox " الموظف صاحب الرقم " & id & " ليس مسئولاً عن احد "

Else
MsgBox " عدد الموظفين الذين بأمره هذا المسئول " & C
End If

rstEmployees.Close

End Sub
```

ملاحظات:

١. **Set rstEmployees = Me.Form.RecordsetClone**

في هذا الأمر تم استنساخ مجموعة التسجيلات المرتبطة بالنموذج وهي في مثالنا سجلات الموظفين من جدول الموظفين وتم تعيينها لمجموعة التسجيلات rstrecordset التي ليس لها علاقة نهائيا في المجموعة المرتبطة بالنموذج.

٢. تم عمل حلقة دورانية تنتهي بالوصول إلى أخر سجل في المجموعة المستنسخة وخلالها نقوم بزيادة العداد بناء على تحقق الشرط الذي بواسطته نحدد أن الموظف الذي أدخلنا رقمه بالمتغير id هو مسؤولا أم لا عن الموظف الذي وصلنا عنده في دورة الحلقة.
٣. وفي النهاية يتم بناء على قيمة c وهي عدد الموظفين الذين تم احصائهم طباعة النتيجة.

درسنا اليوم حول نفس الموضوع السابق وهو استنساخ مجموعة التسجيلات بواسطة Recordsetcloneولكن هذه المرة مع ADO .

قد يظن البعض أن الأمر في غاية البساطة وذلك باستخدم نفس المثال السابق مع تغيير السطر التعريف الأساسي

Dim rstEmployees As DAO.Recordset

إلى

Dim rstEmployees ADODB.Recordset

حسناً هكذا يبدو الأمر ، جربوه ولن ينجح حيث ستحصلون في الجملة

Set rstEmployees = Me.Form.RecordsetClone

على الخطأ Type mismatch أي أن الأنواع غير متوافقة يعني المجموعة المستنسخة من نوع أخر وهذا صحيح ! ، أنسيتم السطرين السابقين باللون الأحمر ، فأنا لم ألونهما جزافاً واعيدهما هنا مرة أخرى للتذكير

من المهم ملاحظة أن نوع مجموعة التسجيلات التي يتم تعريفها في وضع التصميم في قاعدة البيانات .mdb هي بشكل تلقائي DAO Recordset، أما في مشروع أكسيس فتكونADO Recordset

إذاً هذا هو السبب أن مصدر السجلات الذي عرفناه في النموذج والذي سيمثل مجموعة التسجيلات سيكون من النوع . **DAO** حسناً ، الحل بسيط نرجع للمراجع ونجعل مرجع **ADO** يسبق مرجع **DAO** بالتعريف وهكذا يتم تحديد نوع مجموعة التسجيلات من نوع **ADO** ، جربوه وللأسف لن ينجح ، و ياليت الحل كان بهذه البساطة!

ماذا نجري الأن ؟ لم يبقى لنا إلا أن نفعل شيء واحد ونأمل أن ينجح وهو حذف المرجع **DAO** (عدم اختياره من ضمن المراجع) ، جربوا ذلك ، ويا للعجب ، حتى أننا بعد حذف المرجع كاملا لم يتم تعريف مجموعة التسجلات المرتبطة بالنموذج من نوع . **ADO**

ماذا سنعمل إذن ، أولا سنحفظ القاعدة التي في اللون الأحمر حتى لا نتعب أنفسنا أكثر حيث لا يوجد طريقة نهائيا في وضع التصميم لجعل النموذج يرتبط بمجموعة تسجيلات غير **DAO .**

ثأنيا ، لا تبتأسوا كثيرا حيث يوجد حل وهو رائع جدا وهو كما يلي وأيضاً يمكن اعتباره قاعدة ويجب حفظه غيباً.

إذا أردنا أن نجعل مجموعة التسجيلات لنموذج في قاعدة .**mdb** تعمل بتقنية **ADO** يجب عمل الخطوات الثلاث التالية.

١. في وضع التصميم للنموذج نلغي مصدر السجل **Record source** للنموذج (أي نجعله بدون مصدر).
٢. في حدث الفتح للنموذج نضع الكود التالي بدون تغيير سوى بمصدر السجل الذي هو في مثالنا جدول **employees :**

CODE

```
Private Sub Form_Open(Cancel As Integer)
Dim rst As ADODB.Recordset, cn As ADODB.Connection
Set rst = New ADODB.Recordset
rst.ActiveConnection = CurrentProject.Connection
rst.CursorType = adOpenDynamic
rst.CursorLocation = adUseClient
rst.Open "employees", , adOpenKeyset, adLockOptimistic, adCmdTable
Set Me.Recordset = rst
End Sub
```

٣. في حدث الإغلاق للنموذج نضع الكود التالي:

CODE

