الجزء الرابع

# برمجة قواعد البيانات

PDF Factory Pro trial version www.fastio.com

# استخدام ADO.NET

عند الحديث عن قواعد البيانات، يتعالى صدى الحروف ADO.NET بين أحاديث المبـــرمجين، فهي الوسيلة المثلى لربط شيفراتهم البرمجية بملفات قواعد البيانات سواء كانت محلية أو على بعد آلاف الكيلومترات.

هذا الفصل هو مدخلك المبدئي إلى ADO.NET واستخدام فئاتها، ودعني انوه هنا بان هذا الكتاب مختص **ببرمجة** قواعد البيانات وليس **التعامل** مع قواعد البيانات بشكل عام، فلن اتحدث عن طريقة بناء الجداول، اعداد مخططات Entity Relation Diagrams (ERD)، أو ادارة نظـــم قواعد البيانات، بل سيكون حديثي محصور حول طرق الوصول إلى قواعد البيانات من شيفراتك المصدرية والمكتوبة بـــ Visual Basic .NET.

**ملاحظة**

الغالبية الساحقة من فئات هذا الفصل والفصول التي تليه مشمولة في مجالات الاسماء التالية:

```
Imports System.Data
Imports System.Data.OleDb
Imports System.Data.SqlClient
```

فلا تنسى استيرادها في اعلى ملفات شيفراتك البرمجية، أو من صندوق حوار خصائص المشروع Project Property Pages لاختصار كتابة الشيفرات البرمجية.

# مدخلك إلى ADO.NET

**ADO.NET** هي مجموعة من الفئات مشمولة في مجال الاسماء System.Data غرضها الوصول إلى **مصادر البيانات Data Sources**، والتي تمثل بيانات محفوظة تحت أنظمة قواعد بيانات متعددة الانواع، الاجناس، والاعراق مما يعني قدرتك على الوصول إلى أي قاعدة بيانات مهما كانت الشركة المنتجة لها.

## الوضع المتصل والوضع المنفصل

مهما كان نوع مصدر البيانات الذي تتعامل معه، عند استخدام ADO.NET عليك اختيار وضع من وضعين هما: **الوضع المتصل Connected Mode**، **والوضع المنفصل Disconnected Mode**.

في الوضع المتصل يتم الاتصال مع مصدر البيانات وتجري كافة العمليات –افضل ما أوصفها-على الهواء المباشرة، أو كما انك تجري محادثة هاتفية مع احد الاشخاص، فالاتصال مستمر وحي يرزق بين شيفراتك المصدرية وبين مصدر البيانات الأساسي، وأي خلل في مصدر البيانات أو ايقاف مؤقت، سيؤدي إلى خلل في شيفراتك المصدرية.

اما في الوضع المنفصل فهو قريب من فكرة صندوق الوارد Inbox، حيث تأتيك البيانات وتعدل فيها كما تشاء ومن ثم ترسل كافة التعديلات. وهذا ما يحدث مع ADO.NET بالضبط، فبعد قيامك بإجراء اتصال مع مصدر بيانات، ستحصل على البيانات المطلوبة وتنتهي علاقتك بمصدر البيانات الاساسي، وتتعامل مع البيانات –التي عندك- كما تتعامل معه الوضع المتصل. ليس هذا فقط، بل يمكنك تعديلها وتحريرها واجراء كافة جمل الاستعلام SQL عليها، أو –الاكثر من ذلك- تعيد هيكلة البيانات Data Structure كتغيير في علاقات جداولها Table Relationships أو حقولها.

ميزة عظيمة في الوضع المنفصل يظهر من خلال استقلالية البيانات عن هيئة مصدر البيانات، ماذا يعني هذا؟ يعني انك تستطيع الحصول على بيانات من مصدر بيانات من النوع Microsoft SQL Server®، و تتعامل مع البيانات وتحريرها، ومن ثم ارسال البيانات إلى مصدر بيانات من النوع Microsoft Access®.

كان الغرض الاساسي من تطوير فكرة الوضع المنفصل هو تخفيف الضغط على مصادر البيانات في اجهزة الخادم Servers والتي تستخدم من قبل عدد كبير من العملاء Clients،

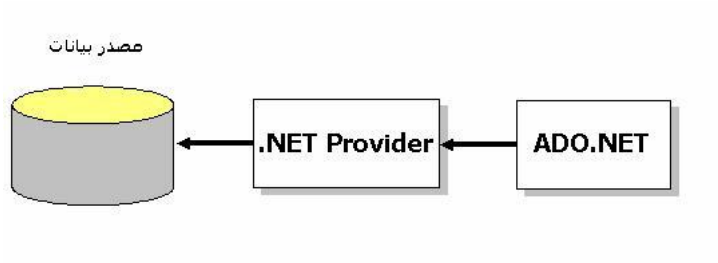
فيمكنك الحصول على البيانات قبيل ساعات الدوام، ومن ثم ارسالها وقت الغداء وهو انسب وقت يخف الضغط على الخادم فيه.

في هذا الفصل سيتمحور حديثي عن الوضع المتصل، حيث يعتبر مفتاحك لاستخدام والتعامل مع مصادر البيانات في الوضع المنفصل والذي سأتطرق اليه في الفصل القادم **ADO.NET للاتصال المنفصل**.

## مزودات .NET Data Providers

الميزة العظيمة التي تجنيها من استخدام ADO.NET هو استقلالية شيفراتك البرمجية عن الهيئات المختلفة لمصادر البيانات، أي ان نفس الشيفرات المصدرية التي استخدمتها لتطوير تطبيقات معتمدة على مصادر بيانات من نوع Microsoft Access®، ستستخدمها ايضا مع انواع مصادر بيانات اخرى كـ Oracle®، Microsoft SQL Server®، Microsoft FoxPro® ...الخ، دون الحاجة لتغيير شيفراتك المصدرية.

السؤال الذي يطرح نفسه، كيف يمكن لـ ADO.NET من فعل ذلك رغم انها كتبت مرة واحدة فقط، وبالتالي ستكون موجه إلى نوع معين من مصادر البيانات؟ والجواب هو ان ADO.NET في الحقيقة لا تصل مباشرة إلى مصادر البيانات وانما تعتبر واجهة للمبرمج فقط، حيث انها تستخدم **مزودات .NET Data Providers (شكل 17-1)**.




**شكل 17-1**: مزودات .NET Providers. تعمل كحلقة وصل بين ADO.NET ومصادر البيانات

مزودات .NET Data Providers. ما هي الا حلقة وصل بين ADO.NET ومصادر البيانات، فعندما تريد من ADO.NET استخدام مصدر بيانات من النوع Microsoft

Access®، عليك توفير مزود NET Data Provider. والخاص بـــ Microsoft Access®، اما ان كبر حجم قاعدة البيانات واردت الاعتماد على خادم كـــــ Microsoft SQL Server®، فكل ما هو مطلوب منك استخدام المزود الخاص بـــ Microsoft SQL Server®.

من منظورك كمبرمج، ستستخدم ADO.NET بغض النظر عن نوع مصدر البيانات، ومن منظورك كمصمم للتطبيق، عليك تحديد مزود NET Data Providers. والذي يلائم نوع مصدر البيانات الذي تستخدمه، كما عليك إرفاق كافة ملفات هذا المزود عند توزيع برنامجك.

توجد العشرات من مزودات NET Data Providers. تمثل انواع مختلفة مـــن مـــصادر البيانات، ويتم تحديث اصدارات كل فترات متعددة، عليك البحث دائما عن اخر اخبار المزود الذي تستخدمه في موقع الشركة المنتجة له.

توجد ثلاثة انواع من مزودات البيانات مدعومة في اطار عمل NET Framework. هي:

**OLE DB .NET Data Provider**:

هذا النوع من المزودات يمكنك من استخدام مزودات OLE DB قديمة مبنية على تقنية COM. يفيدك كثيرا ان اردت استخدام نفس مصادر البيانات والمنجزة قبل تقنية NET. Microsoft، مع العلم ان استخدامها اقل كفاءة من استخدام مزودات NET. وذلك لإجراء عمليات تحويل الشيفرات من NET. إلى COM ومن COM إلى NET. (لا تقلق، فلست مسئولا عن هذه العمليـــات ان استخدمت مزودات خاصة بمصادر بيانات لـــ Microsoft Access® مثلا).

**SQL Server .NET Data Provider**:

هذا المزود موجه -بشكل خاص- إلى مصادر البيانات المنجزة بالاصدار السابع من Microsoft SQL Server® وما بعده، وان اردت التعامل مع اصدارات اقدم، فعليك استخدام المزودات من النوع OLE DB .NET Data Provider السابقة.

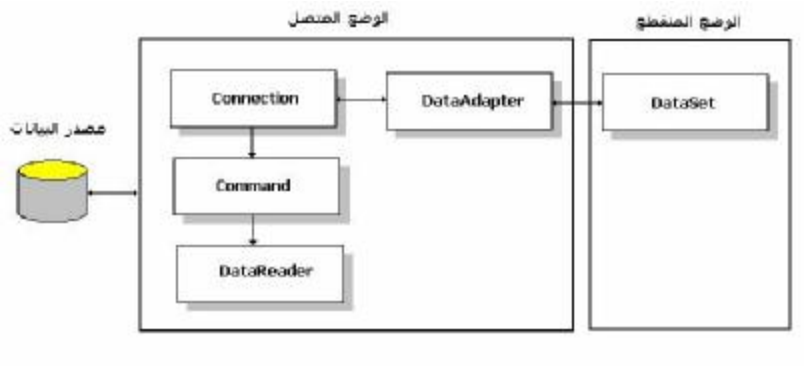
**ODBC .NET Data Provider**:

قبل تقنية NET. وقبل تقنيات الوصول إلى البيانات، كانـــت تقنيـــة ODBC اول تقنيـــة تمكـــن المبرمجين من الوصول إلى مصادر البيانات المختلفة عن طريق **مـــشغلات ODBC Drivers** المختلفة (الفكرة من المشغلات والمزودات شبيهه إلى حد كبير من ناحية نظرية). الغـــرض مـــن المزود ODBC .NET Data Provides هو استخدام مشغلات ODBC Drivers للوصول إلى

مصادر البيانات (وهي اقل كفاءة حتى من مــزودات OLE DB .NET Data Providers ولا انصحك بالاعتماد عليها).

## فئات ADO.NET

تحتوي ADO.NET على مجموعة كبيــرة من الفئات، ولكن اغلبها مشتقة مــن خمــس فئــات رئيــسية هــي: Connection، Command، DataReader، DataAdapter، و DataSet (**شكل 17-2**)، وعملية استيعاب هذه الفئات القاعدية سيسهل عليك الامر كثيرا للنزول إلى الفئات المشتقة وفهم طرق استخدامها والربط بين الكائنات المنشئة منها.




**شكل 17-2**: ابرز فئات ADO.NET.

سأفصل في جميع الفئات -السابق ذكرها- في هذا الفصل والفصل القادم بمشيئة الله، ولكن دعنا الان ناخذ جولة سريعة حولها. حيث تكون البداية دائما وأبدا مع Connection، اذ يترتــب عليك انشاء كائن من هذه الفئة عن طريقه تتصل بمصدر البيانات، كما ستلحق كــل خــصائص ومواصفات الاتصال (كاسم المستخدم، كلمة المرور، مزود NET Data Provider. المــستخدم ...الخ) مع هذا الكائن.

بعد انشائك لكائن اتصال، سترسل مرجع لهذا الكائن إلى طرق وخصائص كائن اخر مــن النوع Command، يمكنك من تنفيذ جمل الاستعلام المختلفــة (كـــ SELECT، DELETE، INSERT وغيرها) على مصدر البيانات والمحدد في الكائن Connection السابق.

بعد تنفيذ جمل الاستعلام في الكائن Command، ستكون النتائج متمثلة في سجلات Records تصل اليها بين ثنايا الكائن من النوع DataReader، لتتمكن من قراءة كل سجل أو كل حقل على حدة، مع العلم انك لن تستطيع تحديث مصدر البيانات من خلال هذا الكائن.

الفئات الثلاث السابق ذكرها تستخدم في حالة الوضع المتصل Connected Mode، اما الكائنات من النوع DataSet فيوجد الكثير لأخبرك به حولها في الفصل القادم. وحتى ان نلتقي هناك، اعلم انها تمثل البيانات المأخوذة من مصدر البيانات.

أخيرا، يمكنك الحصول على البيانات من مصدر البيانات أو ارسالها اليه، وإلحاقها إلى الكائنات من النوع DataSet عن طريق كائن من النوع DataAdapter، يمكنك اعتباره كهمزة الوصل التي تمكنك من ارسال/استقبال البيانات من والى كائنات Connection.

في هذا الفصل سافصل في الفئات Connection، Command، و DataReader. اما البقية فستأتي لاحقا، وسأبدأ الان بكائن الاتصال Connection.

# كائن الاتصال Connection

في عالم ADO.NET، الاتصال الذي تجريه مع مصدر بيانات هو كائن من النوع Connection. وقبل التعامل مع مصدر بيانات، عليك فتح اتصال معها (حتى ولو كنت في الوضع المنفصل Disconnected Mode). في هذا القسم من الفصل سنعرض الاساليب المتعددة للاتصال بمصادر البيانات.

## إنشاء كائن اتصال

عندما تنوي انشاء كائن اتصال Connection فانك لن تقوم بتعريفه من الفئة Connection السابقة، وانما ستستخدم نوعين من الاتصال، النوع الاول للمزودات من النوع OLE DB .NET Data Provider حيث ستعرف كائن اتصال من الفئة OleDbConnection:

```
Dim msAccessCn As New OleDbConnection()
```

اما ان كان مصدر البيانات الذي تنوي الاتصال به يتبع للمزود SQL Server .NET Data Provider، فالفئة SQLConnection هي المسئولة عن انشاء كائنات من هذا النوع:

```
Dim msSQLCn As New SqlConnection()
```

PDF

كلا الفئتين تحتويان على الواجهة IDbConnection مما يمكنك من الاستفادة من مبدأ تعدد الواجهات Polymorphism وتتمكن –مثلا- من كتابة اجراء واحد يستقبل كلا النوعين لاختصار كتابة الشيفرات المكررة:

```
Sub ConnectionDB(ByVal dbCon As IDbConnection)
    …
    …
    …
End Sub
```

## نص الاتصال

بعد انشاء كائن الاتصال، عليك ارفاق **نص الاتصال Connection String** اليه، ونص الاتصال ما هو إلى قيمة حرفية (من النوع String) تحتوي على كل شيء تتعلق بعملية الاتصال بمصدر البيانات، كاسم مصدر البيانات، مسار قاعدة ملف قاعدة البيانات (أو قد تكون اسم الجهاز في الشبكة ان كنت معتمد على خادم قواعد بيانات كـ Microsoft SQL Server®)، اسم المستخدم، كلمة المرور ...الخ، نص الاتصال التالي مناسب جدا لمصادر بيانات من النوع Microsoft Access®:

```
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source=C:\Folder\Data.MDB;"
```

حيث Provider هو الاسم الكامل للمزود، و Data Source اسم ملف قاعدة البيانات. المزيد ايضا، يمكنك اسناد **وقت الانتظار Connection Timeout** بين ثنايا نص الاتصال، ووقت الانتظار هي اطول فترة التي سيضل الكائن منتظرا ردة الفعل من مصدر البيانات عند الاتصال، فنص الاتصال التالي سينتظر 10 ثواني لفتح الاتصال مع مصدر البيانات، وان انتهت الفترة ولم تحدث أي ردة فعل من مصدر البيانات نفسه، سيظهر خطأ (القيمة الافتراضية هي 15 ثانية):

```
Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;" _
    & "Data Source=C:\Folder\Data.MDB; Connection Timeout=10"
```

بعد كتابتك لنص اتصال، أسنده فورا إلى الخاصية ConnectionString والتابعة لكائن الاتصال، كما يمكنك ارسال نص الاتصال مباشرة إلى مشيد الفئة لحظة انشاء الكائن:

```
' اسناد قيمة للخاصية
Dim msAccessCn As New OleDbConnection()

msAccessCn.ConnectionString = connString
```

PDF Factory Pro trial version www.fastio.com

```
' ارسالها مع المشيد
Dim msAccessCn As New OleDbConnection(connString)
```

**ملاحظة**

بدلا من كتابة نص الاتصال كاملا في كل مرة تنوي انشاء كائن اتصال جديد، يفضل وضعه في متغير أو ثابت عام على مستوى المشروع، اما أفضل نصيحة اسطرها لك فهي حفظ نصوص الاتصال في ملف الاعدادات project.exe.config أو أي ملف خارجي اخر، وذلك لتمكن المستخدمين من تغيير نص الاتصال في حال ما ان تم تغيير موقع مصدر البيانات.

يمكنك الاستعلام عن معظم القيم التي اسندتها إلى نص الاتصال عن طريق مجموعة من الخصائص للقراءة فقط ReadOnly، كالخاصية:

```
Dim msAccessCn As New OleDbConnection(connString)

MsgBox(msAccessCn.Provider)           ' Microsoft.Jet.OLEDB.4.0
MsgBox(msAccessCn.DataSource) ' C:\Folder\Data.MDB
MsgBox(msAccessCn.ConnectionTimeout)  ' 10
```

عندما تتعامل مع الكائنات من النوع SqlConnection، فعليك تجاهل اسم المزود عند كتابتك لنص الاتصال، والسبب قد يبدوا بديهيا ان علمت ان الكائنات من النوع SqlConnection لا تقبل الا المزودات من النوع SQL Server .NET Data Provider:

```
Dim msSQLCn As New SqlConnection()

msSQLCn.ConnectionString = "Data Source=DEV4ARABS_SERVER;" _
    & "User ID=تركي العسيري; Password=ولا حتى تحلم فيها;" _
    & "Initial Catalog=قاعدة بيانات المقالات"
```

## فتح وإغلاق الاتصالات

بعد اسنادك لنص الاتصال المناسب للخاصية ConnectionString، يمكنك البدء بفتح الاتصال مع مصدر البيانات باستدعاء الطريقة ()Open:

```
Dim msAccessCn As New OleDbConnection(connString)
Dim msSQLCn As New SqlConnection(connString2)

msAccessCn.Open ()
msSQLCn.Open ()
```

**ملاحظة**

المتغير connString السابق اقصد فيه نص اتصال Connection String، لذلك في كل مرة تجده في هذا الفصل والفصول اللاحقة لا تستغرب من وجوده.

ومن الضروري جدا اغلاق الاتصال عند عدم الحاجة اليه باستدعاء الطريقة Close():

```
msAccessCn.Close ()
msSQLCn.Close ()
```

تستطيع معرفة حالة الاتصال عن طريق الخاصية State التابعة لكائن الاتصال، والتي قد تكون قيمة أو اكثر من القيم: Closed الاتصال مغلق، Open الاتصال مفتوح، Connecting جاري فتح الاتصال، Executing يتم تنفيذ امر استعلام على الاتصال، و Fetching جاري الحصول على بيانات من سجلات مصدر البيانات:

```
Dim cn As New OleDbConnection()
…
…
…

If (cn.State And ConnectionState.Open) <> 0 Then
    cn.Close()
End If
```

المزيد ايضا، عند تغيير حالة الاتصال من Open إلى Closed (أو العكس) سيتم تفجير الحدث StateChange والخاص بكائن الاتصال:

```
Dim WithEvents cn As New OleDbConnection()

Sub cn_StateChange(ByVal sender As Object, _
    ByVal e As System.Data.StateChangeEventArgs) Handles _
    cn.StateChange
```

PDF

```
    If (e.CurrentState And ConnectionState.Open) <> 0 Then
        Label1.Text = "تم فتح الاتصال"
    Elseif e.CurrentState = ConnectionState.Closed Then
        Label1.Text = "تم اغلاق الاتصال"
    End If
End Sub
```

**تفادي الاستثناءات:**

عند التعامل مع مصادر البيانات، فان نسبة وقوع الاستثناءات كبيرة جدا لأي سبب أو خلل فنـــي، لذلك ينصح بشدة من تفادي الاستثناءات ووضع الشيفرات الخاصة بمصادر البيانات داخل التركيب Try … End Try:

```
Dim cn As New OleDbConnection()

Try
    cn.Open()
    …
    …
    …
Catch ex As Exception
    MsgBox(ex.Message)
End Try
```

بالنسبة للطريقة Close() السابقة، عليك اســتدعائها دائمـــا لتغلــق الاتــصال، حيــث ان الاتصالات لا يتم اغلاقها الا لحظة **الموت الحقيقي** للكائنات (عندما تبدأ المجموعـــة Garbage Collection عملها)، وبما اننا لا نعلم متى سيحدث هذا فعلينا استدعائها دائما، سواء وقع استثناء أو لم يقع:

```
Dim cn As New OleDbConnection()

Try
    cn.Open()
    …
    …
    …
Catch ex As Exception
    MsgBox(ex.Message)
Finally
    cn.Close()
End Try
```

PDF Factory Pro trial version www.fastio.com

كما يفضل الاعتماد على كائنات الاستثناءات OleDbException أو SqlException للتفريق بين الاستثناءات الخاصة بكائنات الاتصالات والاستثناءات الاخرى:

```
Try
    …
    …
Catch ex As OleDbException
    …
    …
Catch ex As Exception
    …
    …
End Try
```

**انظر ايضا**

لمزيد من التفاصيل حول الاستثناءات Exceptions وطرق تفاديها، راجع الفصل السابع **اكتشاف الأخطاء**.

# كائن الأوامر Command

بعد تكوين الاتصال مع قاعدة البيانات، تأتي الخطوة التالية وهي ارسال جمل الاستعلام SQL إلى قاعدة البيانات لتعديل محتوياتها، في هذا القسم سنرى كيف يمكنك الاستفادة من كائن الاتصال Connection وتعديل بيانات قاعدة البيانات عن طريق كائن الاوامر Command.

**انظر ايضا**

سأستخدم في هذا القسم –والقسم الذي يليه- مجموعة من جمل الاستعلام SQL، وسأفترض بان لديك خلفية في لغة الاستعلام SQL أو انك قد قرأت المحلق ب **لغة الاستعلام** SQL والملقى في نهاية هذا الكتاب.

PDF

## إنشاء كائن أوامر

عندما تنوي انشاء كائن اوامر Command فانك لن تقوم بتعريفه من الفئة Command، وانما ستستخدم نوع يماثل مزود كائن الاتصال Connection، فان كان مزود كائن الاتصال من النوع OLE DB .NET Data Provider، ستعرف كائن اوامر من الفئة OleDbCommand:

```
Dim msAccessCmd As New OleDbCommand ()
```

اما ان كان مزود كائن الاتصال الذي تنوي استخدامه يتبع للمزود SQL Server .NET Data Provider، فالفئة SqlCommand هي المسئولة عن انشاء كائنات اوامر خاصة لها:

```
Dim msSQLCmb As New SqlCommand()
```

كلا الفئتين تحتويان على الواجهة IDbCommand مما يمكنك من الاستفادة من مبدأ تعدد الواجهات Polymorphism وتتمكن –مثلا- من كتابة إجراء واحد يستقبل كلا النوعين للتقليص من عدد الشيفرات المكررة:

```
Sub CommandDB(ByVal dbCmd As IDbCommand)
    …
    …
    …
End Sub
```

## الربط مع اتصال

بعد إنشائك لكائن اوامر Command، اول خطوة عليك إنجازها هي ربطه مع كائن اتصال Connection، يمكنك اسناد مرجع إلى كائن اتصال عن طريق الخاصية Connection:

```
Dim cn As New OleDbConnection(connString)
Dim cmd As New OleDbCommand()

cn.Open()

cmd.Connection = cn
```

ولا تنسى ضرورة توافق نوع كائن الاوامر مع نوع كائن الاتصال، فلو كان كائن الاتصال يتبع مزود من النوع SQL .NET Data Provider فقد تم إنشائه من الفئة SqlConnection، لذلك

PDF Factory Pro trial version www.fineprint.com

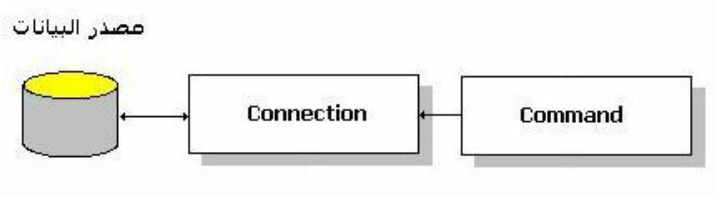
عليك الاعتماد على الفئة SqlCommand عوضا عن OLEDbCommand للربط الصحيح مع الاتصال:

```
Dim cn As New SqlConnection(connString)
Dim cmd As New SqlCommand()

cn.Open()

cmd.Connection = cn
```

ضع في عين الاعتبار، ان كائن الاوامر Command مرتبط ارتباطا وثيقا بكائن الاتصال Connection، فلو تم –مثلا- اغلاق كائن الاتصال بالطريقة ()Close، فلن تتمكن من تنفيذ جمل الاستعلام SQL مع كائن الاوامر:

```
Dim cn As New OleDbConnection(connString)
Dim cmd As New OleDbCommand()

cn.Open()
cmd.Connection = cn
cn.Close()
' لن تتمكن من الاستفادة من كائن الاوامر
cmd.ExecuteNonQuery()
```

سيظهر لك السبب واضحا ان علمت ان كائن الاوامر لا يصل إلى مصدر البيانات بشكل مباشر، وانما يعتمد على كائن اتصال والذي بدوره يصل إلى مصدر بيانات (**شكل 17-3**).




**شكل 17-3**: الكائن Command يعتمد على الكائن Connection للوصول إلى مصدر بيانات.

## تنفيذ جمل الاستعلام SQL

ان اردت تنفيذ جمل استعلام على الكائن Command، فعليك تحديد نوع جملة الاستعلام SQL التي تود تنفيذه، هل هي جملة استعلامية تقليدية أو جملة تنفيذية؟

الجمل الاستعلامية هي تلك الجمل التي لا تؤثر على سجلات قاعدة البيانات وانما تقوم بقراءة محتوياتها، في لغة الاستعلام SQL نستخدم الامر SELECT لهذا النوع من الجمل، وسترى في الفقرة التالية **قراءة السجلات** كيف يمكنك تنفيذها على كائن Command باستدعاء الطرق ExecuteXXX().

اما ان كانت جمل الاستعلام SQL هي جمل تنفيذية، عليك استخدام الطريقة ExecuteNonQuery() والتي تعود بقيمة عددية تمثل عدد السجلات التي تأثرت. الجمل التنفيذية –التي اقصدها في هذا السياق- هي تلك الجمل التي تحدث تغييرا في سجلات جداول قاعدة البيانات وهي اما تكون UPDATE، INSERT INTO، أو DELETE.

حتى تتمكن من استدعاء الطريقة ExecuteNonQuery()، عليك اولا كتابة جملة الاستعلام SQL في الخاصية CommandText والتابعة للكائن Command:

```
Dim cn As New OleDbConnection(connString)
cn.Open()

Dim sqlStatement As String = "UPDATE [جدول الرواتب] SET [الراتب]" _
    & " = 5000 WHERE [رقم الموظف] =  99999"
Dim cmd As New OleDbCommand()


cmd.Connection = cn
cmd.CommandText = sqlStatement
cmd.ExecuteNonQuery()
```

يمكنك دمج السطرين والمتعلقين بتعديل قيم الخاصيتين Connection و CommandText في سطر واحد، وذلك بارسال امر جملة الاستعلام وكائن الاتصال مع مشيد الفئة Command:

```
Dim cmd As New OleDbCommand(sqlStatement, cn)
```

اخيرا، دعني اعيد تذكيرك بضرورة تفادي الاستثناءات لحظة تنفيذ جمل الاستعلام –فقد تتغير صلاحياتك على مصدر البيانات أو يحدث أي خلل فني في عملية التحديث:

PDF Factory Pro trial version www.fineprint.com

```
Try
    cmd.ExecuteNonQuery ()
    …
    …
Catch ex As Exception
    …
    …
End Try
```

## قراءة السجلات

عندما تنوي استخدام الامر SELECT في جمل الاستعلام لقراءة السجلات، فيمكن اختيار طريقة من ثلاث طرق تابعة للكائن Command هـي: ()ExecuteReader، ()ExecuteScalar، و ExecuteXMLReader.

**الطريقة ()ExecuteReader**:

الطريقة ()ExecuteReader تعود بكائن من النوع DataReader تمثل نتيجة جملة الاستعلام في الكائن Command (سأتحدث عن الكائن DataReader لاحقا في القسم **كائن البيانات DataReader** من هذا الفصل):

```
Dim cmd As New OleDbCommand("SELECT * FROM [جدول الحسنوات]", cn)
Dim dr As OleDbDataReader = cmd.ExecuteReader()

Do While dr.Read()
    ListBox1.Items.Add(CInt(dr.Item("الوزن")))
Loop

dr.Close()
```

**الطريقة ()ExecuteScalar**:

تستخدم الطريقة ()ExecuteScalar لقراءة **حقل Field** واحد فقط من حقول السجل، وتعود بقيمة تمثل ذلك الحقل:

```
Dim cmd As New OleDbCommand("SELECT [اسم الضفدع] FROM " _
    & " [الضفادع البشرية] WHERE [المعرف] = 1", cn)

Dim Name As String = cmd.ExecuteScalar()
```

PDF Factory Pro trial version www.fastio.com

تفديك الطريقة ()ExecuteScalar كثيرا ان اردت قراءة قيمة واحدة من من السجل وذلك لزيادة سرعة التنفيذ، ان كانت جملة الاستعلام تعود باكثر من حقل، فالحقل الاول هو الذي سيتم قراءته، وان كانت جملة الاستعلام تعود بأكثر من سجل، فسيتم قراءة الحقل التابع للسجل الاول.

**ملاحظة**

لا اقصد –في فصول الجزء الرابع **برمجة قواعد البيانات** بالتحديد- من كلمة الحقل Field الذي تعرفه في الفئات، وانما اقصد به العامود الموجود في احد جداول قاعدة البيانات.

يمكنك الاستفادة من الطريقة ()ExecuteScalar في معرفة عدد السجلات بشكل سريع – على سبيل المثال لا الحصر:

```
Dim cmd As New OleDbCommand("SELECT COUNT(*) FROM " _
    & " [الضفادع البشرية]", cn)

Dim countOfFrogs As Integer = CInt(cmd.ExecuteScalar())
```

**الطريقة ()ExecuteXMLReader**:

ان استخدم المزود SQL Server .NET Data Provider، فستتمكن من استدعاء الطريقة ()ExecuteXMLReader، حيث ان قواعد البيانات المعتمدة على انظمة Microsoft SQL Server®، يمكن استخدام الامر FOR XML من اوامر لغة الاستعلام SQL معها، والتي تعود بالسجلات بهيئة XML.

اذا كانت الطريقة ()ExecuteReader تعود بكائن من النوع DataReader، فان الطريقة ()ExecuteXMLReader تعود بكائن من النوع System.XML.XmlReader، واستخدامها شبيه –إلى حد كبير- مع DataReader:

```
Dim cmd As New SqlCommand("SELECT * FROM [جدول الذكريات]", cn)
Dim xmlr As System.Xml.XmlReader = cmd.ExecuteXmlReader()

Do While xmlr.Read()
    MsgBox(xmlr.Value)
Loop

xmlr.Close()
```

**انظر ايضا**

سأعود للطريقة ()ExecuteXmlReader وكيفية استخدامها مع الفئة XmlReader في الفصل التاسع عشر **تكامل** ADO.NET **و** XML.

# كائن البيانات DataReader

بعد تنفيذ جملة الاستعلام في الكائن Command باستدعاء الطريقة السابقة ()ExecuteReader، ستعود هذه الطريقة بكائن بيانات من النوع DataReader يمثل جميع السجلات الناتجة من جملة الاستعلام، وقبل ان نرى كيف يمكنك الاستفادة من هذا الكائن، من الجيد معرفة كيف يمكنك الحصول عليه –أي إنشائه.

## إنشاء كائن بيانات

عندما تنوي إنشاء كائن بيانات DataReader فانك لن تستطيع استخدام الامر New، وانما ستضطر إلى استخدام الطريقة ()ExecuteReader والتابعة للكائن Command، فان كان مزود الكائن Command من النوع OLE DB .NET Data Provider، فانك ستعرف كائن بيانات من الفئة OleDbDataReader:

```
Dim cmd As New OleDbCommand("SELECT * FROM [جدول]", cn)
Dim dr As OleDbDataReader = cmd.ExecuteReader()
```

اما ان كان مزود كائن الاوامر Command الذي تنوي استخدامه يتبع للمزود SQL Server .NET Data Provider، فالفئة SqlDataReader هي المسئولة عن إنشاء كائنات بيانات خاصة ملائمة ومتوافقة معها:

```
Dim cmd As New SqlCommand ("SELECT * FROM [جدول]", cn)
Dim dr As SqlDataReader = cmd.ExecuteReader
```

مرة اخرى، الواجهة IDataReader مشمولة في كلا الفئتين OleDbDataReader و SqlDataReader، فيمكنك تعريف اجراء يستقبل في وسيطته كائن من كلا النوعين:

```
Sub ReaderDB(ByVal dr As IDataReader)
    …
    …
    …
End Sub
```

## قراءة السجلات

بعد إنشائك لكائن بيانات DataReader، تستطيع البدء في قراءة سجلاته في خطوتين، الاولى باستدعاء الطريقة Read() لتحميل حقول السجل، والثانية تتم فيها قراءة قيمة الخاصية Item والتي ترسل معها اسم الحقل المراد قراءته:

```
Dim dr As OleDbDataReader = cmd.ExecuteReader()

dr.Read()
MsgBox ( dr.Item("الاسم") & " - " CInt(dr.Item("العمر")) )

dr.Read()
MsgBox ( dr.Item("الاسم") & " - " CInt(dr.Item("العمر")) )

dr.Close()
```

كما يمكنك استخدام رقم فهرس الحقل بدلا من كتابة اسمه:

```
MsgBox ( dr(0) & " - " & CInt(dr(1)) )
```

**ملاحظة**

عند استخدام رقم فهرس الحقل كما في الشيفرة الأخيرة، عليك الانتباه ان الترقيم يعتمد على مكان وجود الحقل في جملة الاستعلام وليس ترتيب الحقل في نفس جدول قاعدة البيانات الأصلي، لذلك قد يختلف رقم فهرس الحقل من جملة استعلام إلى اخرى.

في كل مرة تستدعي فيها الطريقة Read() سيتم نقل المؤشر إلى السجل التالي، وستعود الطريقة بالقيمة False ان وصلت إلى نهاية السجلات، لذلك الاستخدام الامثل لها يكون في حلقة Do … Loop بهذا الشكل:

```
Dim dr As OleDbDataReader = cmd.ExecuteReader()
…
…
Do While dr.Read()
    Msgbox (dr.Item("الاسم") & " - " CInt(dr.Item("العمر")))
Loop

dr.Close()
```

**ملاحظة**

Item هي الخاصية الافتراضية لكائن البيانات DataReader، لذلك يمكنك اختصار استدعائه بهذا الشكل:

MsgBox ( dr("الاسم") & " – " & CInt(dr("العمر")) )

المزيد ايضا، الخاصية Item تعود دائما بقيمة من النوع Object، مما يضطرك إلى الاعتماد على دوال التحويل (CInt()، CLng()، CSng() ...الخ) لقرائة القيمة، مع ذلك يمكنك الاعتماد على الطرق Getxxx() التي تعود بالنوع المكافئ –دون الحاجة لاستخدام دوال التحويل:

```
Dim age As Integer

age = CInt(dr.Item("العمر")
age = dr.GetInt32(1)
```

من المهم التنبيه هنا بضرورة اغلاق كائن DataReader باستدعاء طريقته Close()، السبب في ذلك ليس فقط من اجل تحرير مصادر النظام، وانما يتعدى ذلك بكثير، اذ بمجرد قيامك بإنشاء كائن DataReader، سيتم شلل كافة العمليات الاخرى على كائن الاتصال Connection والاوامر Command، ولن تتمكن من عمل أي شيء الا استدعاء الطريقة Close() لكائن الاتصال في هذه الحالة.

اخيرا، تحتوي الكائنات من النوع DataReader على مجموعة اضافية من الطرق والخصائص المفيدة كالخاصية GetName التي تعود باسم الحقل (وليس قيمته)، الخاصية FieldCounter التي تعود بعدد الحقول، والاهم من ذلك الطريقة IsDBNull التي تعود بالقيمة True ان كان الحقل فارغ Null (عليك التحقق منها دائما قبل قراءة قيمة الحقل حتى تتفادى ظهور الاستثناءات).

PDF Factory Pro trial version www.fastio.com

## خاص بمستخدمي Microsoft SQL Server ®

عند التعامل مع مزودات من النوع SQL Server .NET Data Provider، يفضل دائما قراءة قيمة الحقل باستخدام الطرق GetSqlxxx() عوضا عن الخاصية Item:

```
Dim dr As SqlDataReader = cmd.ExecuteReader()
dr.GetSqlInt32(0)
```

تنصح مستندات Documantation .NET. باستدعاء الطرق السابقة لاتقاء شر التحويلات الغير دقيقة (التضييق Narrowing)، كما انها ستؤدي إلى سرعة وزيادة في كفاءة التنفيذ، ولا تنسى ان حقول جداول Microsoft SQL Server® تحتوي على أنواع لن تجد مكافئ لها في إطار عمل .NET Framework، فالنوع SqlDecimal حجمه 38 بت، بينما النوع المقابل له في إطار عمل .NET Framework. حجمه لا يتجاوز 28 بت وهو المعروف بـ Decimal.

**ملاحظة**

الانواع التي تعود بها الطرق GetSqlxxx() معرفة في مجال الاسماء System.Data.SqlTypes وليس System.

الجدول الظاهر في الصفحة المقابلة يعرض لك هذه الطرق والنوع الذي تعود به في مجال الاسماء System.Data.SqlTypes لها، كما يشمل العامود الاخير النوع المكافئ لتعريف الحقول في جداول الخادم Microsoft SQL Server®:

| الطريقة | النوع الذي تعود به | المكافئ له في خادم SQL Server® |
|---|---|---|
| GetSqlBoolean() | SqlBoolean | bit |
| GetSqlByte() | SqlByte | tinyint |
| GetSqlInt16() | SqlInt16 | smallint |
| GetSqlInt32() | SqlInt32 | int |
| GetSqlInt64() | SqlInt64 | bigint |
| GetSqlSingle() | SqlSingle | real |
| GetSqlDouble() | SqlDouble | float |
| GetSqlDecimal() | SqlDecimal | decimal |
| GetSqlDateTime() | SqlDateTime | datetime<br>smalldatetime |
| GetSqlMoney() | SqlMoney | money<br>smallmoney |
| GetSqlString() | SqlString | char<br>nchar<br>ntext<br>nvarchar<br>sysname<br>text<br>varchar |
| GetSqlBinary() | SqlBinary | binary<br>varbinary<br>image<br>timestamp |
| GetSqlObject() | SqlObject | sql_variant |

## قراءة نتائج متعددة

بعض انواع مصادر البيانات (كخادم Microsoft SQL Server ®) تسمح لك بتنفيذ جملتي استعلام في وقت واحد، وذلك بفصل جمل الاستعلام بالفاصلة المنقوطة ";" :

```
SELECT * FROM [جدول الحسنوات] WHERE [الوزن] < 50;
SELECT * FROM [جدول الحسنوات] WHERE [الوزن] > 90
```

تنفيذ اكثر من جملة استعلام في امر واحد يزيد من سرعة تنفيذ جملة الاستعلام باضعاف اضعاف المرات، وذلك لان عملية المسح على سجلات الجدول ستكون واحدة وليس اثنتين. عند التعامل مع جمل الاستعلامات المتعددة، استدعي الطريقة NextResult() ان أردت قراءة نتائج جملة الاستعلام الاخرى والتي ستعود بالقيمة False ان لم توجد أي جملة استعلام اخرى:

```
Dim cn As New SqlConnection(connString)
cn.Open()
Dim cmd As New SqlCommand("SELECT * FROM [جدول الحسنوات]" _
    & "WHERE [الوزن] < 50; SELECT * FROM [جدول الحسنوات] " _
    & "WHERE [الوزن] > 90", cn)

Dim dr As SqlDataReader = cmd.ExecuteReader()

Do
    Do While dr.Read
        ListBox1.Items.Add(dr.Item("الاسم"))
    Loop
Loop While dr.NextResult

dr.Close()
cn.Close ()
```

كان هذا الفصل مدخلا لك لبرمجة قواعد البيانات باستخدام ADO.NET في الوضع المتصل Connected Mode، وبقي عرض الجزء الهام من ADO.NET والذي يستخدم في سيناريو الوضع المنفصل Disconnected Mode كما سترى بالفصل التالي. على فكرة، نسيت إخبارك بان اختصار ADO.NET هو ActiveX Data Objects وهو اسم مستحدث من تقنية ADO القديمة والمبنية على تقنية COM الأقدم منها في بنيتها التحتية –الشغلة شغلة تسويق يا عيال Microsoft!

# ADO.NET للوضع المنفصل

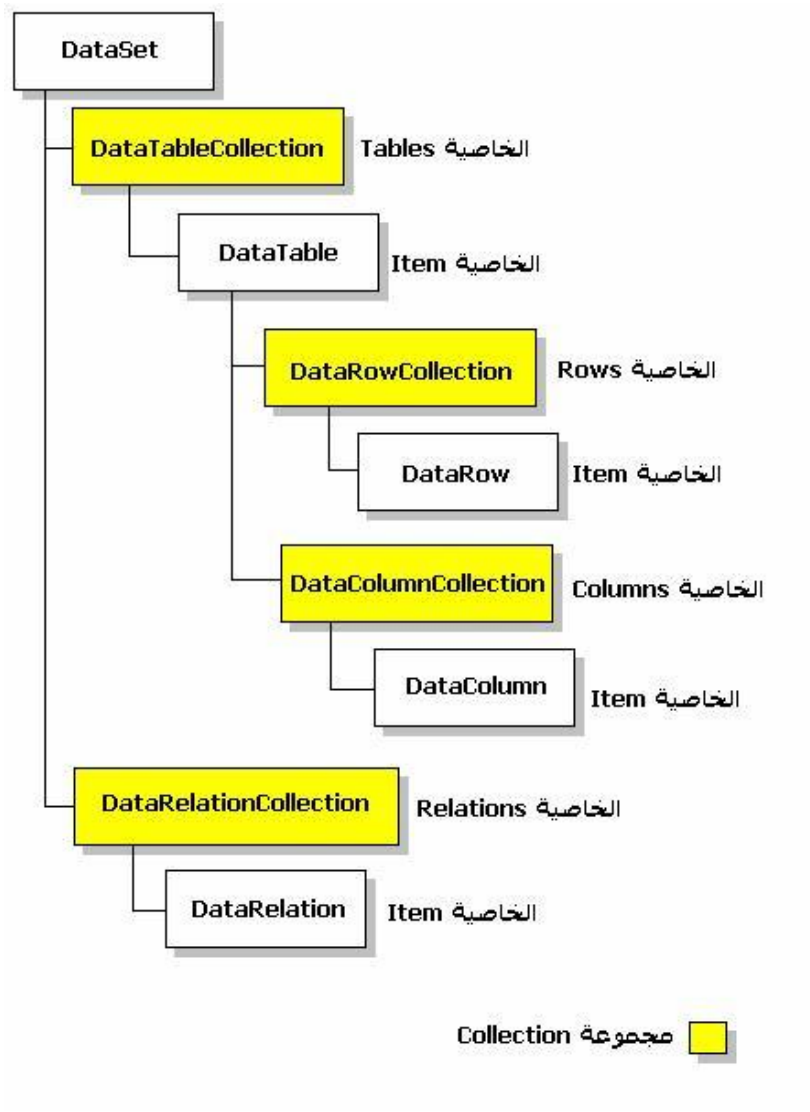
اسلوب الوضع المنفصل يوعد بكفاءة عالية عند الحديث عن تطبيقات العميل/الخادم Client-Server Applications، فمشاكل الازدحام والضغط الشديد على خوادم قواعد البيانات لن تحدث بعد الان، اذ ان عملية استخلاص البيانات ووضعها في أجهزة العملاء، سيتم معالجتها والتعامل معها في مصادر النظام الخاصة بكل عميل من العملاء، مما يخفف الضغط على الخادم.

في هذا الفصل سنرى كيف تعمل ADO.NET للاتصال المنفصل، وقد قسمته إلى قسمين رئيسين الاول يعرض لك الفئة DataSet لمعالجة البيانات، والثاني DataAdpater التي تمثل الجسر الذي سترسل به بياناتك بعد معالجتها.

## كائن البيانات DataSet

في سياق الوضع المنفصل، فان كائن البيانات هو DataSet وليس DataReader كما رأينا في الفصل السابق. الكائن DataSet هو كائن حاضن لعشرات الكائنات والمجموعات يمكنك مراجعة كافة تفاصيلها في مستندات NET Documantation. حيث يعرض لك **(الشكل 18-1 بالصفحة التالية)** بضعة فئات منها والتي تمكنك ليس فقط من تحرير السجلات، بل اعادة هيكلتها في تغيير حقولها او تعديل العلاقات بينها.

دعني انوه هنا بان **(شكل 18-1)** يبين لك علاقات **الاحتواء Containment** بين الفئات وليس الوراثة والاشتقاق الوراثي، وطريقة الوصول إلى الفئات المحضونة من الكائن الرئيسي تكون دائما عن طريق الخاصية Item لكل مجموعة من المجموعات. سترى كيف يتم ذلك بعد قراءتك للفقرات التالية.




**شكل 18-1**: علاقة الاحتواء بين بضعة فئات محضونة في كائن البيانات DataSet.

كما ترى في (**شكل 18-1**)، كائن البيانات يحتوي على مجموعتين رئيسيتين هما DataTableCollection و DataRelationCollection، الاولى تصل اليها عن طريق الخاصية Tables والثانية من الخاصية Relations:

```
Dim ds As New DataSet()
…
…
MsgBox(ds.Tables.Count)
MsgBox(ds.Relations.Count)
```

وكما ذكرت قبل قليل، إن اردت الوصول إلى احد كائنات المجموعة، فاستخدم دائما الخاصية Item والتابعة للمجموعة:

```
Dim ds As New DataSet()
Dim dt As DataTable

dt = ds.Tables.Item("جدول العملاء")
```

كما يمكنك استخدام حلقة For … Each او الطرق Add()، Insert()، Remove() …الخ فهي مجموعة يا عزيزي:

```
Dim ds As New DataSet()
Dim dt As DataTable

For Each dt In ds.Tables
    MsgBox(dt.TableName)
Next


Dim ds As New DataSet()
Dim dt As New DataTable()
…
…
ds.Tables.Add(dt)
```

الوصول إلى الكائنات المحضونة يتم من خلال الكائن الحاضن لها، او الكائن DataSet الرئيسي بشكل مباشر:

PDF

```
Dim ds As New DataSet()
Dim dc As DataColumn

For Each dc In ds.Tables.Item("جدول العملاء").Columns
    MsgBox(dc.ColumnName)
Next
```

اتمنى ان تكون قد اتضحت لك فكرة التعامل مع هذا الهرم الكائني، وان كانت لم تتضح بعد فاعتقد انك بحاجة إلى العودة إلى الفصل السادس **الفئات الأساسية** ومعرفة كيف التعامل مع المجموعات Collection (الفئات التي تحتوي على الواجهة ICollection)، اما ان كنت مدرك تماما للوضع فاستعن بالله وابدأ بالفئة DataTable.

## الفئة DataTable

الفئة DataTable تمثل جدول من جداول قاعدة البيانات، تحتوي على مجموعة كبيرة من الطرق والخصائص كالخاصية Rows (وهي مجموعة من النوع DataRowCollection) التي تمثل السجلات، والخاصية Columns (مجموعة من النوع DataColumnsCollection) تمثل أعمدة (حقول) الجدول.