```
Private Sub Form_Close()
Dim cn As ADODB.Connection
Set cn = Me.Recordset.ActiveConnection
cn.Close
Set cn = Nothing
End Sub
```

وبعدها سيختفي الخطأ Type mismatch الذي تحدثنا عنه في بداية الدرس.

بينما أنتم في انتظار درس مجموعات التسجلات التالي ، ارجو منكم مطالعة هذا الدرس الخفيف والمهم

أولاـ يوجد ٣ أنواع من القيم يجب أن نميز بينها:

null : تشير أن المتغير أو الحقل يحتوي على بيانات غير صحيحة.

empty: تشير أن المتغير لم يتم إعطاءه قيمة إبتدائية حتى ولو تم تعريفه

"": تعني أن المتغير له قيمة ولكنها النص الفارغ.

يعني لكل واحدة معنىً مختلف تماما عن الأخرى ، وحتى نفهم الفرق جيدا للنظر إلى المثال التالي:

CODE

```
Private Sub example_Click()
١)   Dim x As Variant
٢)   If IsNull(Null) Then MsgBox "Null" Else MsgBox "not null"
٣)   If IsEmpty(Empty) Then MsgBox "empty" Else MsgBox "not empty"
٤)   If IsEmpty("") Then MsgBox "empty" Else MsgBox "not empty"
٥)   If IsNull("") Then MsgBox "Null" Else MsgBox "not null"
٦)   If IsNull(x) Then MsgBox "Null" Else MsgBox "not null"
٧)   If IsEmpty(x) Then MsgBox "empty" Else MsgBox "not empty"
٨)   x = ""
٩)   If IsNull(x) Then MsgBox "Null" Else MsgBox "not null"
١٠)   If IsEmpty(x) Then MsgBox "empty" Else MsgBox "not empty"
١١)   x = Null
١٢)   If IsNull(x) Then MsgBox "Null" Else MsgBox "not null"
١٣)   x = Empty
١٤)    If IsNull(x) Then MsgBox "empty" Else MsgBox "not empty"
١٥)    x = Null
١٦)    If IsEmpty(x) Then MsgBox "empty" Else MsgBox "not empty"
End Sub
```

في السطر الأول تم تعريف المتغير x
في السطر الثاني ستكون النتيجة **Null** وهذا يعني أنه يوجد قيمة معروفة في أكسيس تسمى**Null**
في السطر الثالث ستكون النتيجة **Empty** وهذا يعني أن هناك قيمة معروفة في أكسيس تسمى**Empty**
في السطر الرابع ستكون النتيجة **not empty** لأن **""** هو ثابت مع أنه النص الفارغ إلا أنه يعتبر **empty** حيث أن **empty** كما ذكرت تعني عدم تعريف المتغير وليس الفراغ.
في السطر الخامس النتيجة **not null** لأن القيمة **""** لها معناً وهو النص الفارغ.
في السطر السادس النتيجة **not null** ، صحيح أنه تم تعريف المتغير x ولم يتم تحديد قيمة ابتدائية له وهذا لا يجعل قيمته **Null** لأنه لا يحتوي على بيانات سواءً صحيحى أو غير صحيحة.
في السطر السابع النتيجة **empty** حيث أن المتغير x منذ تعريفه لم يتم اسناد قيمة له حتى الآن.
في السطر الثامن في هذه الجملة تم اسناد قيمة النص الفارغ "" للمتغير x
في السطر التاسع وبكل تأكيد هنا النتيجة ليست **null** حيث أنه يوجد قيمة للمتغير x وهي النص الفارغ""
في السطر العاشر وهنا أيضا النتيجة ليست **empty** حيث يوجد قيمة تم اسنادها للمتغير x وبغض النظر عن ما هي.
في السطر الحادي عشر ، شيء رائع الآن أصبحت قيمة x = **null**
في السطر الثاني عشر وهنا بالتاكيد ستكون النتيجة**null**
في السطر الثالث عشر واللآن لاحظوا بدقة ماذا سنفعل جعلنا قيمة x = **empty**
في السطر الرابع عشر ، هنا ربما تستغربون من النتيجة فالبرغم من قيمة x هي **empty** من الجملة السابقة إلا أن النتيجة ! **not empty** أتعرفون لماذا ؟ لأن **Isempty** يسأل فيما إذا كان المتغير اسند له قيمة أم لا بغض النظر عن القيمة حتى ولو كانت **empty** .
في السطر الخامس عشر وأيضا مرة أخرى نجعل قيمة x= **null** وذلك لأتبت لكم القاعدة السابقة
في السطر السادس عشر ، لاحظو وهنا أيضا قيمة x أيضا **not empty** لأنه تم اسناد قيمة للمتغير x وهي **null** .

كيف نحفظ هذه القواعد بعد أن فهمنها وجربنا نتائجها:

لا تدعو ترجمة المعنى الحرفية تخدعكم حيث أن
null، **empty**، **""** إذا ترجمت ستعطي معناً واحداً وهو (لا شيء أو فراغ) هذا الكلام خطأ وهو الخطأ الذي يقع به الكثيرين.

تذكروا دائما التعريفات التي بدأت فيها الدرس ولن تخطئوا باستخدام هذه القواعد أبداً.

الأن ننتقل لدرس تحديث السجلات بتقنيةDAO

في مثالنا التالي سنقوم بوضع ملاحظة لجميع الموظفين الذين يعملون في قسم الإدارة التي رقمها ١٠ والملاحظة هي أن الموظف مسافر في مهمة.

CODE