سأتحدث عن الخاصيتين السابقتين في الفقرتين التاليتين، ودعني هنا اذكر لك الخاصية TableName والتي تمثل اسم الجدول والتي تستطيع الاستغناء عنها بارسال اسم الجدول كمشيد للفئة لحظة انشاء الكائن:

```
Dim Table1 As New DataTable()
Table1.TableName = "جدول2"

Dim Table2 As New DataTable("جدول2")
```

يمكنك تعريف **المفتاح الابتدائي Primary key** لكل جدول عن طريق اسناد مصفوفة من النوع DataColumns إلى الخاصية PrimaryKey، والسبب في كون القيمة التي تستقبلها الخاصية PrimaryKey هي مصفوفة يتضح إن علمت أن المفتاح الابتدائي يمكن ان يشمل اكثر من حقل:

```
Dim pkeys() As DataColumn = {New DataColumn(), New DataColumn()}
Dim myTable As New DataTable()

myTable.PrimaryKey = pkeys
```

PDF

اخيرا، مجموعة من الأحداث التابعة لهذه الفئة كالحدث ColumnChanging والذي يتم تنفيذه عندما يتم اضافة الحقول ثم يليه الحدث ColumnChanged، كذلك الحال مع الحدث RowChanging اضافة سجل جديد ليليه الحدث RowChanged، اما الاحداث RowDeleting و RowDeleted فيرتبطان بعمليات حذف السجلات.

## الفئة DataRow

الفئة DataRow تمثل سجل كامل من سجلات الجدول، لن تتمكن من انشاء كائن باستخدام New من الفئة DataRow، وانما يتحتم عليك إنشائه بطريقتين، الاول باستخدام الطريقة NewRaw() التابعة للفئة:

```
Dim table As New DataTable()
…
…

Dim dr As DataRow = Employees.NewRow()
```

والثانية تعريف مصفوفة من النوع Object تمثل قيم الحقول في السجل، ومن ثم تضيفها إلى الخاصية Rows والتابعة للكائن DataTable:

```
Dim dr2 As Object() = {"محمد عبدالله", 40, True}
table.Rows.Add(dr2)
```

ان عرفت كائن بالطريقة الاولى، فيمكنك اسناد قيم الحقول عن طريق الخاصية Item، كما يمكن ايضا قرائة الحقول من نفس الخاصية ان استخدمت الطريقة الثانية:

```
dr.Item("الاسم") = "محمد عبدالله"
dr.Item("العمر") = 40
dr.Item("متزوج") = True

Dim x As String = tables.Rows(0).Item("الاسم")
…
…
```

PDF Factory Pro trial version www.fastio.com

**ملاحظة**

الخاصية Item هي الخاصية الافتراضية للكائن DataRow، لذلك يمكنك تجاهلها ان اردت:

```
dr("الاسم") = "محمد عبدالله"
dr("العمر") = 40
dr ("متزوج") = True
```

## الفئة DataColumn

الفئة DataColumn تمثل حقل من حقول الجدول، تحتوي على مجموعة من الخصائص ابرزها خاصية اسم الحقل ColumnName والتي يمكنك ارسالها كمشيد:

```
Dim nameField As New DataColumn()
nameField.ColumnName = "حقل1"

Dim nameField2 As New DataColumn("حقل2")
```

خصائص تفيدك للحقول التي تنوي ان تجعلها كحقول معرفة، الخاصية AutoIncrement والتي تسند لها القيمة True ليتم زيادة قيمة الحقل مع كل سجل جديد بمقدار العدد الذي تضعه في الخاصية AutoIncrementSeed، وهناك ايضا الخاصيتان AllowDBNull و Unique اسند القيمة False إلى الاولى ان اردت منع القيم الخالية Null لهذا الحقل، واسند القيمة True إلى الثانية ان اردت عدم تكرار قيمة الحقل في اكثر من سجل:

```
Dim idField As New DataColumn()
With idField
    .AutoIncrement = True
    .AutoIncrementSeed = 1
    .AllowDBNull = False
    .Unique = True
End With
```

## الفئة DataRelation

العلاقات Relationships بين الجداول يمكنك إنشائها عن طريقة الفئة DataRelation، اسند كائن الجدول الأساسي في الخاصية ParentColumns، وكائن الجدول التابع في الخاصية ChildColumns، ولا تنسى تعريف اسم العلاقة في الخاصية RelationNames :

```
Dim Table1 As New DataTable()
Dim Table2 As New DataTable()
…
…
Dim rel As New Relation

rel.RelationName = "علاقة1"
rel.ParentColumns(Table1.Columns("حقل المفتاح الابتدائي"))
rel.ChildColumns(Table2.Columns("حقل المفتاح الاجنبي"))
```

كما يمكنك اختصار الاسنادات الثلاث الاخيرة عن طريق مشيد الفئة:

```
Dim rel As New DataRelation("علاقة1", _
    Table1.Columns("حقل المفتاح الابتدائي"), _
    Table2.Columns("حقل المفتاح الاجنبي"))
```

**ملاحظة**

لابد ان يكون كلا الجدولين معرفان في نفس كائن البيانات DataSet حتى تتمكن من تكوين علاقة بينهما.

## إنشاء كائن بيانات DataSet من الصفر

في الحقيقة، لن تحتاج إلى انشاء كائن بيانات DataSet بنفسك، اذ انك ستحصل على كائن بيانات جاهز من مصدر بيانات وتقوم بتعديله كما سترى في القسم التالي من هذا الفصل **الفئة DataAdapter**. مع ذلك، سأريك كيف يمكنك الربط بين كل ما تعلمته في الفقرات السابقة لانشاء كائن بيانات DataSet من الصفر.

في الخطوات التالية سنحاول تصميم جدولين (**شكل18-2**) تربطهما علاقة **1-ن** بين رقم المعرف للموظف، والعمليات التي قام بها.

| الموظفين | |
|---|---|
| المعرف | الاسم |
| 1 | عباس السريع |
| 2 | برعي ابو حبهة |
| 3 | عبود اللوح |

| المبيعات | | |
|---|---|---|
| رقم الموظف | دفع نقدي | المبلغ |
| 1 | نعم | ر.س. 100.00 |
| 1 | نعم | ر.س. 200.00 |
| 3 | لا | ر.س. 150.00 |
| 2 | نعم | ر.س. 120.00 |

**شكل 18-2**: كائن بيانات DataSet يمثل جدولان قمنا بانشائهما من الصفر.

PDF

الخطوة الاولى هي اسهل الخطوات والتي تتطلب منك فقط انشاء كائن من النوع DataSet:

```
Dim myDataSet As New DataSet()
```

قد تحتاج إلى ارسال اسم لكائن البيانات مع مشيد الفئة –امر اختياري:

```
Dim myDataSetAs New DataSet("المبيعات")
```

بعد ذلك، نبدأ بتصميم الجداول، أنشئ كائنين من النوع DataTable وارسل اسماء الجداول مع مشيداته –ايضا امر اختياري:

```
Dim Employees As New DataTable("الموظفين")
Dim Sales As New DataTable("المبيعات")
```

ان كنت تنوي باضافة الحقول، فيمكن تعريف كائنات من النوع DataColumn، ولكنـــي أفضل إضافتها بشكل مباشر مع الطريقة ()Add والخاصة بالمجموعة Columns لكل جدول:

```
With Employees.Columns.Add("المعرف", GetType(Integer))
    .AutoIncrement = True
    .AutoIncrementSeed = 1
    .AllowDBNull = False
    .Unique = True
End With
With Employees.Columns.Add("الاسم", GetType(String))
    .MaxLength = 100
End With

Sales.Columns.Add("رقم الموظف", GetType(Integer))
Sales.Columns.Add("دفع نقدي", GetType(Boolean))
Sales.Columns.Add("المبلغ", GetType(Decimal))
```

بالنسبة للحقل الاول والخاص بالجدول الاول، فاغلب الظن ان يكـــون هـــو **المفتـــاح الابتـــدائي Primary Key**، لذلك عليك اسناد مـــصفوفة مـــن النـــوع DataColumn إلـــى الخاصـــية PrimaryKey:

```
Employees.PrimaryKey = New DataColumn() {Employees.Columns("المعرف")}
```

بعد الانتهاء من تصميم حقول الجداول وتعريف المفاتيح المبدئية Primary keys، حان الوقت لإنشاء علاقة 1-ن تربط حقل **المعرف** من الجدول الاول إلى حقل **رقم الموظف** في الجدول الثاني، سنعرف كائن من النوع DataRelation بكل تأكيد:

```
Dim rel As New DataRelation("علاقة1", _
    Employees.Columns("المعرف"), _
    Sales.Columns("رقم الموظف"))
```

اخيرا، اهم خطوة هي الخطوة الاخيرة وهي اضافة كلا الجدولين إلى كائن البيانات DataSet، ولاتنسى اضافة العلاقة التي انجزها للتو:

```
myDataSet.Tables.Add(Employees)
myDataSet.Tables.Add(Sales)
myDataSet.Relations.Add(rel)
```

**تعبئة السجلات:**

كائن البيانات DataSet أصبح جاهزا لملئه بالسجلات، الشيفرة التالية تضيف سجل للجدول الاول والثاني:

```
Dim dr As DataRow = Employees.NewRow
Dim dr2 As DataRow = Sales.NewRow

dr("المعرف") = 1
dr("الاسم") = "عباس السريع"

dr2("رقم الموظف") = 1
dr2("دفع نقدي") = True
dr2("المبلغ") = 100

Employees.Rows.Add(dr)
Sales.Rows.Add(dr2)
```

وهذه بقية السجلات بطريقة مختصرة ومفضلة:

```
Employees.Rows.Add(New Object() {2, "برعي ابو جبهة"})
Employees.Rows.Add(New Object() {3, "عبود اللوح"})

Sales.Rows.Add(New Object() {1, True, 200})
Sales.Rows.Add(New Object() {3, False, 150})
Sales.Rows.Add(New Object() {2, True, 120})
```

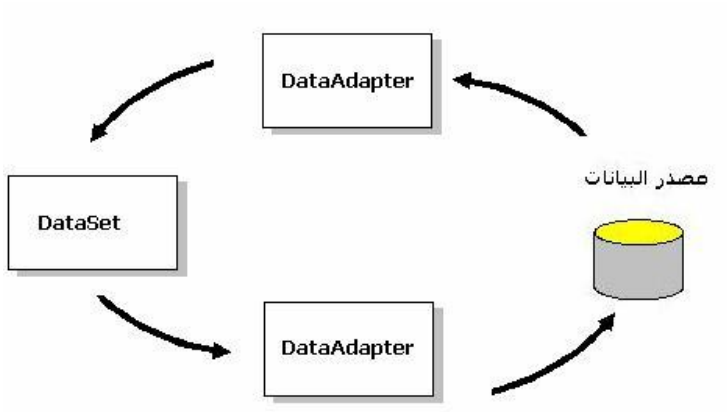
PDF

# كائن المحول DataAdapter

في القسم السابق من هذا الفصل، تعرضنا بشكل مختصر إلى الفئة DataSet، يمكنك مراجعة مكتبة MSDN لمعرفة باقي الفئات التي لم اتطرق لها، وفي رأيي الشخصي، لن تستخدم هذه الفئات بكثرة في برامجك الجدية، اذ ان اغلب استخدامك سيكون مختصر على التعامل مع السجلات فقط، وذلك لان كائن البيانات DataSet ستحمله من ملف قاعدة البيانات نفسه وسيأتيك زي البيضة المئشرة!

## سيناريو الوضع المنفصل

عند التعامل مع ADO.NET في الوضع المنفصل Disconnected Mode، فالسيناريو يوضحه لك (**شكل 18-3**) والذي يكون كالتالي: عندما تفتح اتصالا مع مصدر، اسند مرجع كائن الاتصال Connection إلى كائن DataAdapter، وهو يمثل همزة الوصل بين الكائن DataSet ومصدر البيانات، بعد ذلك ارسل البيانات المراد تعديلها إلى كائن DataSet وبذلك يمكنك انهاء علاقتك مع مصدر البيانات واغلاق الاتصال المنشئ بالكائن Connection.




**شكل 18-3**: سيناريو استخدام كائن المحول DataAdapter مع مصدر البيانات وكائن DataSet.

يمكنك الان معالجة البيانات وتحريرها وتعديلها عن طريق الكائن DataSet، وتذكر انها لا ترتبط بمصدر البيانات، فجميع العمليات ستتم في جهاز العميل Client، مما لا يشكل اي عمل او مهمة على جهاز مصدر البيانات الخادم Server.

بعد معالجة البيانات –الموجودة في الكائن DataSet- يمكنك ارسالها مرة اخرى إلى كائن DataAdapter والذي بدوره يحدث مصدر البيانات بالعمليات التي قمت بها، تستطيع ارسال الكائن DataSet بعد استقباله بدقيقة، ساعة، سنة او اي وقت كما يحلو لك. كما لا يشترط الحصول على الكائن DataSet من مصدر بيانات، اذ يمكنك بناء الكائن DataSet من الصفر – كما فعل في القسم السابق من هذا الفصل- وارساله إلى كائن DataAdapter بالرغم من ندرة حدوث هذا الشئ مع التطبيقات الحقيقية.

ميزة عظيمة عليك تقديرها والرفع من شأنها عندما تتعامل في الوضع المنفصل، وهي لا يشترط توافق نوع مزود مصدر البيانات الذي اخذت منه بيانات الكائن DataSet بنفس نوع مزود مصدر البيانات الذي سيتم ارسال بيانات الكائن DataSet بعد معالجتها، لتتمكن –مثلا- من الحصول على السجلات من قاعدة بيانات Microsoft Access ®، ثم معالجتها وتحديثها إلى خادم Microsoft SQL Server®.

## إنشاء كائن محول DataAdapter

لنبتعد قليلا عن الكلام النظري وننزل إلى مستوى الشيفرات المصدرية، ان كنت تنوي انشاء كائن محول DataAdapter فعليك تحديد نوع المزود المستخدم، فلو كنت تتعامل مع مصدر بيانات ذو مزود من النوع OLE DB .NET Data Provider، فعليك تعريف كائن من الفئة OleDbDataAdapter:

```
Dim msAccessDa As New OleDbDataAdapter()
```

اما ان كنت من مستخدمي المزود SQL Server .NET Data Provider، فالفئة SqlDataAdapter هي التي يفترض دائما استخدامها:

```
Dim msSQLDa As New SqlDataAdapter()
```

المزيد ايضا، كلا الفئتين OleDbDataAdapter و SqlDataAdapter مشتقتان وراثيا من الفئة القاعدية DbDataAdapter (والمشمولة في مجال الاسماء System.Data.Common)، لذلك قد تفيدك هذه الفئة القاعدة من تعريف إجراءات تستقبل وسيطات من مختلف انواع كائنات المحولات DataAdapter:

```
Sub UpdateDate(ByVal da As Common.DbDataAdapter)
    …
    …
    …
End Sub
```

**ملاحظة**

الفئة DbDataAdapter هي فئة مجردة Abstract Class، فقد عُرفت بالكلمة المحجوزة MustInherit، لذلك لن تتمكن من إنشاء كائنات منها باستخدام New او اي طريقة اخرى بشكل عام:

```
' احلق شواربي ان نجحت!
Dim x As New DbDataAdapter()
```

## الربط مع اتصال

الخطوة الاولى التي عليك إنجازها (سواء كنت تنوى قراءة او تحديث البيانات)، هي ربط كائن المحول DataAdapter مع كائن اتصال، توجد مجموعة من الاساليب التي تمكنك من عمل ذلك واسهلها ارسال مرجع كائن الاتصال Connection مع مشيد الفئة DataAdapter مع ضرورة ارسال جملة الاستعلام كوسيطة اولى:

```
Dim cn As New OleDbConnection(connString)
cn.Open()
…
…
Dim da As New OleDbDataAdapter("SELECT * FROM [جدول]", cn)
```

PDF Factory Pro trial version www.fastio.com

ولا تنسى ضرورة توافق نوع كائن المحول مع نوع كائن الاتصال، فلو كان كائن الاتصال يتبع مزود من النوع SQL .NET Data Provider فقد تم انشائه من الفئة SqlConnection، لذلك عليك الاعتماد على الفئة SqlDataAdapter عوضا عن OleDbDataAdapter للربط الصحيح مع الاتصال:

```
Dim cn As New SqlConnection(connString)
cn.Open()
…
…
Dim da As New SqlDataAdapter("SELECT * FROM [جدول]", cn)
```

من المهم التنويه هنا، بان كائن المحول DataAdapter مرتبط ارتباطا وثيقا بكائن الاتصال Connection، فلو تم اغلاق كائن الاتصال بالطريقة Close()، فلن تتمكن من الاستفادة من هذا الكائن (حاله كحال كائن الاوامر Command الذي تحدثت عنه في الفصل السابق).

قد تستغرب الجملة السابقة وتظن انها لا تناسب فكرة ADO.NET للوضع المنفصل، حيث اني ذكرت في بداية هذا القسم من الفصل انك ستغلق الاتصال بمجرد الحصول على البيانات، وانا هنا لا انوي ان أعارض احادثي بل سأزيد توضيحها بقول: يمكنك قطع الاتصال واغلاقه بالطريقة Close() متى ما شئت **ولكن بعد تحميل البيانات إلى كائن DataSet**.

المزيد ايضا، يمكنك الربط مع اتصال مباشرة دون الحاجة لتعريف كائن Connection بشكل واضح Explicitly، وذلك بارسال نص الاتصال Connection String مع مشيد الفئة:

```
Dim da As New OleDbDataAdapter("SELECT * FROM … …", _
"Provider=Microsoft.Jet.OLEDB.4.0;Data Source= … …")
```

لا تعتقد ان السطر السابق لا ينشئ كائن اتصال، بل قام بانشاء كائن اتصال من النوع Connection ايضا ولكنه محضون بداخل الكائن DataAdapter سيتم اغلاقه مباشرة بمجرد اغلاق الكائن الحاضن DataAdapter نفسه.

## قراءة البيانات

في هذه الفقرة سأريك كيف تقرأ البيانات، اما في الفقرة القادمة سنقوم بتحديث البيانات إلى مصدر البيانات. وحتى تتمكن من قراءة البيانات، عليك اولا وضع البيانات التي حصلت عليها من كائن المحول DataAdapter إلى كائن البيانات DataSet، يتم ذلك بارسال كائن البيانات DataSet المطلوب إلى الوسيطة الاولى للطريق Fill() والخاصة بالمحول DataAdapter:

```
Dim cn As New OleDbConnection(connString)
cn.Open()
Dim da As New OleDbDataAdapter("SELECT * FROM [جدول الحبايب]", cn)
Dim ds As New DataSet()
…
…
da.Fill(ds, "جدول الحبايب")
```

وبالمناسبة، لست بحاجة الان لا لكائن الاتصال ولا للاتصال الذي يحمله، فيمكنك الان اغلاق الاتصال بدون خوف على البيانات:

```
cn.Close()
```

**ملاحظة**

الوسيطة الثانية للطريقة ()Fill السابقة، تمثل اسم الجدول الذي تود ظهوره في الخاصية TableName التابعة للكائن DataTable والمحضون في الكائن DataSet (**شكل** 18-1).

بمجرد نقل البيانات من كائن المحول DataAdapter إلى كائن البيانات DataSet، فان البيانات موجودة في جهاز العميل الان وملكك تستطيع التصرف فيها كما تشاء، وكل ما هو مطلوب منك إتقان الهرم الكائني لكائن البيانات DataSet –رغم إيماني الشديد بان استخدام الفئة DataRow سيكون كافي إلى حد كبير لاغلب الاغراض:

```
Dim x As DataRow

For Each x In ds.Tables("جدول الحبايب").Rows
    ListBox1.Items.Add(x("الاسم"))
Next
```

**الأخطاء في عملية التحميل:**

كما ذكرت مرارا، عند التعامل مع قواعد البيانات، يفضل دائما تفادي الاستثناءات باستخدام التركيب Try…End Try او العبارة الأخرى On Error:

```
Try
    da.Fill(ds, "جدول الحبايب")
    …
    …
    …
Catch ex As Exception
```

PDF Factory Pro trial version www.fineprint.com

```
    …
    …
    …
End Try
```

مع ذلك، تتصحك مستندات NET. Documentation بالاعتماد على الحدث FillError عوضا عن تفادي الاستثناء بالطرق التقليدية، والسبب في ذلك ان اي عملية خطأ في تعبئة كائن البيانات DataSet (كوجود نوع في مصدر البيانات غير معرف في إطار عمل NET. Framework مثلا) ستؤدي إلى الغاء عملية تحميل كامل البيانات إلى الكائن DataSet حتى لو كان سبب حدوث الاستثناء حقل واحد.

الحدث FillError التابع للكائن DataAdapter يتم تنفيذه بمجرد حدوث اي خطأ في عملية تعبئة البيانات إلى الكائن DataSet (اي عند استدعاء الطريقة ()Fill)، يرسل هذا الحدث وسيطة من النوع FillErrorEventArgs تحتوي على الخاصية Errors التي تمكنك من معرفة الاستثناء ، كما يمكنك اسناد القيمة True إلى الخاصية Continue لإكمال عملية تحميل البيانات إلى كائن البيانات دون توقف:

```
Sub LoadDataSet()
    …
    …
    Dim da As New OleDbDataAdapter("… …", cn)
    Dim ds As New DataSet()

    AddHandler da.FillError, AddressOf FillError
    da.Fill(ds, "جدول الحبايب")
    …
    …
End Sub

Sub FillError(ByVal sender As Object, ByVal e As FillErrorEventArgs)
    If TypeOf e.Errors Is Exception Then
        …
        …
        …
        e.Continue = True
    End If
End Sub
```

## تحديث البيانات

يمكنك تحديث مصدر البيانات بكائن بيانات DataSet بعد معالجته بارساله مع الطريقة Update() التابعة للكائن DataAdapter، والتي تتطلب ايضا اسم الجدول في مصدر البيانات:

```
Dim da As New OleDbDataAdapter("[جدول]", cn)
Dim ds As New DataSet()
…
…
…
da.Update(ds, "جدول")
```

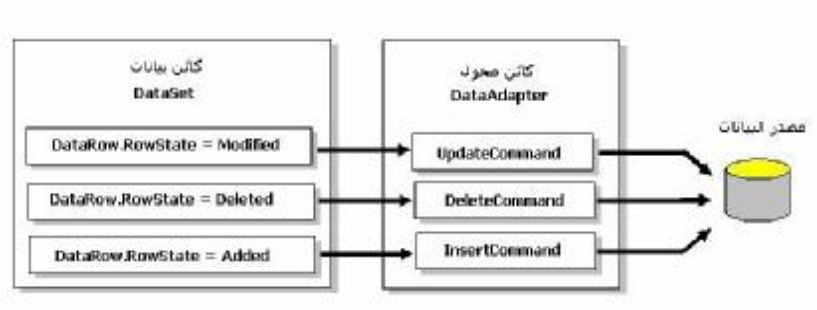
مهلا مهلا يا عزيزي، فالموضوع ليس بالبساطة التي كنت تتوقعها، اذا ان عملية ارسال كائن بيانات DateSet إلى محول DataAdapter قد تتأثر بالتعارضات التي تحدث على السجلات، فمثلا تخيل انك تود تحديث سجل تم حذفه من قبل مستخدم اخر، او انك تنوي اضافة سجل جديد يتعارض رقم مفتاحه الابتدائي Primary Key مع سجل اخر، او انك قد تضيف سجل يختلف في تركيبة حقوله عن تركيبة حقول مصدر البيانات الأساسي.

كما ترى في سطور الفقرة السابقة، نسبة حدوث التعارضات عند عملية تحديث البيانات إلى مصدر البيانات الاساسي كبيرة جدا، وسأتطرق لمعظم حالات هذه التعارضات لاحقا في القسم **اتقاء شر التعارضات**، اما هنا فدعنا نفترض ان هذه التعارضات لم ولن تحدث، ولنتخيل ان مستخدم قاعدة البيانات هو مستخدم واحد فقط، وذلك لتبسيط الامور وتوضيح الاساسيات قبل التعامل مع الحالات المعقدة.

كما اخبرتك قبل قليل، عندما تنوي تحديث مصدر البيانات وارسال السجلات بعد معالجتها ستستخدم الطريقة Update() والتي تعود بعدد السجلات التي تأثرت (تم اضافتها، حذفها، او تعديلها)، ولكن قبل استدعاء هذه الطريقة عليك التعامل مع ثلاث خصائص اخرى تابعة لكائن المحول DataAdapter ايضا هي InsertCommand، UpdateCommand، و DeleteCommand.

حتى تفهم كيف تسند القيم المناسبة للخصائص الثلاث الاخيرة، من الجيد جدا استيعاب طريقة عمل كائن المحول DataAdapter والتي تكون كالتالي: عندما تنوي تحديث البيانات إلى مصدر بيانات باستدعاء الطريقة Update()، سيقوم كائن المحول DataAdapter بالتحقق من الخاصية RowState والتابعة لكل كائن سجل DataRow من سجلات كائن البيانات DataSource، فان كانت Modified سيتم تنفيذ جملة الاستعلام في الخاصية UpdateCommand، وان كانت Deleted سيتم تنفيذ الامر التابع للخاصية

DeleteCommand، اما ان كانت قيمة الخاصية RowState لكائن الحقل هي Added فسيتم تنفيذ جملة الاستعلام SQL في الخاصية InsertCommand والتابعة لكائن المحول DataAdapter (**شكل 18-4**).




**شكل 18-4**: تحديد الخاصية المناسبة لتنفيذ جملة الاستعلام.

الخصائص InsertCommand، UpdateCommand، و DeleteCommand ليست خصائص حرفية، وانما خصائص من النوع كائن الأوامر Command (الذي تحدثت عنه في الفصل السابق)، مع ذلك لست بحاجة لاستخدامه، اذ يوفر لك إطار عمل NET. Framework كائن توليد الاوامر CommandBuilder والذي تفصله لك الفقرة الفرعية التالية.

**كائن توليد الاوامر CommandBuilder**:

الغرض من كائن توليد الامر CommandBuilder هو توليد جمل الاستعلام SQL بحيث توافق العمليات التي تحدثها على السجلات، جمل الاستعلام هذه تتعلق بالاوامر SELECT، INSERT، و UPDATE، وحتى يمكنك استخدام الكائن، أنشئه من الفئة OleDbDataAdapter وارسل مع مشيده كائن المحول DataAdapter:

```
Dim da As New OleDbDataAdapter("… …", cn)
Dim cmd As New OleDbCommandBuilder(da)
```

PDF Factory Pro trial version www.fastio.com

ولا تنسى في حالة استخدامك لكائن محول لمزود من النـــوع SQL Server .NET Provider، عليك تعريف كائن من الفئة SqlCommandBuilder:

```
Dim da As New SqlDataAdapter("… …", cn)
Dim cmd As New SqlCommandBuilder(da)
```

بعـــد إنـــشائك لكـــائن مولـــد اوامـــر Commandbuilder، يمكنـــك اســـتدعاء طرقـــه GetxxxCommand() لتـــسندها إلـــى خـــصائص كـــائن المحـــول DataAdapter الـــثلاث xxxCommand:

```
da.UpdateCommand = cmd.GetUpdateCommand
da.DeleteCommand = cmd.GetDeleteCommand
da.InsertCommand = cmd.GetInsertCommand
```

**مثال تطبيقي:**

دعني اريك مثالا لتحديث سجلات مصدر بيانات في الوضع المنفصل، أنـــشئ قاعـــدة باســـتخدام Microsoft Access® مثلا، وأضف جدولا بالاسم **جدول** يحتوي على ثلاث حقول هـــي **رقـــم المعرف، الاسم، والعمر**.

سنبدأ بفتح اتصال مع الجدول وقراءة بياناته، ومن ثم استدعاء الطريقة Fill() لكائن المحول DataAdapter حتى نتمكن من تعبئة كائن البيانات DataSet:

```
Dim cn As New OleDbConnection(connString)
cn.Open()
Dim da As New OleDbDataAdapter("SELECT * FROM [جدول]", cn)
Dim ds As New DataSet()

da.Fill(ds, "جدول")
```

الخطوة التالية هي تعديل السجلات، في الشيفرة التالية قمت بإجراء عملية تعـــديل للـــسجل الاول، حذف للسجل الثاني، واضافة سجل جديد عن طريق كائن البيانات DataSet:

```
With ds.Tables("جدول")
    .Rows(0)("الاسم") = "عباس السريع"
    .Rows(0)("العمر") = 99
End With

ds.Tables("جدول").Rows(1).Delete()
ds.Tables("جدول").Rows.Add(New Object() {Nothing, "برعي ابو جبهة", 55})
```

PDF Factory Pro trial version www.fastio.com

والان يمكننا تحديث قاعدة البيانات بعد إجراء التعديلات على سجلاتها باستدعاء الطريقة Update()، ولكن عليك اسناد قيم الخصائص xxxCommand قبل استدعائها، يتم ذلك بتعريف كائن مولد الاوامر CommandBuilder:

```
Dim cmd As New OleDbCommandBuilder(da)

da.UpdateCommand = cmd.GetUpdateCommand
da.DeleteCommand = cmd.GetDeleteCommand
da.InsertCommand = cmd.GetInsertCommand

da.Update(ds, "جدول")
```

نقطة هامة عليك أخذها بعين الاعتبار، وهي ان كائن مولد الاوامر CommandBuilder يعطيك نتائج خاطئة ان كانت اسماء الجداول او الحقول تحتوي على مسافات او حروف خاصة، فالجملة التالية:

```
SELECT * FROM جدول العمليات WHERE رقم المعرف = 4
```

جملة استعلام خاطئة، فهي لابد ان تكتب الأسماء بين القوسين [ و ] هكذا:

```
SELECT * FROM [جدول العمليات] WHERE [رقم المعرف] = 4
```

وحتى تضع هذه الاقواس، اسند قيمة في الخاصية QuotePrefix تحدد فيها الحرف الذي سيسبق اسم الجدول والحقل، والخاصية QuoteSuffix الحرف الذي يلي اسم الجدول او الحقل:

```
Dim da As New OleDbDataAdapter("… … …", cn)
Dim cmd As New OleDbCommandBuilder(da)

cmd.QuotePrefix = "["
cmd.QuoteSuffix = "]"

da.UpdateCommand = cmd.GetUpdateCommand
da.DeleteCommand = cmd.GetDeleteCommand
da.InsertCommand = cmd.GetInsertCommand
…
…
…
```

## تخصيص أفضل للخصائص xxxCommand

في الفقرة السابقة اعتمدنا على كائن مولد الاوامر CommandBuilder لتخصيص اوامر جمل الاستعلام SQL في خصائص كائن المحول xxxCommand، ولكن اكبر عيب من عيوب كائن مولد الاوامر CommandBuilder هو ضرورة القيام بمسح على مصدر البيانات للحصول على تركيبة البيانات (اي يقوم كائن الاوامر باستدعاء العبارة SELECT قبل توليد الاوامر) مما يسبب بطئ في عملية التحديث.

ولو كان الامر على المسح الإضافي الذي يجريه لهانت المشكلة، اذ ان خوارزم توليد جمل الاستعلام الذي يتبعه هذا الكائن يشترط وجود حقل لمفتح ابتدائي Primary Key في الجدول، مما يجعل استخدامه محصور على الجداول التي تحتوي على مفاتيح ابتدائية، ليس هذا فقط، بل يشترك ايضا تضمين حقل المفتاح الابتدائي في نفس جملة الاستعلام SELECT –المستخدمة لحظة إنشاء كائن المحول DataAdapter.

يبقى الحل الاخير هو بناء جمل الاستعلام بنفسك وإسنادها إلى الخصائص xxxCommand والتابعة لكائن المحول DataAdapter، وهو امر يتطلب خبرة منك في تعريف الوسيطات Parameters في جمل الاستعلام SQL.

بالنسبة لجمل الاستعلام والخاصة باضافة الحقول (التي تسندها إلى الخاصية InsertCommand)، فيمكنك الاكتفاء بكائن مولد الاوامر CommandBuilder وذلك ان الخوارزم الذي يتبعه يولد بشكل صحيح، اما الحديث عن جمل الاستعلام الاخرى والخاصة بالحذف او التعديل (التي تسندها إلى الخاصيتين UpdateCommand و DeleteCommand)، فعليك انشاء جمل استعلامها بنفسك عن طريق الكائن Command:

```
Dim delSQL As String
Dim updateSQL As String

delSQL = "DELETE FROM [جدول] WHERE [رقم المعرف] = ?"
updateSQL = "UPDATE [جدول] SET [الاسم] = ? , [العمر] = ? WHERE" _
            & " [رقم المعرف] = ?"

Dim cmdDel As New OleDbCommand(delSQL, cn)
Dim cmdUpd As New OleDbCommand(updateSQL, cn)
```

وحتى نخبر الكائن Command أننا استخدمنا وسيطات Parameters في جمل الاستعلام (علامات الاستفهام ? هي الوسيطات)، عليك استخدام المجموعة Parameters Collection لتعريف هذه الوسيطات والحقول التي تمثلها بالاضافة إلى نوع القيم التي تحملها:

PDF Factory Pro trial version www.fineprint.com

```
With cmdDel.Parameters.Add("@p1", GetType(Integer))
    .SourceColumn = "المعرف"
End With

With cmdUpd.Parameters.Add("@p1", GetType(String))
    .SourceColumn = "الاسم"
End With

With cmdUpd.Parameters.Add("@p2", GetType(Integer))
    .SourceColumn = "العمر"
End With

With cmdUpd.Parameters.Add("@p3", GetType(Integer))
    .SourceColumn = "رقم المعرف"
End With
```

والان كل ماهو مطلوب منك اسناد كائنات الاوامر إلى الخصائص المناسبة

```
Dim da As New OleDbDataAdapter("… … …", cn)
…
…
da.DeleteCommand = cmdDel
da.UpdateCommand = cmdUpd
…
…
```

# اتقاء شر التعارضات

عملية تحديث البيانات إلى مصادر البيانات ليست مفروشة بالزهور الوردية، بل ان حدائق الشوك احد سماتها، وكل ما فعلناه في الفقرات السابقة كان بافتراض ان كل شيء يتم على ما يرام، فعندما تحدث او تحذف سجل فانك متأكد من ان ذلك السجل موجود، وعندما تضيف سجل جديد فالمفتاح الابتدائي Primary key لا يتعارض مع اخر.

ولكن عندما يتعدد المستخدمين لقاعدة البيانات، فعليك –شئت ان أبيت- من اتقاء شر التعارضات، واسهل ما يمكنك القيام به هو استدعاء الطريقة Update() بين فكي التركيب Try … End Try:

```
Try
    …
    da.Update(ds, "جدول")
Catch
    …
End Try
```

PDF

يعيب الاسلوب السابق، ان اي خطأ في عملية تحديث احد السجلات إلى مصدر البيانات سيوقف عملية التحديث لكافة السجلات الأخرى (والموجودة في كائن البيانات DataSet)، لذلك قد تسند القيمة True إلى الخاصية ContinueUpdateOnError حتى تمكن كائن المحول من متابعة تحديث باقي السجلات ان حدث خطأ في عملية تحديث احدها:

```
da.ContinueUpdateOnError = True
da.Update(ds, "جدول")
```

ويمكنك معرفة ان حدث فعلا خطأ لحظة التحديث عن طريق الخاصية HasChanges والتابعة لكائن البيانات DataSet –وليس المحول DataAdapter:

```
If ds.HasChanges() Then
    …
    …
End If
```

يمكنك الاستعانة بالحدث RowUpdated التابع لكائن المحول DataAdapter، والذي يتم تنفيذه بعد عملية تحديث كل سجل من السجلات، الا انك لن تحتاج اليه في اغلب الاحوال، حيث ساعرض لك في الفقرة التالية طريقة يمكنك اتباعها لمعرفة السجلات التي حدثت بها التعارضات.

## عرض التعارضات

الاسلوب الاول البرمجي مفضل لدى اغلب المبرمجين، والسبب هو توفير عناء كتابة الشيفرة المصدرية وازالة العبء عنهم، بل وإعطاء المستخدم الحرية المطلقة في عمل ما يريد. الفكرة من هذا الاسلوب ببساطة هي عدم فعل اي شيء وابقاء السجلات **الغير محدثه** كما هي، وكل ما يتطلبه منك كتابة نص -او اي شيء اخر- في الخاصية RowError والتابعة للكائن DataRow، ومن المهم استدعاء الطريقة ()RejectChanges للسجلات التي كنت ترغب في حذفها وفشلت، وذلك حتى لا تختفي من كائن الجدول DataTable التابع لكائن مصدر البيانات DataSet:

```
Dim da As OleDbDataAdapter
Dim ds As DataSet
…
…

da.ContinueUpdateOnError = True
da.Update(ds, "جدول")
```

```
If ds.HasChanges Then
    Dim dr As DataRow

    For Each dr In ds.Tables("جدول").Rows
        If dr.RowState = DataRowState.Added Then
            dr.RowError = "خطأ اثناء اضافة هذا السجل"
        ElseIf dr.RowState = DataRowState.Modified Then
            dr.RowError = "خطأ اثناء تعديل هذا السجل"
        ElseIf dr.RowState = DataRowState.Deleted Then
            dr.RowError = "خطأ اثناء حذف هذا السجل"
            ' استدعي هذه الطريقة حتى لا يتم
            ' حذف السجل من كائن البيانات
            ' DataSet
            dr.RejectChanges()
        End If
    Next
End If
```

## الحدث RowUpdated

عملية تصحيح التعارضات تلقائيا عملية ليست مستحيلة، ولكنها تعتمد اعتماد كلي على اسلوبك ومتطلبات المشروع الذي تنجزه، فبعض المبرمجين سيقومون بنقل السجلات المتعارضات إلى جدول خاص بها، واخرون قد يحفظونها في ملفات نصية، وهناك من سيحاول اضافة سجل جديد ونسخه ان تعارضت عملية تحديث سجل مع مستخدم اخر. لذلك، دعني اعود إلى الحدث RowUpdated واخبرك كيف يمكنك الاستفادة منه لحل مشاكل التعارضات بنفسك:

```
Dim WithEvents da As OleDbDataAdapter

Private Sub da_RowUpdated(ByVal sender As Object, _
    ByVal e As System.Data.OleDb.OleDbRowUpdatedEventArgs) _
    Handles da.RowUpdated
    …
    …
End Sub
```

الحدث RowUpdated يتم تنفيذه في كل مرة يتم فيها تحديث سجل وارساله إلى مصدر البيانات (سواء نجحت عملية التحديث او فشلت للسجل)، الوسيطة التي يرسلها الحدث تحتوي على مجموعة من الخصائص، منها الخاصية StatementType والتي تعود بقيمة تمثل نوع العملية التي أجريت على السجل فيما لو كانت Insert اضافة للسجل، Delete حذف، او Update تحديث. وبالنسبة للقيمة Select فلا تستخدمها لانها لن تحدث.

الخاصيتين Command و Row تعودان بمرجع لكائنين، الاولى من نوع كائن الاوامر Command يمثل الامر الذي استخدم لحظة تحديث السجل، وهو نفس الكائن الموجود في الخصائص UpdateCommand، DeleteCommand، وInsertCommand. اما الخاصية Row فهي تعود بكائن من النوع DataRow يمثل السجل الذي تم تحديثه.

بالنسبة للخاصية RecordsAffected فهي تعود بعدد السجلات التي تأثرت لحظة التحديث، ومن البديهي ستكون قيمتها 1 ان تم كل شيء على ما يرام، او صفر ان فشلت عملية تحديث السجل. اما الخاصية Errors فهي ستعود بكائن استثناء Exception يفيدك كثيرا لمعرفة تفاصيل اكثر عن الخطأ الذي منع عملية تحديث السجل، وان كان نوع هذا الاستثناء هو DBConcurrencyException فذلك يعني حدوث تعارض، وغير ذلك فقد يكون خطأ في عدم وجود صلاحيات لك في نفس مصدر البيانات.

اهم خاصية من خصائص وسيطة الحدث RowUpdated هي الخاصية Status، والتي يمكنك اسناد القيمة Continue لها ان اردت اكمال عملية التحديث، القيمة ErrorsOccurred والتي تجعل الحدث يعبر عن فشل عملية تحديث السجل كخطأ، القيمة SkipCurrentRow تتجاهل عملية تحديث السجل، او القيمة SkipAllRemainingRows التي تتجاهل عملية تحديث السجل الحالي وكافة السجلات اللاحقة.

كم رأيت، التعامل مع فئات ADO.NET في الوضع المنفصل ليس بالأمر الهين ويتطلب منك الحذر الشديد رغم انه يعطيك مرونة كبيرة في عملية نقل البيانات من مصدر بيانات إلى اخر، كما انك ستعتمد عليه ايضا ان كنت تود ربط أدوات Windows Forms بمصادر البيانات كما سنرى في الفصل التالي.

PDF

# ربط البيانات والتكامل مع XML

يمكنك عرض البيانات على المستخدم يدويا باستخدام الفئات السابق ذكرها، إلا ان عملية ربط البيانات بالأدوات ستوفر عليك الكثير من الجهد والوقت. كما تكامل ADO.NET مع لغة وصف البيانات XML سيوفر عليك عشرات الشيفرات ان اردت تبادل البيانات مع التطبيقات الاخرى سواء في الشبكة المحلية أو على الانترنت أو بشكل خدمات ويب Web Services.

في هذا الفصل سأتطرق إلى موضوعين مستقلين، الاول يختص بربط أدوات Windows Forms مع مصادر البيانات، والثاني يتحدث عن سمات XML التي يمكن ان تجنيها عند التعامل مع ADO.NET.

## ربط البيانات

**ربط البيانات Data Binding** هي عملية ربط مصدر بيانات Data Source بأداة من أدوات Windows Form بحيث تمكن المستخدم من قرائة مصدر البيانات وتحديثه ايضا عن طريق الأدوات (كالاداة TextBox، ListBox ...الخ).

ربط مصادر البيانات بأدوات Windows Forms سيسهل عليك الكثير ويوفر عليك عناء كتابة الشيفرات المصدرية الطويلة، فكل ما هو مطلوب منك اعداد بضعة خصائص للادوات وربطها مع مصدر البيانات، وستصبح النافذة موجه إلى مصدر البيانات، وتمكن المستخدم ايضا من التنقل بين سجلاتها بالطرق التقليدية (كاضافة زر التالي Next، السابق Back، مؤشر رقم السجل الحالي، ...الخ) (**شكل 19-1**).

PDF

**شكل 19-1**: ربط الادوات بمصدر بيانات.

## أنواع الربط

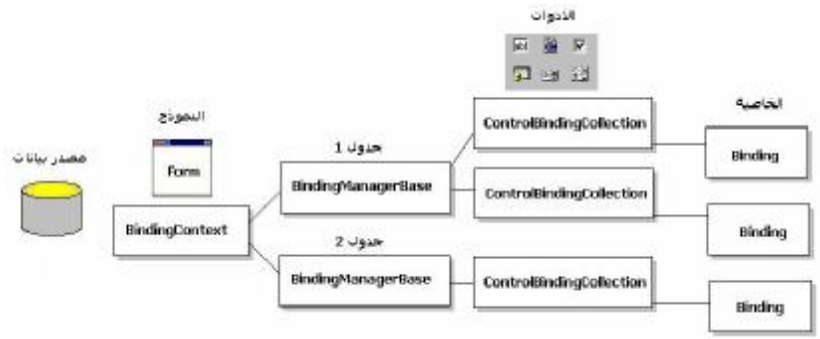
من منظور تحديث البيانات إلى مصادر البيانات، فيوجد نوعين من الربط هما **ربط البيانات أحادي الاتجاه One-way data binding**، **وربط البيانات ثنائي الاتجاه Two-way data binding**. في الربط الأحادي فان الأداة سيتم ربطها بمصدر البيانات على شكل للقراءة فقط ReadOnly، ولن يتم تحديث مصدر البيانات عند إجراء أي تعديل على محتويات الأداة المربوطة به. وفي الربط ثنائي الاتجاه، فان الأدوات سيتم ربطها بنسق قابل للقراءة والكتابة، واي تعديل في محتوى الأداة المربوطة سيتم ابلاغ وتحديث مصدر البيانات بشكل تلقائي.

اما من منظور عدد الخصائص والأدوات المربوطة بمصدر البيانات، فيمكننا ان نصنفها إلى نوعين هما **الربط البسيط Simple Binding والربط المعقد Complex Binding**، في الربط البسيط فانك تربط خاصية واحدة للأداة بحقل مكافئ لها بمصدر البيانات، كأن تربط –مثلا- الخاصية Text لأداة TextBox بحقل الاسم في مصدر البيانات. اما في الربط المعقد، فانك تربط مجموعة قيم وخصائص للأداة مع أكثر من حقل (أو أكثر من سجل) في مصدر البيانات، كان تربط –مثلا- المجموعة Items والخاصية بالاداة ListBox بمجموعة سجلات تمثل المبيعات التي قام بها الموظف.

ميزة اخرى تجنيها من ربط البيانات بالأدوات وهي عدم حصر عملية الربط على نوع معين من مصادر البيانات، بل يمكن لاي كائن يحتوي على الواجهة IList ان تقوم بربطه، كان تربط مصفوفة Array بالادوات، أو تستخدم كائنات ADO.NET لربط بقواعد البيانات -كما سترى لاحقا. ودعنا الان نرى كيفية عمل الربط حتى تستوعب الفكرة بالفقرة التالية.

## ميكانيكية الربط

يمكنك استخدام بيئة التطوير .NET Visual Studio لربط أدواتك مع مصدر البيانات باستخدام الفأرة، إلا ان استيعاب ميكانيكية الربط أمر في غاية الأهمية حتى تتقن استخدام الشيفرة والكائنات التي تنجزها (**شكل 19-2**).




**شكل 19-2**: ميكانيكية ربط البيانات Data Binding.

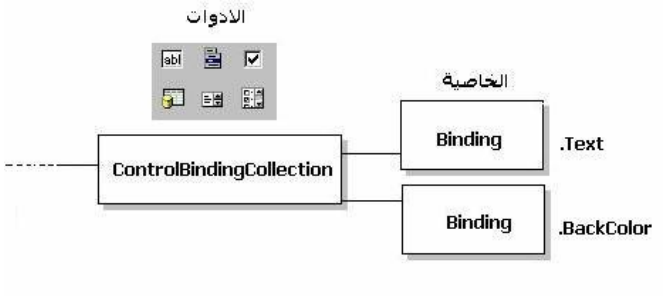
لنبدأ من اقصى اليسار ونتحدث عن الكائن BindingContext، الخاصية التي تحمل نفس اسم الكائن والتابعة لنافذة النموذج (أو الاداة الحاضنة ايضا) تعود بكائن من هذا النوع، والذي يمثل رابط لمجموعة من الادوات مع مصدر البيانات، والحاجة اليه تتضح ان علمت ان نافذة النموذج قد يتم ربطها مع اكثر من مصدر بيانات.

الخاصية BindingContext السابقة تتطلب وسيطة تمثل مصدر البيانات، لتعود بكائن من النوع BindingManagerBase، وهو الحاضن أو الرابط الرئيسي للادوات مع مصدر البيانات، يمكن الاستفادة منه –مثلا- في تعيين قيمة لخاصيته Position، والتي تمكنك من تغيير موقع السجل الحالي الذي تعرضه الادوات.

تحتوي كل اداة من ادوات Windows Forms على الخاصية DataBindings، والتي تمثل مجموعة من النوع ControlBindingCollection. في هذه المجموعة، تستطيع اضافة، حذف، أو تعديل كائنات الربط والتي من النوع Binding، مع العمل ان يمكن للاداة ان تحتوي على اكثر من كائن ربط.

كائن الربط Binding هو حجر الاساس في عملية **ربط خاصية واحدة فقط بحقل واحد فقط** بمصدر البيانات، وان اردت ربط اكثر من خاصية باكثر من حقل في نفس الاداة، عليك انشاء كائن ربط جديد.

من الكلام السابق، يتضح لنا انه يمكن للاداة الواحدة من ان تربطها باكثر من حقل وذلك بتكوين علاقة بين احد خصائصها والحقل المطابق لها في مصدر البيانات، وذلك في عدد كائنات الربط Binding التابعة للمجموعة ControlBindingCollection والخاصة بكل اداة، فيمكنك مثلا انشاء كائنين ربط بين الخاصية Text والحقل CarName، والخاصية Backcolor والحقل CarColor بنفس الاداة ونفس مصدر البيانات (**شكل 19-3**).