```
Private Sub Command٠_Click()
Dim db As Database
Dim rst As DAO.Recordset

Set db = CurrentDb
Set rst = db.OpenRecordset("SELECT EmpNotes FROM Employees WHERE DepId='١٠'; ", _
dbOpenDynaset)

 MsgBox rst.RecordCount & " هذا العدد لا يشير الى عدد السجلات "

rst.MoveLast
rst.MoveFirst
MsgBox rst.RecordCount & " عدد السجلات في المجموعة المختارة "
Do While Not rst.EOF
rst.Edit
rst!EmpNotes = " مسافر في مهمة "
rst.Update
rst.MoveNext
Loop
rst.Close
End Sub
```

ملاحظات :

١. في جملة الفتح نلاحظ أننا نستطيع استعمال جملة sql مباشرة وفي هذه الحالة نوع مجموعة التسجيلات يكون**Dynaset**.

٢. نلاحظ أن العدد الذي سنحصل عليه من **recordcount** قبل عملية الذهاب إلى أخر سجل **movelast** وثم الرجوع إلى أول سجل هو ليس عدد السجلات الصحيح ، وحتى نحصل على عدد السجلات الصحيح يجب إتباع هذا الأسلوب دائما وبعدها فقط نستطيع أن نعرف عدد السجلات الناتجة من التعريف.

٣. قبل تعديل أي حقل في مجموعة التسجيلات يجب جعل السجل في وضع التعديل edit وبعد التعديل يجب عمل **Update** (تحديث) وإلا لن يتغير شيء.

٤. يمكن كتابة الأوامر بدون ذكر اسم مجموعة التسجيلات كل مرة باستخدام **with** كما يلي:

CODE

```
Private Sub Command٠_Click()
Dim db As Database
Dim rst As DAO.Recordset

Set db = CurrentDb
Set rst = db.OpenRecordset("SELECT EmpNotes FROM Employees WHERE DepId='١٠'; ",_
OpenDynaset)
With rst
MsgBox .RecordCount & " هذا العدد لا يشير الى عدد السجلات "

.MoveLast
.MoveFirst

MsgBox .RecordCount & " عدد السجلات في المجموعة المختارة "

Do While Not .EOF
.Edit
!EmpNotes = " مسافر في مهمة "

.Update
.MoveNext
Loop
.Close
end with
End Sub
```

يعني عمل **with** هو تمكيننا من الرجوع للخصائص العائدة لمجموعة التسجيلات بدون كتابة اسم مجموعة التسجيلات **rst** في كل مرة.

سيلحق هذا الدرس (بإذن الله) نفس المثال ولكن بإستخدام تقنية ADO.

ونكمل درسنا مع نفس المثال السابق ولكن بتقنية ADO

المثال يصبح كما يلي:

CODE

```
Private Sub Command٠_Click()
Dim Cnn As ADODB.Connection
Dim rst As ADODB.Recordset
Dim strCnn As String

Set Cnn = New ADODB.Connection
strCnn = CurrentProject.Connection
Cnn.Open strCnn

Set rst = New ADODB.Recordset
rst.Open "SELECT EmpNotes FROM Employees WHERE DepId='١٠'; ", Cnn, _
adOpenKeyset, dLockOptimistic


 MsgBox rst.RecordCount & " عدد السجلات  هنا صحيح وهو"

rst.MoveLast
rst.MoveFirst
  MsgBox rst.RecordCount & "عدد السجلات في المجموعة المختارة"

Do While Not rst.EOF
rst.Edit
 rst!EmpNotes ="مسافر في مهمة"
rst.Update
rst.MoveNext
Loop
rst.Close
End Sub
```

الملاحظات:

١. كما ذكرنا سابقا **Dynaset** في **DAO** يقابلها **Keyset** في تقنية **ADO** هذا بالنسبة للتعريفات.

٢. .مشكلة أسلوب الحصول على عدد السجلات في المجموعة التي اضطررنا إلى الذهاب فيها إلى أخر سجل ثم إلى أول سجل ، قد اختفت هنا ولا حاجة سوى للذهاب إلى أول سجل قبل الدخول في الحلقة ولكنني أبقيتها هنا حتى تتأكدوا من ذلك بأنفسكم.
٣. بالنسبة لإستخدام **with** أو عدم استخدامها لا علاقة لها بتقنية **ADO** أو **DAO** ويبقى الهدف من استخدمها ان استخدمت كما ذكرت سابقا هو تسهيل عملية الوصول إلى الخصائص مباشرة بكتابتها بعد النقطة وبدون كتابة اسم مجموعة التسجيلات كما أوضحت ذلك في الدرس السابق.
٤. لا يوجد هنا خاصية .**Edit** ولا داعي لإستخدامها كما في **DAO** أما بالنسبة لخاصية التحديث .**Update** فسواءً تم وضعها أو لا النتيجة واحدة.

اخوكم / خضر الرجبي .. منتديات اوفيسنا