**شكل 19-3**: ربط أكثر من خاصية للأداة بأكثر من حقل.

## الربط إلى مصفوفة

في اغلب برامجك الجدية، فانك ستقوم بربط الادوات مع مصادر بيانات لقواعد بيانات تقليدية، ويمكنك عمل ذلك باستخدام فئات ADO.NET، ولكني فضلت إعطائك طريقة ربط الادوات بمصادر البيانات من النوع قيم لمصفوفات حتى تسهل عليك عملية استيعابها، عرف الفئة التالية:

```
Class Person
    Private m_name As String
    Property Name() As String
        Get
            Return m_name
        End Get
        Set(ByVal Value As String)
            m_name = Value
        End Set
    End Property
```

```
    Private m_age As Integer
    Property Age() As Integer
        Get
            Return m_age
        End Get
        Set(ByVal Value As Integer)
            m_age = Value
        End Set
    End Property

    Private m_married As Boolean
    Property Married() As Boolean
        Get
            Return m_married
        End Get
        Set(ByVal Value As Boolean)
            m_married = Value
        End Set
    End Property


    Sub New(ByVal name As String, ByVal age As Integer, _
        ByVal married As Boolean)

        Me.Name = name
        Me.Age = age
        Me.Married = married
    End Sub
End Class
```

يمكنك تعبئة مصفوفة من الفئة السابقة بقرائها من ملف نصي أو كتابتها برمجيا:

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    …
    …
    Dim Persons() As Person = {New Person("A", 1, True), _
        New Person("B", 2, False), New Person("C", 3, True)}
End Class
```

اضف أداتي TextBox واداة CheckBox في نافذة النموذج، واضف كائن ربط Binding في كل خاصية DataBindings لكل اداة، يمكنك تعريف كائن ربط باستخدام الكلمة المحجوزة New، أو اضافته مباشرة بالطريقة ()Add التابعة للخاصية DataBindings (وهي المجموعة ControlBindingCollection)، وفي كلا الحلتين فيتطلب كائن الربط Binding اسم الخاصية ومصدر البيانات واسم الحق في مصدر البيانات:

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    …
    …
    Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load

        Dim b As New Binding("Text", Persons, "Name")

        txtName.DataBindings.Add(b)
        txtAge.DataBindings.Add("Text", Persons, "Age")
        CheckBox1.DataBindings.Add("Checked", Persons, "Married")
    End Sub

End Class
```

بهذا تكون انتهيت من عملية ربط الادوات بمصدر البيانات، مع ذلك قد تحتاج إلى اضـــافة ازرار Button لتمكن المستخدم من التنقل بين السجلات، يتم ذلك باسناد القيمة المناسبة للخاصية Position والتابعة للكائن المحضون BindingManagerBase والذي تصل اليه من الخاصـــية BindingContext:

```
Public Class Form1
    Inherits System.Windows.Forms.Form
    …
    …
    Private Sub cmdNext_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdNext.Click

        Me.BindingContext(Persons).Position += 1
    End Sub

    Private Sub cmdBack_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdBack.Click

        Me.BindingContext(Persons).Position -= 1
    End Sub

    Private Sub cmdLast_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdLast.Click

        Me.BindingContext(Persons).Position = _
            Me.BindingContext(Persons).Count - 1
    End Sub
```

```
    Private Sub cmdFirst_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles cmdFirst.Click

        Me.BindingContext(Persons).Position = 0
    End Sub
End Class
```

## الربط باستخدام ADO.NET

عملية الربط باستخدام ADO.NET لا تختلف كثيرا عن ما فعلنا بالمصفوفة Persons السابقة، فكل ما هو مطلوب منك استبدال مصدر البيانات ووضع الكائن DataSet:

```
Dim cn As New OleDbConnection(connString)
cn.Open()
Dim da As New OleDbDataAdapter("SELECT * FROM [جدول]", cn)
Dim ds As New DataSet()
da.Fill(ds, "جدول")
cn.Close()

txtName.DataBindings.Add("Text", ds, "جدول.الاسم")
```

صحيح ان عملية الربط هي ثنائية الاتجاه Two-way data binding، ولكن عملية التحديث ستتم على كائن البيانات DataSet فقط، وذلك هو اسلوب ADO.NET للوضع المنقطع Disconnected Mode، ويبقى الباقي للقيام بعملية التحديث الفعلية في مصدر البيانات باستدعاء الطريقة Update() لكائن المحول DataAdapter –كما تعلمنا سابقا في الفصل الثامن عشر **ADO.NET للوضع المنقطع**.

**ملاحظة**

تمكنك بيئة التطوير Visual Studio .NET من توليد الشيفرات الضرورية لربط الادوات بمصادر البيانات دون الحاجة لكتابتها بنفسك، وذلك بفتح نافذة مستكشف الخوادم Server Explorer (تصل اليها من قائمة View)، ومن ثم تحديد الجدول المراد ربط وسحبه وإلقائه Drag & Drop إلى نافذة النموذج، لتستخدم الفأرة فقط في ربط الادوات مع مصدر البيانات (راجع مستندات NET Documentation. لمزيد من التفاصيل).

PDF

المزيد ايضا، يحتوي الكائن BindingManagerBase (**شكل 19-2**)، علـــى مجموعـــة كبيرة من الطرق والخـــصائص، كالطريقـــة AddNew() لإضـــافة ســـجل جديـــد، الطريقـــة CancelCurrentEdit() لالغاء التحديثات، والطريقة RemoveAt() لحذف سجل، كما يمكنـــك قنص حدثه PositionChanged الذي يتم تنفيذه بمجرد انتقال المؤشر إلى سجل اخر:

```
Dim WithEvents Bmb As BindingManagerBase

Private Sub Form1_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load
    …
    …
    Dim ds As New DataSet()
    …
    …
    Bmb = Me.BindingContext(ds)
End Sub

Private Sub Bmb_PositionChanged(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Bmb.PositionChanged

    ' اكتب الاكواد هنا
    …
    …
End Sub
```

## الربط المعقد Complex Binding

ان عمليات الربط التي قمنا بها في الفقرات السابقة، تصنف من ضمن الربط البـــسيط Simple Binding، وذلك لان ميكانيكية الربط تشمل **خاصية واحدة لحقـــل واحـــد** –حتـــى وان تعـــددت الخصائص المربوطة بأكثر من حقل في نفس الأداة.

اما الربط المعقد Complex Binding فلا تدعمه كافة الادوات، وان أردت تطبيقه فعليك استخدام أداة من الأدوات ListBox، ComboBox، أو DataGrid، بحيث تمكن من ربط الأداة باكثر من سجل أو اكثر من حقل.

اكثر ما يفيدك الربط المعقد لمصادر البيانات لحظة العلاقات Relationships بين الجداول، فافترض مثلا ان لدينا جدول الموظفين يحتوي على مجموعة من حقول كاسم الموظف، الراتـــب، العمر، وغيرها، بالاضافة إلى حقل يمثل القسم الذي يعمل فيه الموظف وهذا الحقل سيكون مربوط بعلاقة مع جدول اخر يمثل الاقسام وأسمائها.

إن أردت ربط الأدوات بالجدول الاول، فقد تتبع أسلوب الربط البسيط Simple Binding، ولكن يعيبه ان المستخدم سيضطر إلى كتابة رقم القسم الذي يعمل فيه الموظف والمـــشمول فـــي

PDF

العلاقة مع جدول الاقسام (**شكل 19-4 A**). اما ان اردت التسهيل عليه، فيمكنك استخدام الاداة ComboBox التي تعرض جميع سجلات جدول الاقسام، وتمكن المستخدم ايضا من تحديد القسم الذي يعمل فيه الموظف باستخدام الفأرة ودون الحاجة إلى كتابة رقم القسم يدويا (**شكل 19-4 B**).

**شكل 19-4**: استخدام الاداة ComboBox للربط المعقد.

ان اردت استخدام الاداة ComboBox بهذه الطريقة، فعليك اسناد القيم المناسبة لثلاث خصائص هي: DisplayMember، ValueMember، و DataSource. في الخاصية الاولى تحدد الحقل الذي تود عرض سجلاته في الاداة (سيكون اسم القسم في مثالنا)، وفي الثانيه تحدد فيها الحقل الذي تود استخدام قيمته الفعلية (سيكون رقم القسم)، اما الاخير فتحدد فيها كائن مصدر البيانات. قد تسند الخصائص السابق بهذا الشكل:

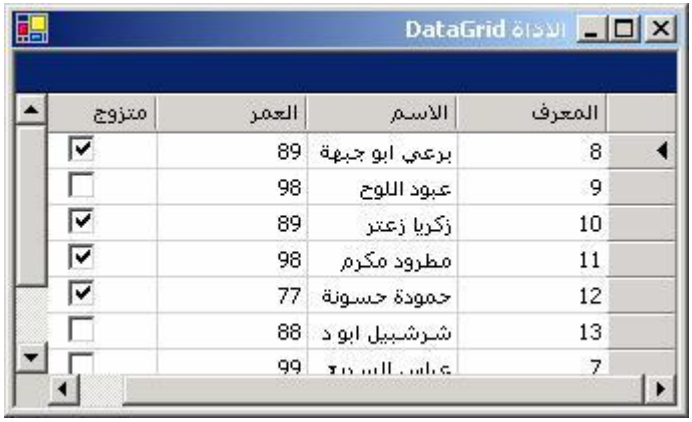
```
Dim da2 As New OleDbDataAdapter("SELECT * FROM [جدول الاقسام]", cn)
Dim ds2 As New DataSet()
da2.Fill(ds2, "جدول الاقسام")

ComboBox1.DisplayMember = "[جدول الاقسام].[اسم القسم]"
ComboBox1.ValueMember = "[جدول الاقسام].[رقم القسم]"
ComboBox1.DataSource = ds2
```

بعد اسنادك للقيم المناسبة، تأتي عملية ربط اخرى لنفس الاداة ComboBox1 بمصدر البيانات الاول (جدول الموظفين) وذلك لاننا نريد تغيير قيمة حقل "رقم القسم" في الجدول الاول بالاختيار الذي يحدده المستخدم في الخاصية SelectedValue:

```
' ds هو كائن مصدر البيانات الاول
ComboBox1.DataBindings.Add("SelectedValue", ds, _
    "[جدول الموظفين].[رقم القسم]")
```

اجمل ربط معقد تنجزه لك الاداة DataGrid بحيث تعرض لك كامل جدول كائن البيانات DataSet بكافة حقوله وسجلاته، وكل ما تطلبه منك هو مصدر البيانات في الخاصية DataSource، والجدول المراد ربطه في الخاصية DataMember (**شكل 19-5**):

```
Dim da As New OleDbDataAdapter("SELECT * FROM [جدول]", cn)
Dim ds As New DataSet()
…
…
da.Fill(ds, "جدول")

DataGrid1.DataSource = ds
DataGrid1.DataMember = "جدول"
```




**شكل 19-5**: الربط المعقد باستخدام الأداة DataGrid.

# فئات خاصة بـ XML

في الحقيقة، لست بحاجة إلى استخدام فئات خاصة بلغة وصف البيانات XML ان اردت الاستفادة من سمات XML والتي توفرها لك فئات ADO.NET، إلا أنني فضلت عرض بضعة فئات من اطار عمل NET Framework. قد تحتاج لها يوما من الايام عند تعاملك مع XML.

**ملاحظة**

عند استخدام فئات XML قد تحتاج إلى استيراد مجال الاسماء التالي:

Imports System.XML

كما يمكنك قراءة الملحق أ: **لغة وصف البيانات** XML في نهاية هذا الكتاب، للحصول على مقرر سريع حول لغة وصف البيانات XML ان كنت تجهلها.

## الفئة XmlTextReader

كما كنا سابقا –في الفصل الثامن **الملفات والمجلدات**- نعتمد على الفئة StreamReader لقراءة الملفات النصية، والفئة BinaryReader للملفات الثنائية، فإننا سنعتمد على الفئة XmlTextReader ان اردت فتح وقراءة الملفات بصيغة XML.

مع ذلك، طريقة التعامل مع كائن XmlTextReader ليس كالتعامل مع كائنات StreamReader أو BinaryReader، وانما يشبه إلى حد كبير التعامل مع الكائن DataReader والذي استخدمنها للوضع المتصل Connected Mode في الفصل السابع عشر **استخدام ADO.NET**.

ان كان الكائن DataReader يتطلب كائن Command ليتم إنشائه، فان الكائن XmlTextReader لايحتاج منك سوى اسم الملف ترسله مع مشيده:

```
Dim xmlData As New XmlTextReader("C:\mydata.xml")
```

وكما تعلم، ملفات XML ما هي إلا ملفات بيانات أشبه بجداول قواعد البيانات، ستستعلم عنها في حلقة Do … Loop لتستدعي الطريقة Read() –تماما كما فعلت مع الكائن DataReader:

```
Dim xmlData As New XmlTextReader("C:\mydata.xml")

Do While xmlData.Read
    …
    …
    …
Loop
```

PDF

يفضل دائما معرفة نوع القيمة التي تقرأها من ملف XML، كأن تكون عنصر جذري للملف XML Document، أو عنصر XML Element، قيمة سجل، واصفة XML Attribute، تعليق Comment ...الخ، عن طريق الخاصية NodeType:

```
Do While xmlData.Read
     Select Case xmlData.NodeType
         Case XmlNodeType.Document
             …
             …
         Case XmlNodeType.Attribute
             …
             …
         Case XmlNodeType.Element
             …
             …
         Case XmlNodeType.Entity
             …
             …
     End Select
 Loop
```

تستطيع قراءة العنصر أو أي قيمة تقرأها عن طريق الخاصية Name:

```
Do While xmlData.Read
   If xmlData.NodeType = XmlNodeType.Element Then
       MsgBox ("<" & xmlData.Name & ">")
   End If
Loop
```

اخيرا، لا تنسى إغلاق الملف دائما باستدعاء الطريقة Close():

```
xmlData.Close()
```

## الفئة XmlTextWriter

تنشئ كائنات من الفئة XmlTextWriter ان اردت كتابة ملفات XML، واستخدامها سهل جدا ولا يحتاج إلى تفصيل، فكل ماهو مطلوب منك ارسال اسم الملف وصفحة المحارف Encoding مع المشيد:

```
Dim xmlData As New XmlTextWriter("C:\Test.xml", _
    System.Text.Encoding.UTF8)
```

PDF Factory Pro trial version www.fastio.com

ومن ثم البدء باستدعاء الطريقة WriteStartDocument حتى تتمكن من البدء في تعبئة الملف:

```
xmlData.WriteStartDocument()
```

الان تستطيع كتابة كل ما تريده في الملف باستخدام عشرات الطرق المتشابهة والتي تكون صيغتها WriteStartxxx() و WriteEndxxx()، الطريقة الاولى تبدأ الكتابة في المستوى الفرعي، والاخرى تقوم باغلاق المستوى حتى تتم الكتابة إلى نفس المستوى الحالي، المثال التالي يكتب مجموعة من العناصر XML Elemnts:

```
xmlData.WriteStartElement("Table")

    xmlData.WriteStartElement("Name")
    xmlData.WriteString("تركي العسيري")
    xmlData.WriteEndElement()

    xmlData.WriteStartElement("Name")
    xmlData.WriteString("عبدالله ابراهيم")
    xmlData.WriteEndElement()
xmlData.WriteEndElement()

xmlData.Close()
```

مخرجات الشيفرة السابقة ستكون:

```
<?xml version="1.0" encoding="utf-8" ?>
    <Table>
        <Name>تركي العسيري</Name>
        <Name>عبدالله ابراهيم</Name>
    </Table>
```

راجع مكتبة MSDN للحصول على تفاصيل كافة الطرق الأخرى.

# تكامل ADO.NET و XML

في هذا القسم سأريك كيف يمكنك الاستفادة من دعم ADO.NET للغة وصف البيانات XML، وعرض كيف تقرأ من مصادر البيانات لتخرجها على نسق XML، وفي المقابل ايضا كيف ترسل البيانات بنسق XML إلى مصادر البيانات.

PDF

## كتابة البيانات بصيغة XML

كل ما هو مطلوب منك لكتابة البيانات من مصادر البيانات بصيغة XML، استدعاء الطريقة WriteXml() والتابعة لكائن البيانات DataSet:

```
Dim cn As New OleDbConnection(connString)
cn.Open()

Dim daEmp As New OleDbDataAdapter("SELECT * FROM [Employees]")
Dim daDep As New OleDbDataAdapter("SELECT * FROM [Departments]")
Dim ds As New DataSet()

daEmp.Fill(ds, "Employees")
daDep.Fill(ds, "Departments")
cn.Close()

ds.Relations.Add("1علاقة", ds.Tables("Departments").Columns("ID"), _
    ds.Tables("Employees").Columns("DepartmentID"))

ds.WriteXml("C:\test.xml")
```

الطريقة السابقة ستحول جميع الجداول، السجلات، والحقول التابعة لكائن البيانات DataSet إلى ملف بصيغة xml والذي قد تكون مخرجاته شيئا مثل:

```
<?xml version="1.0" encoding="utf-8" ?>
    <NewDataSet>
        <Employees>
            <Name>خالد ابراهيم</Name>
            <Age>99</Age>
            <DepartmentID>1</DepartmentID>
        </Employees>
        <Employees>
            <Name>عمر عبدالله</Name>
            <Age>88</Age>
            <DepartmentID>2</DepartmentID>
        </Employees>
        <Employees>
            <Name>احمد محمد</Name>
            <Age>77</Age>
            <DepartmentID>2</DepartmentID>
        </Employees>
        …
        …
        <Departments>
            <ID>1</ID>
            <Name>قسم المبيعات</Name>
        </Departments>
        <Departments>
            <ID>2</ID>
```

PDF

```
            <Name>قسم التسويق</Name>
        </Departments>
        …
        …
    </NewDataSet>
```

ان امعنت النظر في شيفرة XML السابقة، سيتضح لك انه يوجد جدولين Employees و Departments، وتم عرض مخرجات سجلاتهم بشكل مستقل رقم وجود علاقة تربط بينهما، لذلك قد تفضل عرض السجلات بشكل متداخل Nested حتى تتداخل السجلات التابعة مع المتبوعة، لعمل ذلك عد إلى كائن البيانات DataSet واسند القيمة True للخاصية Nested والتابعة لكائن العلاقة:

```
ds.Relations("علاقة").Nested = True
ds.WriteXml("C:\test.xml")
```

العناصر ستكون متداخلة بهذا الشكل:

```
<?xml version="1.0" encoding="utf-8" ?>
    <NewDataSet>
        …
        <Departments>
            <ID>1</ID>
            <Name>قسم المبيعات</Name>
            <Employees>
                <Name>خالد ابراهيم</Name>
                <Age>99</Age>
                <DepartmentID>1</DepartmentID>
            </Employees>
            …
            …
        </Departments>

        <Departments>
            <ID>2</ID>
            <Name>قسم التسويق</Name>
            <Employees>
                <Name>عمر عبدالله</Name>
                <Age>88</Age>
                <DepartmentID>2</DepartmentID>
            </Employees>
            <Employees>
                <Name>احمد محمد</Name>
                <Age>77</Age>
                <DepartmentID>2</DepartmentID>
            </Employees>
            …
            …
```

PDF Factory Pro trial version www.fastio.com

```
        </Departments>
        …
        …
        …
    </NewDataSet>
```

## قراءة البيانات بصيغة XML

وكما هو الحال مع كتابة البيانات بالطريقة ()WriteXml يمكنك قراءة البيانات من أي ملف XML وإرساله إلى كائن بيانات DataSet باستدعاء الطريقة ReadXml فقط:

```
Dim ds As New DataSet
ds.ReadXml ("C:\test.xml")
```

كائن البيانات DataSet السابق يحمل كل السجلات الموجودة في الملف test.xml، قد تحتاج إلى اعادة تسمية الحقول والجداول ان لم تتوافق مع مصدر البيانات الذي تنوي ارساله اليه عن طريق كائن المحول DataAdapter.

بهذا نكون قد وصلنا إلى نهاية الجزء الرابع **برمجة قواعد البيانات**، وتعلمنا أساسيات استخدام فئات ADO.NET للوضعين المتصل والمنفصل، يتبقى الأمر عليك ان اردت استخدام فئات ADO.NET بالشكل الامثل ومعرفة كافة الطرق والخصائص التي لم اتطرق لها من مكتبة MSDN. حاليا، يمكنك العودة إلى نماذج Windows Forms لاستخدام ADO.NET معها أو المضي قدما إلى **برمجة ويب** عنوان الجزء الخامس والأخير من هذا الكتاب.

PDF

الجزء الخامس

# برمجة ويب

PDF

# تطبيقات ASP.NET (1)

ان كنت احد اعضاء موقع **شبكة المطورون العرب**، ستلاحظ ان اسم الدخول الخاص بك سيظهر في يمين صفحات الموقع، وان كنت تعتقد اننا قمنا بتصميم صفحة خاصة لكل عضو، فاعتقادك ليس في محله، اذ كل ما فعلناه هو تصميم صفحة واحدة تختلف باختلاف اعدادات زائر الموقع.

في هذا الفصل ستكون بدايتك لتطوير تطبيقات ASP.NET، وسيتمحور هذا الفصل حول نماذج Web Forms وطريقة عملها واضافة الادوات عليها وكتابة الشيفرة بها، وسيكون مدخلك الابتدائي لتطوير تطبيقات ASP.NET، اما الفصل القادم فسنتحدث عن تطوير تطبيقات ASP.NET بشكل عام.

**ملاحظة**

يتطلب هذا الفصل معرفة مسبقة بأغلب مصطلحات الانترنت وتصميم المواقع الشائعة، كما يفترض إلمامك التام بلغة تنسيق البيانات HTML، ويفضل ان تكون لديك خبرة سابقة في برمجة ويب (وكأني منزل إعلان وظيفة شاغرة في احد الصحف!).

## الخادم IIS

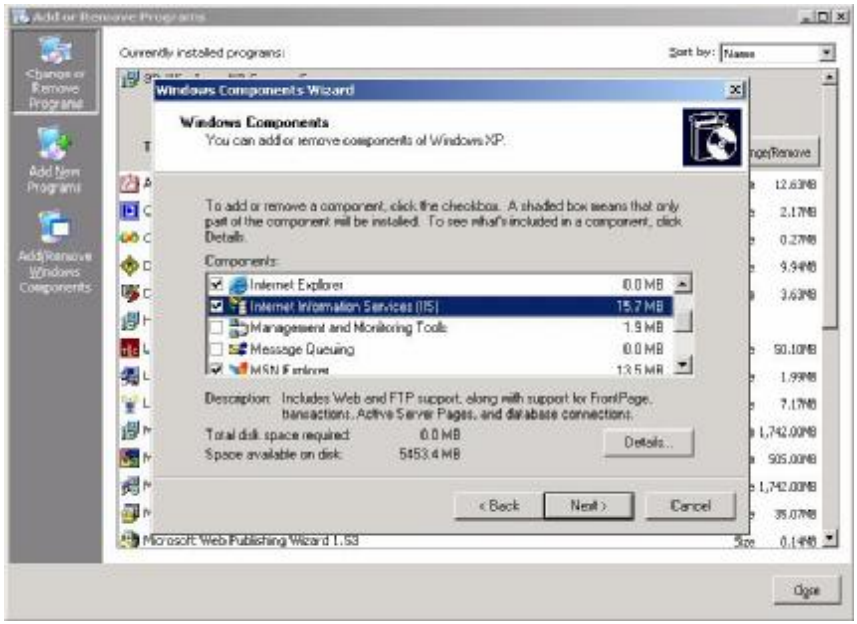
هذا الكتاب مختص ببرمجة Visual Basic .NET، والتحدث عن تطبيقات او برامج اخرى خارج نطاق الكتاب، الا ان هذا القسم موجه الى أولئك الأشخاص الذين ليس لديهم خلفية باستخدام الخادم IIS، ويوضح لهم –باختصار شديد- طريقة تركيبه وعمله، حيث انه يعتبر احد المتطلبات الأساسية لتشغيل تطبيقات ASP.NET على الجهاز.

## تركيب الخادم IIS

حتى تتمكن من تشغيل تطبيقات ASP.NET لابد ان يكون جهازك **خادم ويب Web Server**، وحتى تتمكن من تحويل جهازك الى خادم Web Server، عليك القيام بتركيب الخادم Internet Information Server من شركة Microsoft.

يمكنك تركيب انواع اخرى من الخوادم المنتشرة في الأسواق، ولكن على حسب علمي – حتى لحظة كتابة هذا الفصل - يعتبر الخادم IIS هو الخادم الوحيد الذي يمكنك من تشغيل صفحات ويب والمبنية على تقنية ASP.NET.

ان كنت تستخدم الإصدار Windows 2000 وما بعده، فيسرني إخبارك بان الخادم IIS موجود ضمن الاسطوانة الأصلية لنسخة نظام التشغيل Windows. وكل ما هو مطلوب منك لتركيب الخادم IIS، هو تشغيل الرمز Add or Remove Programs الموجود في لوحة التحكم Control Panel، ومن ثم الانتقال الى خانة التبويب Add/Remove Windows Components في يسار صندوق الحوار، واختيار العنصر Internet Information Services (IIS) (**شكل 1-20**).




**شكل 20-1**: تركيب الخادم IIS.

بعد تثبيتك للخادم IIS، قد لا يطلب منك برنامج الإعداد إعادة تشغيل الجهاز، مع ذلك من المستحسن دائما إعادة تشغيله، وبمجرد بدء اقلاع نظام التشغيل فقد تحول جهازك الشخصي الى خادم ويب Web Server يمكن له ان يستضيف مواقع في أقراصه الصلبه لو كنت تملك بنية اتصال قوية.

وحتى نتأكد من نجاح عملية تركيب الخادم، قم بتشغيل المتصفح Internet Explorer، واكتب هذا الرابط:

```
http://localhost
```

ان تم فتح الصفحة بنجاح، فهذا يعني ان عملية تركيب الخادم IIS قد تمت بنجاح وتحول جهازك الى خادم ويب Web Server. اما ان لم تظهر لك نتائج ايجابية، حاول مراجعة مكتب الدعم الفني لشركة Microsoft في منطقتك.

**ملاحظة**

قد لا تكون متصل بشبكة محلية Local Network او طلب هاتفي Dial-up Networking، لذلك يفضل تحديد الاختيار Bypass proxy server for local addresses والخاصة باعدادات البروكسي التابعة للمتصفح Internet Explorer.

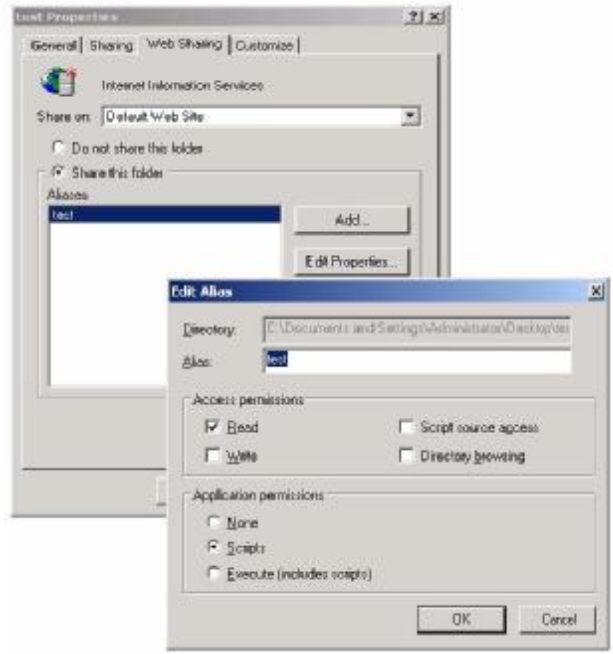
## الأدلة الوهمية

الموقع الذي تريد استضافته ما هو الى مجموعة مترابطة من الملفات، وكما تفعل مع ملفاتك الشخصية بترتيبها وتوزيعها على مجلدات، فعليك توزيع ملفات المواقع على مجلدات، ولكن السؤال الذي يطرح نفسه، كيف يمكن لزائر موقعك او انت من الوصول الى المجلد الذي وضعت الملفات فيه، والإجابة عن طريق كتابة المجلد في عنوان الرابط URL.

ولكن المشكلة تكون ان الزائر لا يعلم شيئا عن الموقع الفيزيائي للمجلد ومساره الكامل، وذلك لان الزائر لا يصل الى ملفات كما تصل انت، لذلك عليك إنشاء **دليل وهمي Virtual Directory** ما هو الا اسم وهمي مستعار يتم إيصالك الى المجلد الحقيقي.

حتى تقوم بعمل ذلك، أنشئ مجلد Folder كما تفعل دائما، وانقر بزر الفأرة الأيمن على المجلد ثم اختر الامر Properties من القائمة المنبثقة، انتقل الى خانة التبويب Web Sharing ثم

حدد الاختيار Share this folder، وان لم يظهر لك صندوق حوار جديد فاضغط علـــى الـــزر Add (**شكل 20-2**).




**شكل 20-2**: إنشاء دليل وهمي.

في خانة Alias اكتب الاسم الوهمي والمستعار الذي يمكن الزائر من الوصول الى المجلد بكتابته عند رابط URL، فلو كان الاسم الوهمي test يمكنك الوصول الى ملفات المجلـــد بكتابـــة شيئا مثل:

```
http://localhost/test
```

وحتى تتأكد من ذلك، أنشئ ملف test.html واكتب فيه ما تشاء، ثم انسخه في نفس المجلد الـــذي أنشأته للتو، واكتب العنوان التالي في نافذة المتصفح:

```
http://localhost/test/test.html
```

ستلاحظ ان صفحتك التي صممتها للتو قد ظهرت في المتصفح.

**ملاحظة**

ان تجاهلت كتابة اسم الملف، سيقوم الخادم IIS بالبحث عن مجموعة ملفات افتراضية كـ Default.html، Index.html، ...الخ.

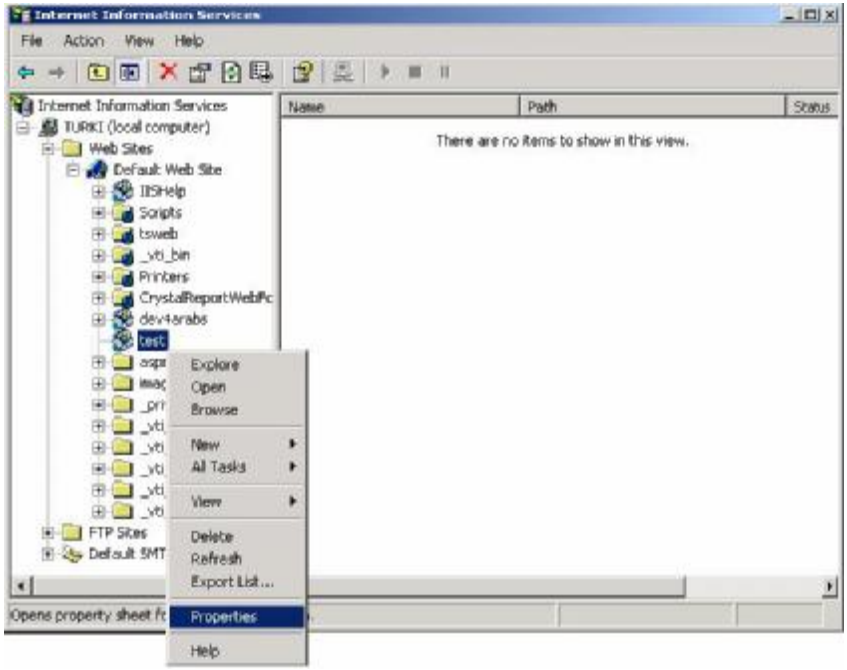
المزيد ايضا، ان كنت متصلا بشبكة الانترنت، فقد تسمح للزوار بدخول موقعك ان علموا برقم المعرف **IP Address**، ليصلوا إليك بكتابة شيئا مثل:

```
http://999.999.999.999/test/test.html
```

## الوصول إلى الأدلة الوهمية

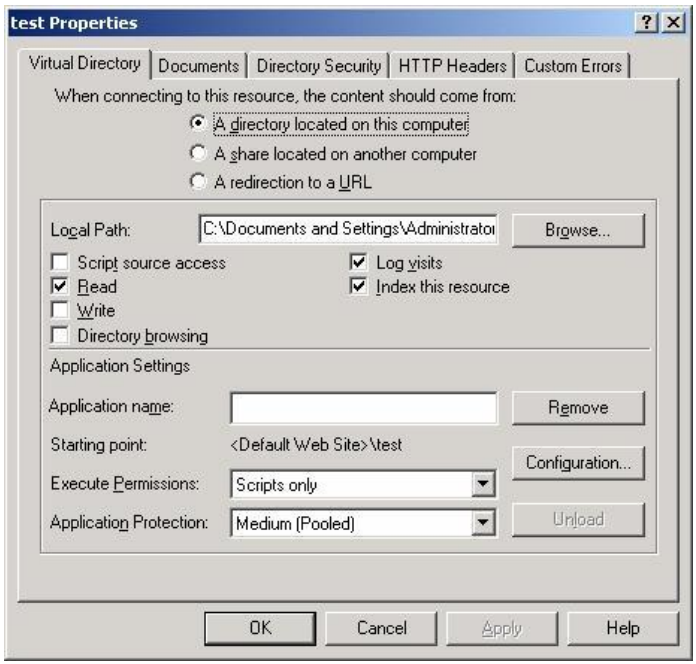
يمكنك الوصول الى كافة الأدلة الوهمية والمعرفة في الخادم IIS باختيار الرمز Internet Information Services من المجموعة Administrative Tools في لوحة التحكم Control Panel (**شكل 20-3**).

PDF Factory Pro trial version www.fineprint.com




**شكل 20-3**: الأدلة الوهمية في الخادم IIS.

المزيد ايضا، تستطيع اضافة المزيد من الأدلة الوهمية من نفس النافذة، كما يمكنك حذفها ايضا، وان اردت تعديل خصائص الأدلة الوهمية فاضغط بزر الفارة الأيمن على عنصر الدليل الوهمي في الشجرة اليسرى، واختر الأمر Properties ليظهر لك الخادم IIS عشرات الخيارات والأوامر التي يمكنك تعديلها (**شكل 20-4**)، يمكنك مراجعة ملفات التعليمات والخاصة بالخادم IIS لمزيد من التفاصيل.




**شكل 20-4**: تعديل خصائص الدليل الوهمي.

# مدخلك إلى نماذج Web Forms

عندما تنوى تطوير تطبيقات ASP.NET، فانك من البديهي لن تظهر نوافذ لنماذج Windows Forms، وانما ستعتمد على نماذج أخرى شبيهة بها الى حد كبير - تسمى **نماذج Web Forms** تمثل الصفحات التي تود عرضها على المستخدم في المتصفح Browser.

ان كان الغرض من نماذج Windows Forms هو عرض واجهة الاستخدام للمستخدم على شكل نوافذ، فان نماذج Web Forms هدفها هو تحويل المخرجات الى شيفرات HTML ليتم عرضها على المتصفح.

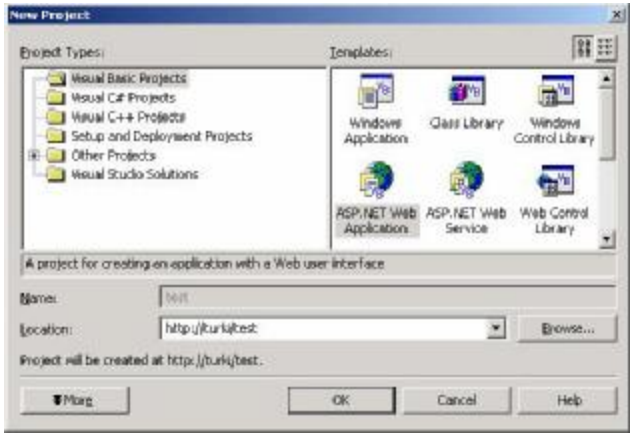
تتميز نماذج Web Forms عن نماذج Windows Forms بعدم ضرورة توفر نسخة من إطار عمل NET Framework. في جهاز الزائر، بل حتى لا يشترط وجود نظام تشغيل من نظم تشغيل Windows، ولكن يعيبها انها اقل إمكانيات ومميزات من نماذج Windows Form – بشكل مبدئي.

عندما يود المستخدم الوصول الى صفحة من صفحات موقعك، فسيقوم بكتابة اسم الموقع الكامل ويشمل ملف الصفحة والذي يكون بالامتداد aspx.، والذي يمثل صفحة نموذج Web form page، فلو كان تطبيقك سيعرض 10 نماذج Web forms، فستحتاج الى 10 ملفات aspx.* بشكل مبدئي، كما يمكنك الربط بين الملفات المتعددة -كأن تخصص ملف لرأس جميع صفحات موقع Header او أسفل جميع الصفحات Footer- كما سنرى لاحقا.

## إنشاء المشروع

يمكنك الاعتماد على المفكرة Notepad ان كنت تنوي تطوير تطبيقات ASP.NET، كما هو الحال ما سائر تطبيقات NET. الاخرى (Windows Applications، Console Application ...الخ)، الا انني لا اجد سببا مقنعا يمكنك من الاستفادة من بيئة Visual Studio NET. والتي توفر لك أدوات في قمة الروعة لتسهيل حياتك البرمجية.

اختر الامر New->Project من قائمة File، ليظهر لك صندوق حوار مشروع جديد New Project (**شكل 20-5**)، حدد القالب ASP.NET Web Application، ستلاحظ ان خانة اسم المشروع Name معتمة ولا يمكن تعديلها فهي تحمل اسم مسار المشروع، ستلاحظ ان خانة مسار الملفات Location تطلب منك رابط URL للموقع الذي تريد تحميل الملفات اليه (الدليل الوهمي Virtual Directory)، اضف مثلا http://localhost/test، كما يمكنك كتابة اسم مسار المجلد –عوضا عن موقعه- بالضغط على الزر Browse.

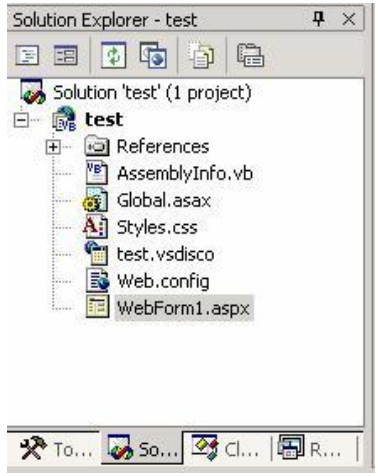



**شكل 20-5**: تحديد رابط الموقع المحلي الذي تود تحميل الملفات اليه.

**ملاحظة**

ان كنت قد وضعت حالة المتصفح Internet Explorer الى حالة العمل دون اتصال Work Offline، فلن يتم إنشاء ملفات المشروع بشكل صحيح وستظهر رسالة خطأ، لذلك عليك الغاء هذه الحالة بتشغيل المتصفح Internet Explorer نفسه، وإلغاء الاختيار Work Offline من قائمة File.

بعد ضغطك للزر OK، ستبدأ بيئة التطوير Visual Studio .NET بفتح مشروع جديد يحتوي على مجموعة من الملفات (**شكل 20-6**)، سنتعرض الى هذه الملفات بالتفصيل لاحقا، رغم ان بعض الملفات كـ AssemblyInfo.vb ليست غريبة عليك ان كنت قد طورت مشاريع Windows Application، وبالنسبة للملف Styles.css فهو ملف اختياري تكتب فيه أنماط Cascading Style Sheet، اما الملف WebForm1.aspx فهو ملف صفحة نموذج Web Form التي تصممها وتود عرضها على المستخدم في المتصفح.

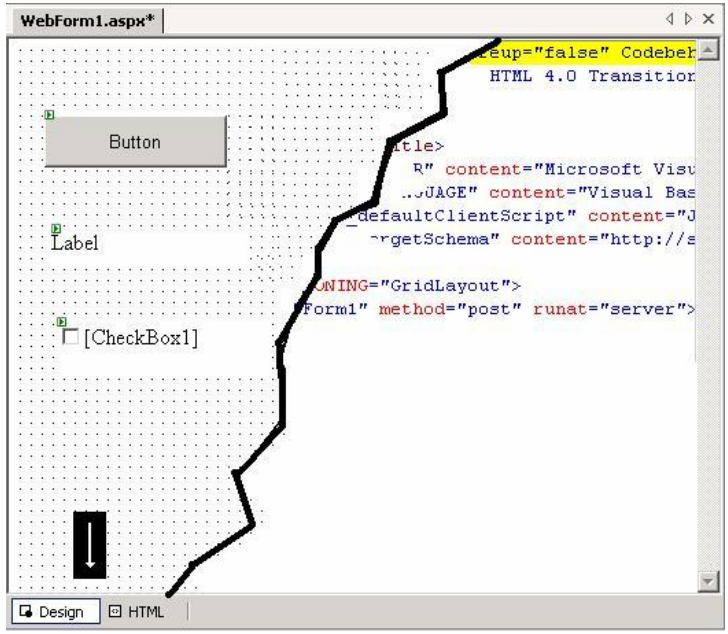



**شكل 20-6**: ملفات المشروع الابتدائية في نافذة مستكشف الحل Solution Explorer.

**ملاحظة**

انماط Cascading Style Sheet (CSS) غرضها تعريف انماط لتنسيقات ثابتة تستخدمها لعرض وسوم صفحات HTML. تمتلئ مواقع الانترنت بدروس ومقالات حولها، كما يمكنك العودة لمكتبة MSDN للحصول على مرجع كامل لجميع التنسيقات للانماط التي تعرفها.

وكما هو الحال ما اغلب برامج تصميم المواقع، يمكنك عرض صفحة النموذج Web Form في الوضع Design او الوضع HTML، في الوضع الاول يمكنك تصميم صفحة النموذج بشكل مرئي Visual كما يمكنك اضافة الادوات عليها وتعديل خصائصها، اما في وضع HTML فانك ستقوم بتصميم الصفحة بكتابة وسوم HTML يدويا بنفسك، يمكن التبديل بين كلا الوضيع عن طريق الضغط على الزر المناسب في الزاوية اليسرى السفلية من نافذة المصمم (**شكل 20-7**).




**شكل 20-7**: التحويل بين عرض التصميم Design او شيفرات HTML.

## ضبط الاعدادات الرئيسية

قبل البدء بوضع الأدوات على صفحات النماذج وكتابة الشيفرات، من المهم جدا ضبط اعـــدادات لثلاث خصائص هامة تؤثر في الشيفرة التي ولـــدها المـــصمم هـــي Page Layout، Target Schema، و Client Script Language، يمكنك الوصول لهذه الاعدادات في خانة التبويـــب Designer Defaults الموجودة في صندوق حوار خصائص المـــشروع Project Property Pages (**شكل 20-8**).

**شكل 20-8**: ضبط الاعدادات الرئيسية للمصمم.

في هذه الاعدادات تخبر المصمم الطريقة المثلى لتحويل تصاميمك الى وسوم HTML، في الخانة Page Layout تحدد فيها نسق وضع الادوات على صفحات النماذج وهو اما يكون Flow او Grid، في الحالة الاولى يتم وضع الادوات على صفحات النمـــاذج بحيـــث توافـــق اســـلوب HTML لعرضها، وهو الحال كما تفعل مع برامج معالجة النصوص، وضع في عين الاعتبار ان اي تحجيم في نافذة المتصفح سيؤدي الى زحزحة الادوات وتغيير مواقعها ما لم تضبط الخصائص المناسبة لحجمها، اما في حالة Grid فسيتم وضع الادوات كما تفعـــل مـــع نمـــاذج Windows Forms وستعرض الادوات على المتصفح تماما مثل تصميمها بمصمم صفحات النمـــاذج Web Forms.

بالنسبة للاختيار Target Schema ففيه تحدد نوع المتصفح الذي ستعرض صفحات موقعك عليه، الميزة في اختيارك اصدارات قديمة لـ Internet Explorer 3.03 و Netscape Navigator 3 سيؤدي الى توافق وتطابق حقيقي مع اغلب المتصفحات المستخدم، الا ان العيب فيها هو عدم الاستفادة من الإمكانية المتقدمة التي يوفرها الإصدار الأحدث Internet Explorer 5 وما بعده.

بعض شيفراتك البرمجية والتي يفترض ان تعمل في جهاز العميل Client (اي زائر الصفحة)، سيتم تحويلها الى لغة JScript والمتوافقة مع جميع المتصفحات، اما ان كنت تتوقع عرض موقعك في متصفحات Internet Explorer فقط (كان يعرض في شبكة محلية)، فيمكن تغيير الاختيار Client Script Language الى VBScript.

## كتابة الشيفرات

التعامل مع صفحة النموذج Web Form شبيه الى حد كبير مع التعامل مع نافذة النموذج Windows Form، وحتى نرى الامر واقعا اضف اداة Label في اعلى صفحة النموذج، واسفلها زر Button يليه اداة TextBox، وانقر نقرا مزدوجا على الزر لتتمكن من فتح نافذة الشيفرة واكتب الامر السطر التالي في حدثه Click:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    Label1.Text = "مرحبا بك يا" & TextBox1.Text
End Sub
```

مبروك! لقد قمت بالانتهاء من تصميم اول تطبيق لك مبني على تقنية ASP.NET، ان لم تصدق كلامي جرب الضغط على المفتاح [F5] لتنفيذ البرنامج، سيتم فتح المتصفح Internet Explorer الى الرابط http://locahost/xxx/yyy.aspx (حيث تمثل الحروف xxx الدليل الوهمي لمشروعك، والحروف yyy اسم ملف صفحة النموذج Web Page)، ويعرض لك صفحة النموذج التي صممتها للتو، جرب كتابة اسمك ثم اضغط على الزر Button، ستلاحظ ان عملية طلب Request جديدة تمت للصفحة وعادت بنفس الصفحة السابقة ولكن بعد تنفيذ الشيفرة السابقة لتظهر الاسم في الاداة Label.

PDF

**شكل 20-9**: بعد الضغط على الزر Button فتحت صفحة جديدة ظاهرة النص في الاداة Label.

PDF

**نقطة هامة جدا جدا جدا: الشيفرة البرمجية تم تنفيذها على الخادم Server وليس العميل Client!**

**الشائب رقم Q315158 عند تنفيذ:**

اعلنت شركة Microsoft قبل فترة عن ظهور شائب (تجده في المقال رقم Q315158 لموقع MSDN) ظهر في معظم اجهزة المستخدمين، وبالتحديد الذي يودون تجربة تنفيذ تطبيقات ASP.NET على أجهزتهم الشخصية تحت الخادم IIS، وذلك تظهر لهم رسالة (**شكل 20-10**) عند تنفيذ شيفرة ASP.NET التي أنجزوها.

**شكل 20-10**: شائب عند تنفيذ صفحة ASP.NET.

ان كنت تود معرفة تفاصيل سبب حدوث هذا الخطأ، فيمكنك توجيه متصفحك الى الرابط http://support.microsoft.com/default.aspx?scid=kb;en-us;Q315158 حيث اني لن اعرض لك هنا الا حل من الحلول الثلاثة التي يعرضها الرابط السابق.

افتح ملف الاعدادات Machine.config وابحث عن العنصر processModel، وستجد بين ثنايا مواصفاته المواصفة userName Attribute:

```
<processModel enable="true"
…
…
…
userName="machine" password="AutoGenerate" logLevel="Errors"
…
…
…
/>
```

غير قيمة هذه المواصفة من machine الى SYSTEM:

```
userName="SYSTEM" password="AutoGenerate" logLevel="Errors"
```

احفظ التعديلات، واعد تنفيذ مشروعك لتستمع ببرمجة ASP.NET.

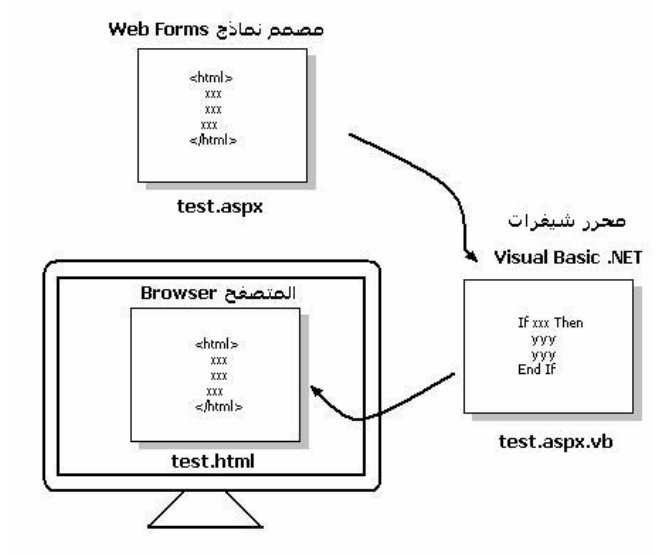
**انظر ايضا**

لمزيد من التفاصيل حول ملفات الاعدادات Configuration Files، راجع الفصل الحادي عشر **المجمعات** Assemblies.

## تحليل الشيفرة

تقتضي فلسفة تطبيقات ASP.NET والمبنية على نماذج Web Forms فصل الشيفرات البرمجية عن الشيفرات المخصصة لعرض واجهة الاستخدام، وحتى تفهم ماذا وكيف ولماذا يحدث ذاك وذاك، عليك معرفة ان كلمة الشيفرة –التي اقصدها في هذا السياق- مقسمة إلى ثلاثة أقسام (**شكل 20-11**).

PDF Factory Pro trial version www.fineprint.com




**شكل 20-11**: الأقسام الثلاث للشيفرات.

اول شيفرة تم توليدها هي شيفرة HTML من مصمم نماذج Web Forms، بحيث تحتوي على بيانات النموذج والادوات المحضونة فيه، والشيفرة التي تليها هي شيفرتك المكتوبة بلغة Visual Basic .NET والتي تقوم بمعالجة الطلبات Requests من الزوار واجراء اللازم، لتنتج الشيفرة النهائية HTML والتي يتم عرضها على المتصفح Browser، واليك تفاصيلها:

**شيفرة HTML مولدة من مصمم النماذج:**

بعد وضع الادوات على صفحة النموذج Web Form، سيقوم مصمم النماذج بتوليد الشيفرة التالية:

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb" Inherits="test.WebForm1"%>
<HTML dir="rtl">
    <HEAD>
        <title>WebForm1</title>
        …
        …
        </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
```

```
            <asp:Label id="Label1" runat="server"></asp:Label>
            <asp:Button id="Button1" runat="server"></asp:Button>
            <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
        </form>
    </body>
</HTML>
```

ضع في عين الاعتبار، ان الشيفرة السابقة شبيهة الى حد كبير بشيفرة HTML ولكن لا يوجد متصفح يستطيع فهمها، وعندما تمعن النظر بين وسومها ستلاحظ المواصفتين ID و runat، الاولى تمثل اسم الاداة التي وضعتها، والثانية تحدد فيها اين يتم معالجة شيفرة الاداة، والقيمة "server" المسندة لها تؤكد لنا على **ان جميع شيفرات وأوامر الأدوات يتم تنفيذها في الخادم Server وليس جهاز العملي Client**.

الشيفرة السابق ستحفظ في نفس ملف مصمم النماذج Web Form بالامتداد aspx.

**شيفرة Visual Basic .NET**:

عندما يقوم الزائر بطلب الصفحة وكتابة اسمها في المتصفح، سيتم تنفيذ الشيفرة التي كتبناها بلغة Visual Basic .NET:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    Label1.Text = "مرحبا بك يا " & TextBox1.Text
End Sub
```

والتي تقوم بمعالجة البيانات واسناد قيمة النص الموجود في الاداة TextBox إلى الأداة Label، الشيفرة السابقة -كما قلت للمرة الثالثة- سيتم تنفيذها على جهاز الخادم لمعالجة الطلبات من العملاء Clients، فلا تعرض فيها نوافذ نماذج Windows Forms او صناديق حوار Message Box او اي مخرجات على الشاشة، والسبب ان المخرجات يفترض ان تكون بصيغة HTML لترسل الى جهاز العميل Client وتعرض في داخل المتصفح Browser.

هذه الشيفرة ستحفظ في ملف يحمل نفس اسم ملف صفحة النموذج السابقة ولكن بالامتداد aspx.vb.، فلو كان اسم ملف النموذج test.aspx فسيكون اسم هذا الملف test.aspx.vb .

PDF Factory Pro trial version www.fineprint.com

**شيفرة HTML النهائية:**

اخيرا، بعد قيام الخادم بمعالجة الطلب Request، سيتم توليد شيفرة HTML النهائية والتي يمكن للمتصفح Browser من استيعابها وعرضها بتنسيقات مختلفة أمام انف الزائر:

```
<HTML dir="rtl">
    <HEAD>
        <title>WebForm1</title>
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form name="Form1" method="post" action="WebForm1.aspx"
id="Form1">
            …
            …
            …
            <span id="Label1" style="font-size:Large;font-
weight:bold;height:72px;width:335px;Z-INDEX: 101; LEFT: 92px;
POSITION: absolute; TOP: 48px">مرحبا بك يا عباس السريع</span>
            …
            …
        </form>
    </body>
</HTML>
```

## أساليب تنفيذ الصفحة

وضحت في الفقرة السابقة توزيع الشيفرات عند طلب الزائر لصفحة aspx من صفحات موقعك، وسأظهر لك هنا المزيد من التفاصيل حول الأساليب الثلاث التي يمكن معالجة صفحة ASP.NET بها، ولكن دعني اتفق معك اني في كل مرة اذكر فيها عبارة **موجه الصفحة @Page**، فإنني اقصد هذا السطر الموجود في اعلى ملف صفحة النموذج aspx:

```
<%@Page Language="vb" AutoEventWireup="false"
Codebehind="test.aspx.vb" Inherits="test.WebForm1"%>
```

**فئات Code-Behind Classes:**

عندما تقوم بتنفيذ برنامجك، ستقوم بيئة التطوير Visual Studio .NET بترجمة ملفات مشروعك المكتوبة بلغة Visual Basic .NET (والتي تنتهي بالامتداد aspx.vb.*) الى مكتبة بالامتداد DLL.، وبمجرد قيام الزائر بطلب صفحة نموذج aspx، سيقوم محرك ASP.NET بقراءة القيمة Inherits في موجه الصفحة @Page، القيمة تمثل الاسم الكامل لفئة الصفحة التي كتبت بها شيفرات الأدوات –كما فعلنا سابقا مع الاداة Button، وتنفيذ الإجراء المناسب في فئة الصفحة.

ملفاتك المكتوبة بلغة Visual Basic .NET والتي تم ترجمتها تسمى **الشيفرة الخلفية Code-Behind**.

**الترجمة عند الطلب On-Demand Compilation**:

لست مضطرا لترجمة ملفات الشيفرة الخلفية في مكتبة DLL، اذ يمكنك استخدام المواصفة src في موجه الصفحة @Page وكتابة اسم الملف المصدري للشيفرة الخلفية:

```
<%@ Page Language="vb" src="WebForm1.aspx.vb"  _
Inherits="test.WebForm1"%>
```

عندما يقوم الزائر بطلب صفحة aspx، سيتم البحث عن الملف الموجود في المواصفة src ويتم ترجمته الى ملف DLL عند الطلب، يعيب هذا الأسلوب البطء الشديد عند اول استدعاء للصفحة، كما ان شيفراتك المصدرية ستكون قابلة للتعديل فهي مازالت محفوظة في ملفاتها النصية *.aspx.vb .

**الأسلوب الكلاسيكي Classic ASP Style**:

الأسلوب الكلاسيكي يتبع أسلوب استخدام تقنية ASP القديمة في كتابة الصفحات، يعيب هذا الأسلوب مثل ما يعيب تقنية ASP القديمة حيث تدمج شيفرات واجهة الاستخدام، بنفس شيفرات لغة Visual Basic .NET، مما يعقد تركيبة ملفات aspx –خاصة ان كبر حجمها. على اية حال، يمكنك اتباع الاسلوب الكلاسيكي بوضع الشيفرة التي تنوي تنفيذها على الخادم داخل التركيب <SCRIPT RUNAT="server"> ... </SCRIPT>:

```
<%@ Page Language="vb"%>
…
…
…

<SCRIPT RUNAT="server">
    Private Sub Button1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button1.Click

        Label1.Text = "مرحبا بك يا " & TextBox1.Text
    End Sub
</SCRIPT>
```

PDF Factory Pro trial version www.fastio.com

## الخلاصة

نستنتج من كل عرضناه في الفقرات السابقة، ان فلسفة ASP.NET تقتضي ربط الادوات التي تعرض عند المستخدم في صفحة مستعرضه، بشيفرات برمجية تكتب بين فكي احداث هذه الادوات. ان لم تبرمج سابقا بأي لغة برمجة ويب لجهة الخادم Server side script (كـ ASP، PHP، Perl، CGI، ...الخ)، فعليك تغيير منطقك البرمجي ليميل الى جانب الخادم وليس العميل، وان لم تضع ذلك في عين الاعتبار دائما، ستواجه الكثير من المتاعب.

# الفئة Page

ان كانت الفئة System.Windows.Forms.Form تمثل نافذة نموذج Windows Form، فان الفئة System.Web.UI.Page تمثل صفحة نموذج Web Form:

```
Public Class WebForm1
    Inherits System.Web.UI.Page
    …
    …
    …
End Class
```

مع ذلك الادوات وصفحات النماذج ليست كالأدوات ونوافذ النماذج التابعة لـ Windows Forms، حيث تحتوي الثانية على عدد اكثر بكثير من الخصائص، الطرق، والاحداث –كما ستكتشفه بنفسك.

في هذا القسم سأتحدث عن الفئة Page وسأحاول التعرض الى مجموعة من خصائصها، طرقها، واحداثها. كما سأذكر مجموعة اضافية من الوسوم التي تستخدمها بالاضافة الى موجه الصفحة @Page.

## خصائص صفحة النموذج

عندما تنوي انشاء صفحة نموذج عربية، فأول خاصية عليك تغيير قيمتها الخاصية dir والتي تسند لها القيمة rtl لتحويل اتجاه الصفحة من اليمين الى اليسار، والخاصية Culture التي تحدد الاعدادات الإقليمية Regional Settings لصفحة النموذج.

عند حدوث خطأ ووقوع استثناء Exception اثناء تنفيذ شيفرة الصفحة ولم تتفاداه، سيقوم محرك ASP.NET بعرض صفحة خاصة به توضح الاستثناء ومصدره، مع ذلك قد تعين صفحة خطاء خاصة بموقعك وذلك باسناد رابط الصفحة الى الخاصية ErrorPage:

```
me.ErrorPage = "http://www.dev4arabs.com/error.aspx"
```

تفيدك الخاصية IsPostBack لمعرفة ما اذا كانت هذه اول مرة تعرض فيها الصفحة، او تم عرضها بعد ارسال البيانات لها –اي قام الزائر بالضغط على زر Submit، تعود هذه الخاصية بالقيمة True ان تم فعلا ارسال البيانات الى هذه الصفحة.

بالنسبة للخاصية Controls فه تمثل مجموعة للادوات المحضونة في صفحة النموذج، يمكنك تطبيق الحلقة For Each … Next عليها بكفاءة –كما فعلنا سابقا مع نوافذ النماذج:

```
Dim ctl As Control
For Each ctl In Me.Controls
    …
    …
Next
```

معظم الخصائص الاخرى (على وجه الخصوص Request، Response، Server، Session، Application) تعود بكائنات من النوع HttpRequest، HttpResponse، HttpServerUtility، HttpSessionState، HttpApplicationState سأتحدث عنها في الفصل القادم **تطبيقات ASP.NET (2)** بمشيئة الله.

**الخاصية EnableViewState والخاصية ViewState:**

عندما يقوم الزائر بزيارة الصفحة، فان القيم التي تحملها الادوات ستكون هي القيم الافتراضية – والتي عرفتها لحظة تصميم صفحة النموذج، مع ذلك قد تود بالاحتفاظ بالتعديلات التي قام بها المستخدم، تخيل مثلا انك قمت بوضع مجموعة من الادوات TextBox تطلب من المستخدم ادخال معلومات، وافترض ان المستخدم قام بتعبئتها جميعا وضغط على الزر Submit، وتخيل ان احد المعلومات التي ادخلها خاطئة، لذلك يتوجب عليك إظهار ادوات TextBox له من جديد حتى يصحح تعديلاته.

PDF

المشكلة تظهر بعد هذه الفاصلة، عندما يتم عرض الادوات للمرة الثانية فان القيم التي ادخلها المستخدم في الزيارة الاولى للصفحة (قبل الضغط على زر Submit) سيتم إلغائها وكان شيئا لم يحدث، مما يحد المستخدم الى اعادة كتابة جميع البيانات من جديد.

من هنا يأتي دور الخاصية EnableViewState والتي تسند لها القيمة True لتمكنك من حفظ القيم الحالية للادوات لحظة ارسالها للمرة الثانية الى نفس الصفحة:

```
Me.EnableViewState = True
```

ضع في عين الاعتبار، ان جميع الادوات سيتم حفظ قيمها ان كانت قيمة الخاصية EnableViewState لكل اداة هي True، وهذا بحد ذاته سيكلفك الكثير من البيانات المرسلة في كل مرة للصفحة، لذلك لا تستخدم هذه الخاصية **الا ان كنت فعلا بحاجة لها**، واسند القيمة False دائما للادوات الغير مرغوبة حفظ قيمها –خاصة التي تحتوي على قيم كثيرة:

```
Listbox1.EnableViewState = False
```

بمجرد اسنادك للقيمة True للخاصية EnableViewState سيتم حفظ قيم الادوات، ولكن ما هي علاقة الخاصية ViewState المسطورة كعنوان لهذه الفقرة؟ والاجابة ببساطة هو إمكانية تعريفك لمتغيرات او قيم –ان صح التعبير- يمكنك الاحتفاظ بها حتى تستفيد منها للاستدعاء الثاني للصفحة، الشيفرة التالية مثلا تعرض لك كم مرة تم استدعاء الصفحة:

```
If Me.ViewState("Counter") Is Nothing Then
    Me.ViewState("Counter") = CInt(1)
Else
    Me.ViewState("Counter") = CInt(Me.ViewState("Counter") + 1)
End If
Label1.Text = CStr(Me.ViewState("Counter"))
```

**ملاحظة**

الخاصية ViewState هي كائن من النوع Dictionary –وهو شبيه الى حد ما بالمجموعات Collection، وقد تمكنا من تعبئة عناصره دون استخدام الطرق التقليدية للمجموعات Add()، Insert()، ...الخ. لم اتحدث في هذا الكتاب عن الكائن Dictionary، لذلك أنصحك بمكتبة MSDN ان اردت المزيد من التفاصيل حول هذا الكائن.

**الخاصية SmartNavigation**:

عندما يقوم الزائر بالضغط على الزر Submit، ستعود نفس الصفحة اليه في المستعرض، وقد يلاحظ المستخدم التغيير في نزول النسخة الجديدة من الصفحة مما يسبب التشويش عليه، كما ان موقع اشرطة التمرير ScrollBars للمتصفح ستبدأ من اعلى الصفحة عند انزال النسخة الجديد، مع ذلك يمكنك اسناد القيمة True للخاصية SmartNavigation لمنع حدوث كل ذلك.

**ملاحظة**

مفعول الخاصية SmartNavigation سيظهر ان كان المتصفح الإصدار الخامس من InternetExplorer وما بعده.

## طرق صفحة النموذج

من طرق صفحة النموذج الطريقة MapPath() والتي تحول مسار الدليل الوهمي Virtual Directory الى المسار الفيزيائي (الحقيقي) له:

```
Dim F As New System.IO.StreamReader(Me.MapPath("/test/file.txt"))
```

كما توجد الطريقة –أشبه بخاصية- هي HasControls() والتي تعود بالقيمة True ان وجدت ادوات على صفحة النموذج.

## أحداث صفحة النموذج

عندما يتم طلب صفحة نموذج، فاول حدث تقوم بتنفيذه هو Init، وهو حدث ابتدائي يمثل عملية استدعاء للصفحة ولا يمكنك قراءة قيم الادوات التي كتبها المستخدم ولا حتى معرفة ما اذا كانت هذه المرة الاولى لعرض الصفحة، وذلك لان الحدث يتم تنفيذه قبل الربط الفعلي مع ادوات النموذج، فلا تعتمد على هذا الحدث كثيرا.

بعد ان يتم ربط الادوات بأحداثها وتحميل بياناتها، يمكنك كتابة الشيفرات في الحدث Load والذي يمكنك من قراءة قيم الادوات كما تستطيع الاستعلام عن الخاصية IsPostBack. وبمجرد تنفيذ جميع الشيفرات في هذا الحدث، سيتم تنفيذ احداث باقي الادوات في النموذج.

الحدث الاخير الذي سيتم تنفيذه هو الحدث Unload والذي تكتب فيه جميع الشيفرات الضرورية لتفريغ المصادر كاغلاق اتصالات قواعد البيانات مثلا.

PDF

الغرض الوحيد للخاصية ErrorPage هو فقط عرض صفحة معينة ان حدث استثناء لم يتم تفاديه، اما إن اردت إجراء عمليات اخرى فيمكنك كتابة شيفرات هذه العمليات في الحدث Error.

## وسوم إضافية

بعض الخصائص التي عرضتها عليك سابقا تكتب في داخل موجه الصفحة Page@، جرب مثلا تغيير قيمة الخاصية SmartNavigation لتجد القيمة المسندة في موجه الصفحة Page@ مكتوب:

```
<%@ Page Language="vb" … … … smartNavigation="True"%>
```

يمكنك كتابة وسم صفحة Page@ واحد فقط في كل صفحة ملف aspx، كما يفضل وضعه في اعلى الصفحة حتى تتمكن من الوصول اليه بشكل سريع.

من الوسوم الاضافية الوسم Import@ الذي يحاكي الكلمة المحجوزة Imports (في لغة Visual Basic .NET) لاستيراد مجال اسماء في نفس صفحة aspx:

```
<%@ Import namespace=" System.Data.OleDb"%>
```

والوسم Implements الذي يحاكي الكلمة المحجوزة Implements (في لغة Visual Basic .NET) والذي يضمن واجهة Interface في صفحة النموذج:

```
<%@ Implements Interface="System.IDisposable"%>
```

**ملاحظة**

إن أصبت بوسواس من الوسمين السابقين واعتقد انك ستستخدمهما دائما، فدعني اذكرك هنا بانهما خاصان بصفحة النموذج aspx فقط، اما صفحة الشيفرة الخلفية aspx.vb فهي لا زالت بلغة Visual Basic .NET التي تعرفنا عليها منذ الفصل الاول لهذا الكتاب **تعرف على** Visual Basic .NET!

PDF

اخيرا، لديك الوسم Reference الذي يمكنك من ربط صفحة نموذج اخرى في نفس الصفحة، للاستفادة من كائناتها (مع العلم انها لا تقوم بإظهارها وانما ربطها كمرجع كما تفعل مع صندوق حوار المراجع Reference):

```
<%@ Reference page="WebForm2.aspx" %>
```

# الأدوات

كما ذكرت في بداية القسم السابق، التعامل مع صفحات نماذج وادوات Web Forms شبيه الى حد كبير التعامل مع نماذج وادوات Windows Forms، ولكن عشرات الخصائص، الطرق، والاحداث ليست مدعومة في ادوات Web Forms، لدرجة ان بعض الادوات لا تحتوي على اية احداث!، والسبب –كما ذكرت- يتعلق بمكان تنفيذ الشيفرات المصدرية، فكيف تريد تنفيذ الحدث MouseMove على اداة Web Form ان كانت شيفراتها لا تنفذ الى في الخادم (لحظة ارسال البيانات الى الصفحة). ولو كان الامر كذلك، لتم تنفيذ الصفحة بالخادم والاتصال به في كل مرة تحرك الفأرة فوق الاداة من داخل متصفحك!

## أدوات Web Forms Controls

هذه المجموعة من الادوات تجدها في خانة التبويب Web Forms من صندوق الادوات ToolBox، ودعني اريك هنا بعض التغييرات التي تميزها عن ادوات Windows Forms.

اول تغيير قد لاحظته ان خاصية الاسم هي (ID) وليس (Name)، وذلك يتعلق بصيغة لغة HTML لبناء المعرفات في وسومها، كما توجد خصائص اضافية كالخاصية BorderStyle التي تحدد فيها نقش الحد الخارجي للاداة، والخاصية BorderWidth لتذكر فيها عرض الحد.

بالنسبة للخاصية cssClass ففي تذكر اسم فئة النمط Class Style والخاص بالانماط CSS التي تعرفها بنفسك. وضع في عين اعتبارك ان خاصية حجم الخط التابعة لكائن الخط Font لا تسند لها قيم عددية، اذ عليك استخدام الاحجام المعرفة (كـ Small، Big، Medium ...الخ)، اما ان اردت استخدام قيم عددية، فلا تنسى اضافة الحرفين **pt** بعد الرقم (10pt، 20pt، ...الخ).

معظم الخصائص قد تم اختصارها لبعض الادوات، فنجد مثلا الاداة CheckBox والاداة RadioButton تحتوي على الخاصية Checked والتي تعود بالقيمة True ان تم اختيار العنصر، وعلى ذكر الاداة RadioButton فدعني أنبهك هنا بامكانية تحديد اكثر من اداة

RadioButton في نفس الاداة الحاضنة! وحتى تمنع الزائر من عمل ذلك، اربط ادوات الـ RadioButton في مجموعات عن طريق كتابة اسم المجموعة في خاصيتها GroupName.

ان كنت على دراية تامة بلغة تنسيق البيانات HTML، فالتعامل مع الادوات سيكون سهلا عليك، فكل ادا تراها لها مقابل في لغة HTML، كاداة الجداول Table تمثل الوسم <table>، اداة Label تمثل الوسم ، اداة الصور Image تمثل الوسم <img>، اداة الرابط HyperLink تمثل الوسم <a>، اداة Panel تمثل الوسم <Panel>، اداة الـ ... علي تغيير عنوان الكتاب الى HTML ان اردت مني اكمال وشرح هذه الادوات.

## أدوات HTML Forms Controls

الادوات السابقة تستخدمها ان اردت برمجتها وكتابة شيفرات مصدرية يتم تنفيذها في الخادم Server، اما ادوات HTML Forms Controls فالغرض منها اضافة ادوات في صفحة النموذج **بحيث تكتب الشيفرات التي تود تنفيذها عند العميل (اي في داخل المتصفح)**. ولن تستطيع استخدام اي لغة من لغات NET. لبرمجة هذه الادوات، وليس لك مخرج إلا الاعتماد على احد لغات ويب للصفحات الديناميكية كـ VBScript، J Script، او حتى Java Script.

يمكنك استخدام هذه الادوات من خانة التبويب HTML في صندوق الادوات Toolbox **(10-20)**.

PDF Factory Pro trial version www.fastio.com

**شكل 20-12**: ادوات HTML Forms تصل إليها من خانة التبويب HTML في صندوق الادوات.

## أدوات التحقق Validation

قد تلاحظ في خانة التبويب Web Forms من صندوق الأدوات Toolbox مجموعة من الادوات اسمها يحمل الصيغة xxxValidator، تستخدم هذه الادوات –في اغلب الاحوال- مع ادوات النص TextBox بحيث تمكنك من التحقق من البيانات قبل ارسالها الى الخادم، وعملية التحقق تتم بتوليد شيفرة مصدرية بلغة Jscript او VBScript بصفحة العميل في المتصفح.

من هذه الادوات، الاداة RequiredFieldValidator للتحقق من انه تم ادخال قيمة في الاداة (التي ستكون اداة نص في اقصى الاحوال)، الاداة RangeValidator التي تمكنك من تحديد مجال للقيم يمكن ان تحمله الاداة، الاداة CompareValidator التي تمكنك من التحقق ما اذا كانت قيمة الاداة اكبر من، اصغر من، او تساوي قيمة اداة اخرى في نفس الصفحة، والاداة CustomValidator لتخصيص عمليات التحقق Validation بنفسك برمجيا (يمكن ان تكون شيفرة التحقق في الخادم ايضا عن طريق هذه الاداة).

**ملاحظة**

جميع هذه الأدوات (باستثناء RequiredFieldValidator) تجري عملية التحقق ان لم تكن القيمة خالية Empty، وان كانت خالية فسيعتبر التحقق ناجح. وتذكر انه سيتم تحويلها الى شيفرات Script لتعمل في المتصفح، لذلك قد يكون الزائر قد ألغى عمل تنفيذ شيفرات الـ Scripts في متصفحه ولن تعمل هذه الأدوات بنجاح.

الأدوات السابق ذكرها مشتق من الفئة القاعدية BaseValidator، واستخدامها متشابه الى حد كبير، ففي الخاصية ControlToValidate تحدد فيها اسم الأداة التي تود التحقق منها، ومن ثم تسند قيمة حرفية الى الخاصية ErrorMessage حتى تظهر رسالة الخطأ في وجه الزائر. وبمجرد قيام الزائر بالضغط على زر Submit، ستجرى عملية التحقق (**شكل 20-13**).

PDF

**شكل 20-13**: استخدام الأداة RequiredFieldValidator.

PDF

كان غرضي من هذا الفصل تعريفك بأساسيات تطوير تطبيقات ASP.NET Applications، وكان جل تركيزي على نماذج Web Forms وأدواتها. الفصل التالي سيعرض عليك مزيد من التفاصيل التي يتطرق بعضها حول نماذج Web Forms، والبعض الآخر حول مشاريع ASP.NET بشكل عام.

PDF Converter

# تطبيقات ASP.NET (2)

عرفتك في الفصل السابق على أساسيات تصميم تطبيقات ASP.NET وإنشاء صفحات نماذج Web Forms حتى تستوعب الفكرة من عمل صفحات ASP.NET، ولكن تبقى لنا مجموعة كبيرة من المواضيع المتفرقة التي عليك معرفتها حتى تنجز تطبيقات ASP.NET متكاملة.

## كائنات صفحات ASP.NET الأساسية

الخصائص الخمس لصفحة النموذج Request، Response، Server، Application، و Session تعود بكائنات من النوع HttpRequest، HttpResponse، HttpServerUtility، HttpApplicationState، و HttpSessionState –على التوالي، في هذا القسم سنتحدث عن هذه الكائنات بشكل سريع، ولمزيد من التفاصيل فان مستندات NET Documentation. بانتظارك.

### الكائن HttpRequest

يمثل الكائن HttpRequest البيانات القادمة من متصفح العميل Client Browser، ويحتوي على مجموعة كبيرة من الطرق والخصائص –للقراءة فقط ReadOnly- تمكنك من الحصول على معلومات حول العميل والبيانات التي أرسلت للصفحة.

اول هذه المعلومات هي طريقة الحصول عليها ومعرفة كيف تم إرسالها عن طريق الخاصية HttpMethod، والتي تعود بقيمة حرفية –من النوع String- تكون اما GET او POST:

```
Label1.Text = Me.Request.HttpMethod ' GET او POST
```

بالنسبة لارسال البيانات بالاسلوب POST في اغلب الاحوال يتم بالضغط على الزر Submit في صفحة العميل، اما الاسلوب GET فتتم بكتابة الرابط URL مع إضافة القيم عليه:

```
http://www.dev4arabs.com/test.asp?id=1&site=vb
```

البيانات مرسلة في الرابط السابق بالاسلوب GET، وان اردت قراءة القيم السابقة استخدم الخاصية QueryString:

```
Label1.Text = "ID & = " & Me.Request.QueryString("id")
Label 1.Text = "Site & = " & Me.Request.QueryString("site")
```

كما يمكنك تحديد نوع عملية ارسال البيانات (اما POST او GET) عن طريق الخاصية RequestType:

```
Me.Request.RequestType = "POST"
```

المزيد ايضا، تستطيع معرفة حجم البيانات المرسلة من صفحة العميل الى الصفحة عن طريق الخاصية ContentLength:

```
Label1.Text = CInt(Me.Request.ContentLength)
```

اما عند رغبتك في الحصول على رقم معرف IP Address للزائر فاستعلم عنه في الخاصية UserHostAddress، وان كان العميل يحتوي على اسم DNS فيمكن استخدام الخاصية UserHostName:

```
Label1.Text = Me.Request.UserHostAddress
Label2.Text = Me.Request.UserHostName
```

وعند الحديث عن المتصفح Browser، فيمكنك استخدام الخاصية Browser والتي تمثل كائن من النوع HttpBrowserCapabilities التي تحتوي على 25 خاصية تعود بمعلومات حول المتصفح، كاسم المتصفح، رقم اصداره، السماح ودعمه للغة VBScript، نظام التشغيل ...الخ:

```
Label1.Text = Me.Request.Browser.Browser            ' IE
Label2.Text = Me.Request.Browser.MajorVersion       ' 6
Label3.Text = Me.Request.Browser.Platform           ' WinNt
Label4.Text = Me.Request.Browser.VBScript           ' True
```

PDF

اخيرا، ان كانت الصفحة ترسل لك ملف Upload، فيمكنك الوصول الى هذه الملفات والتحكم فيها عن طريق المجموعة Files، والتي تحتوي على كائنات من النوع HttpPosttedFile، الشيفرة التالية تقوم بحفظ جميع الملفات المرسلة الى الصفحة:

```
Dim uploadedFile As HttpPostedFile

For Each uploadedFile In Me.Request.Files
    uploadedFile.SaveAs(uploadedFile.FileName)
Next
```

## الكائن HttpResponse

الكائن HttpResponse يمثل البيانات المرسلة من الخادم الى صفحة العميل، يمكنك ارسال وسوم HTML ان اردت باستخدام الطريقة Write() لكتابة الوسوم كوسيطة لها، او استدعاء الطريقة WriteFile() ان كانت الوسوم محفوظة في ملف اخر:

```
Me.Response.Write ("<p><b>مرحبا بك</b></p>")
Me.Response.Write ("<p><b>مرحبا بك</b></p>")
Me.Response.WriteFile ("data.html")
```

من الطرق ايضا، الطريقة Redirect() التي توجه العميل الى صفحة اخرى، والطريقة End التي توقف عملية اكمال ارسال الصفحة:

```
Me.Response.Redirect("http://www.dev4arabs.com/accessDenied.aspx")
Me.Response.End()
```

عند التعامل مع الـ Cookies، فعليك إنشاء كائن من النوع HttpCookie تحدد في مشيده اسم الـ Cookie وقيمته، وقد تعدل خاصيته Expires، ومن ثم تضيفه الى المجموعة Cookies التي تقبل كائنات من هذا النوع:

```
Dim userName As New HttpCookie("NAME", "تركي العسيري")
Dim userPassword As New HttpCookie("PASSWORD", "1234")

userName.Expires = Date.Now.AddYears(1)
userPassword.Expires = Date.Now.AddYears(1)

Response.Cookies.Add(userName)
Response.Cookies.Add(userPassword)
```

PDF Factory Pro trial version www.fastio.com

اما عند القراءة، فلا تنسى ان البيانات المرسلة من العميل الى الخادم تـصطادها مـن الخاصـية Request وليس Response:

```
Label1.Text = Me.Request.Cookies("NAME").Value        ' تركي العسيري
Label2.Text = Me.Request.Cookies("PASSWORD").Value    ' 1234
```

## الكائن HttpServerUtility

الكائن HttpServreUtility هو الخادم Server الذي يتم تنفيذ موقعك فيه، لا يحتوي هذا الكائن الا على خاصيتين الاولى MachineName تعود باسم جهاز الخـادم، امـا الخاصـية الثانيـة ScriptTimeout فتستطيع ان تحدد فيها فترة Timeout بوحدة الثواني، وهذه الفترة تمثل أقصى مدة يمكن تنفيذ الصفحة عليها ومن ثم انهاء تنفيذها ان تجاوزت الحد، تفيدك الطريقة السابقة عنـد الصفحات التي تقوم بتكوين حلقات لا نهائية –بطريق الخطأ- ولا تعود للمستخدم، او الـصفحات التي تنجز مهام لجمل استعلام طويلة جدا جدا مما تبطئ جهاز الخادم نفسه :

```
TextBox1.Text = Me.Server.MachineName
Me.Server.ScriptTimeout = 20
```

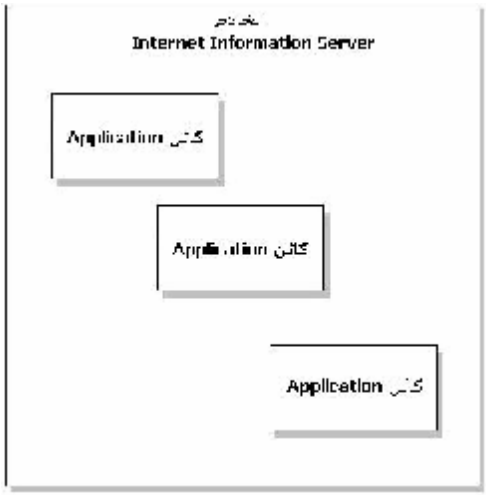
اما الطرق، فتوجد بها مجموعة من الطرق كالطريقة ()Execute والتي تمكنك من تنفيـذ صفحة aspx في داخل الصفحة الحالية.

## الكائن HttpApplicationState

انت تعلم ان الخادم IIS يمكنك من استضافة المواقع على جهازك، واذا نظرنا الى الموضوع بشكل تجاري، فان اغلب مواقع الاستضافة Hosting لا تقوم باستضافة كل موقع فـي جهـاز خـادم Server خاص، بل قد يحتوي الخادم على أكثر من موقع.

كيف يمكنك –كمبرمج ASP.NET- من التفريق بين موقعك والمواقع الأخرى على نفس الخادم IIS؟ والجواب هو عن طريق الكائن HttpApplicationState الذي يمثل موقعك الخاص والمعطى لك من المستضيف (**شكل 21-1**).

PDF




**شكل 21-1**: الخادم IIS قد يحتوي على اكثر من تطبيق ASP.NET Application.

**ملاحظة**

ان كنت قد ثبت نسخة من الخادم IIS في جهازك الشخصي، فكل موقع في الدليل الوهمي الجذري http://localhost/sitename يمثل موقع ويحتوي على كائن HttpApplicationState مستقل.

يمكنك الاستفادة من هذا الكائن بتعريف قيم يتم مشاركتها بين جميع الزوار لموقعك:

```
Me.Application("المشرف المناوب") = "عباس السريع"
Label1.Text = Me.Application("المشرف المناوب")
```

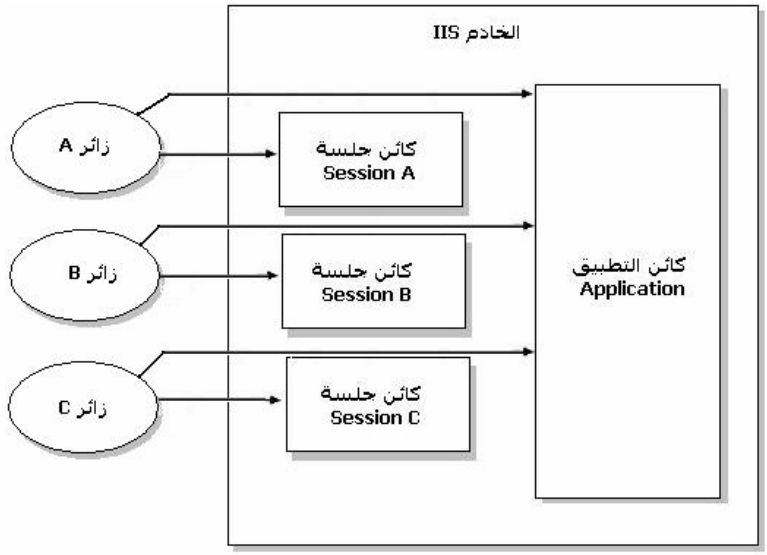
مع ذلك، عليك استدعاء الطرق Lock() و Unlock() قبل وبعد تعيين القيمة، وذلك حتى لا تسبب التعارضات عند استدعاء الصفحة من قبل اكثر من زائر بنفس الوقت:

```
Me.Application.Lock()
Me.Application("عدد الزوار") = CInt(Me.Application("عدد الزوار") + 1)
Me.Application.Unlock()
```

القيم التي أنشئت للتو تم حفظها في الخاصية الافتراضية Item() لذلك سمح لنا بتجاهلها، ولكن ضع في عين الاعتبار ان القيم السابقة ستحفظ من بداية تشغيل موقعك في المستضيف حتى نهايته وإيقافه من قبل المستضيف.

## الكائن HttpSessionState

ان كان الكائن HttpApplicationState يمثل موقعك، فان الكائن HttpSessionState يمثل احد زوار موقعك، ويتم إنشاء كائن HttpSessionState لكل زائر من زوار موقعك بمجرد طلبه لأول صفحة من الصفحات (**شكل 21-1**).




**شكل 21-2**: كائن HttpSessionState لكل زائر.

التعامل مع الكائن HttpSessionState شبيه بالتعامل مع الكائن السابق HttpApplicationState، ولكن ضع في عين الاعتبار ان القيم التي تحفظ في الكائن HttpSessionState **خاصة وتابعه لعميل واحد فقط**، وكل عميل سيكون له نسخة خاصة من الكائن:

```
Me.Session("Name") = "عباس السريع"
Me.Session("Password") = "123"
```

PDF Factory Pro trial version www.fastio.com

توجد خاصية رائعة جدا وهي الخاصية IsNewSession التي تعود بالقيمة True ان كان الطلب للصفحة هو اول طلب للزائر، مما يعني كائن الجلسة HttpSessionState الخاص به جديد وتم إنشائه للتو:

```
If Me.Session.IsNewSession Then
    Me.Application("Counter") = Me.Application("Counter") + 1
End If
```

# الملف Global.asax

عندما أنشأنا مشروع ASP.NET جديد في الفصل السابق، قامت بيئة التطوير Visual Studio .NET. باضافة الملف Global.asax ضمن قائمة الملفات في نافذة مستكشف الحل، وعليك معرفة ان كل تطبيق ASP.NET يحتوي –على الأكثر- ملف Global.asax واحد فقط يتم وضعه في المجلد الجذري للمشروع.

يقوم محرك ASP.NET في الخادم بتحميل الملف Global.asax عند بداية تنفيذ موقعك، وسيضل هذا الملف قيد العمل حتى يتم إيقاف عمل الموقع، وفي كل مرة تستدعى فيها صفحة من صفحات موقعك aspx، يتم تنفيذ الشيفرة والموجودة في هذا الملف.

تستطيع تعريف جميع المتغيرات او الثوابت العامة Global في الملف Global.asax لتتمكن من الوصول لها بين صفحات موقعك المختلفة، كما يمكنك الاستفادة من مجموعة من الإجراءات (تسمى أحداث ايضا) لها طابع خاص كما سترى في الفقرات التالية.

## الإجراءات xxxStart() و xxxEnd()

عندما تفتح هذا الملف، ستجد ان بيئة التطوير قد عرفت الفئة Global تحتوي على مجموعة من الإجراءات:

```
Imports System.Web
Imports System.Web.SessionState

Public Class Global
    Inherits System.Web.HttpApplication

    Sub Application_Start(ByVal sender As Object, _
        ByVal e As EventArgs)

    End Sub
    …
End Class
```

PDF

الإجراء Application_Start() يتم تنفيذه مع بداية تشغيل موقعك، وان تم ايقاف موقعك من قبل المستضيف سيتم تنفيذ الاجراء Application_End()، بينما الاجـــرائين Session_Start() و Session_End() سيتم تنفيذهما بمجرد انشاء كائن HttpSessionState جديد او إنهائـــه، اي – بعبارة اخرى- بمجرد دخول زائر جديد الى موقعك، لذلك يمكنك الاستفادة من هـــذه الإجـــراء – مثلا- في معرفة عدد الموجودين حاليا في موقعك:

```
Public Class Global
    Inherits System.Web.HttpApplication

    …
    …
    Sub Application_Start(ByVal sender As Object, ByVal _
        e As EventArgs)

        Me.Application("CurrentVisitors") = 0
    End Sub

    Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
        Me.Application.Lock()

        Me.Application("CurrentVisitors") = _
            CInt(Me.Application("CurrentVisitors") + 1)

        Me.Application.UnLock()
    End Sub

    Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
        Me.Application.Lock()

        Me.Application("CurrentVisitors") = _
            CInt(Me.Application("CurrentVisitors") - 1)

        Me.Application.UnLock()

    End Sub

End Class
```

## الإجراء Global_Error()

عرفت في الفصل السابق كيف يمكن لصفحة النموذج Web Form من تعريف حدثها Error يتم تنفيذه بمجرد حدوث استثناء غير متفادى في الصفحة، مع ذلك لست بحاجة الى كتابـــة الـــشيفرة المطلوبة في كل ملف من ملفات صفحات موقعك –خاصة ان كان عددها كبيرا، اذ يكفي تعريفك

للاجراء Global_Error() والذي تكتب فيه الشيفرة المطلوبة ليتم تنفيذها بمجرد وقوع استثناء غير متفادى في اي صفحة من صفحات موقعك.

يمكنك معرفة الاستثناء الذي وقع عن طريق الخاصية GetLastError والتابعة للكائن Server، مع ذلك هذه الخاصية تعود دائما بكائن استثناء HttpUnhandledException، لذلك ستحتاج الى الوصول الى الخاصية InnerException والتابعة لكائن الاستثناء:

```
Public Class Global
    Inherits System.Web.HttpApplication

    …
    …
    Sub Global_Error(ByVal sender As Object, ByVal e As EventArgs)
        Dim exp As Exception

        exp = Me.Server.GetLastError.InnerException

        Response.Write(exp.Message)
        Response.End()
    End Sub
End Class
```

# الأمان Security

حسنا، موضوع الأمان موضوع كبير جدا جدا، وعند ربطه بتطبيقات ASP.NET فانه يشمل الخادم IIS ونظام التشغيل Windows، وبما ان هذا الكتاب لا يزال مختص ببرمجة Visual Basic .NET، فلن اتحدث بالتفصيل حول هذه المواضيع، الا انني ملزم بالتلميح اليها -ولو باختصار- حتى تستوعب العلاقة بينها وبين صفحات ASP.NET.

## مدخلك إلى الصلاحيات

حتى تفهم الصلاحيات عليك التفريق بين المصطلحين **التصديق Authentication والسماح Authorization**. بالنسبة للمصطلح الاول، فيمكننا تعريف التصديق على التعرف على العميل (الذي هو زائر الصفحة) والتصديق على هويته، وهذه بحد ذاتها تتعلق بحساب العميل User Account في خادم نظام التشغيل Windows Server، وان لم يكن لهذا العميل حساب في نظام التشغيل، فيمكن ان يدخل كزائر Guest، وحينئذ نطلق على الطلبات **بالطلبات المتاحة Anonymous Requests**.

PDF

اذا اخذنا الموضوع من منظور شركات الاستضافة على الانترنت لمواقعك، فلن تنشئ هذه الشركات حسابات لكل زوار موقعك في خادمتها، لذلك فهي ستسمح بالطلبات المتاحة لتمكن الزوار من الدخول الى موقعك، ويمكن للخادم IIS من السماح لهذا النوع من الطلبات.

بعد التعرف على العميل فان عملية التصديق Authentication تنتهي، وتأتي عملية اخرى وهي التفويض او السماح Authorization والتي تحدد فيها الصلاحيات للعميل الذي تم تعريفها للوصول الى المصادر، والمصادر لا يشترط ان تحصرها في صفحات aspx فقط، بل تشمل كافة مصادر جهاز الخادم كالملفات الموجودة في الخادم، او الوصول الى قواعد البيانات.

خيارات التصديق والسماح يتم إعدادها من خلال اعدادات المستخدمين في نظام التشغيل Windows والخادم IIS، وفي اغلب الظن لن تكون لك قدرة في تغيير هذه الاعدادات الا ان كنت تعمل على جهازك الشخصي، او لك صلاحية المشرف Administrator على الشبكة المحلية، اما عند الحديث عن شركات الاستضافة لمواقع الانترنت، فلا اعتقد انك تستطيع تغيير شئ فيها.

لذلك، ضع في عين اعتبارك دائما قضية الصلاحيات التي ستكون لك عند تصميم موقعك وتنوي استضافته على شبكة الانترنت من قبل شركات استضافة، فمعظم شيفراتك البرمجية لن تعمل كما هو متوقع بسبب تغيير الصلاحيات، تخيل مثلا ان احد صفحاتك تقوم بانشاء ملف في القرص الصلب وأردت نقل ملفات الموقع على مستضيف لا يسمح لك بالقيام بهذه العملية.

## أوضاع التصديق في ASP.NET

بعيدا عن الخادم IIS ونظام التشغيل Windows، دعنا نلقي الضوء على اساليب التصديق Authentication التي يمكنك السيطرة عليها والتحكم فيها لتعرض صفحات موقعك بالشكل المطلوب، والسبب الذي جعلني اذكرها في هذا الكتاب هو قدرتك على تحريرها والتحكم فيها من خلال ملفات التهيئة Configuration Files كما سترى لاحقا.

مشروع ASP.NET الذي تنجزه يمكن له ان يعتمد وضع او اسلوب من اربعة اوضاع هي None، Windows، Forms، و Passport. في الوضع الاول فانك لا تطلب من صفحات موقع اجراء اي عمليات تصديق Authentication او سماح Authorization، وستكتفي بالاعدادات الموفرة لموقعك من خلال الخادم IIS ونظام التشغيل Windows.

اما الوضع الثاني –وهو Windows- فان عملية التصديق Authentication ستتم من قبل الخادم IIS ونظام التشغيل، ولكنك تستطيع التحكم في عملية السماح Authorization. يعيب هذه الطريقة، ان تعريف المستخدم (للتصديق عليهم) تتم باضافة حساباتهم من قبل نظام التشغيل Windows.

بالنسبة للوضع الثالث Forms، فيعطيك انت –كمطور الموقع- امكانية تحديد عملية التصديق Authentication والسماح Authorization بنفسك، وهو هذا هدفي من هذا القسم، لذلك خصصت فقرة كاملة ستراها قريبا.

اخيرا، في الوضع Passport فان عملية التصديق Authentication ستعتمد على خدمة Microsoft Passport، وهي خدمة عالمية تمكن المستخدمين من استخدام نفس الحساب User Account للدخول الى المواقع المختلفة، كخدمة Microsoft Messenger® تعتمد هذا على Microsoft Passport لتسجيل دخول الاعضاء (لمزيد من التفاصيل حول هذه الخدمة، يمكنك زيارة الموقع http://www.passport.com).

## ملفات التهيئة

قبل التحدث عن الوضع Forms للتصديق بودي تذكيرك بموضوع ملفات التهيئة، وتغيير طفيف يحدث فيه عند مشاريع ASP.NET.

في الفصل الحادي العشرات **المجمعات Assemblies** تحدثنا عن ملفات التهيئة Configuration Files، وذكرت ان لكل مجمع تنشئه يحتوي على ملف تهيئة التطبيق Application ينتهي بالامتداد config. تضع فيه كافة الخصائص التي تؤثر في سلوك تنفيذ المجمع.

اما عند الحديث عن تطبيقات ASP.NET، فان ملف تهيئة التطبيق يحمل الاسم web.config تضعه في المجلد الرئيسي للتطبيق، كما يمكنك تخصيص ملف تهيئة web.config لكل مجلد فرعي من مجلدات المشروع.

## الوضع Forms للتصديق

من خصالي كمبرمج، فاني أفضل كثيرا وضع Forms للتصديق Authentication، ففيه احدد قائمة بالمستخدمين الذين اود التصديق عليهم للدخول الى الصفحات الحساسة والإدارية في موقعي، وذلك يتطلب مني –كمبرمج- من تصميم صفحة تسجيل الدخول Login Page، ليتمكن المستخدمين من كتابة اسماء معرفاتهم وكلمات المرور الخاصة بهم.

الوضع Forms للتصديق يعمل بالشكل التالي: عندما يقوم العميل بطلب صفحة aspx وكانت هذه الصفحة مخصصة للمصدق عليهم Authenticated فقط، سيقوم محرك ASP.NET بالتحقق من تذكرة التصديق Authentication Ticket -والتي قد تكون محفوظة في الـ

Cookies مثلا، ان تم التحقق من تذكرة التصديق فسيتم عرض الصفحة المطلوبة، واذا لا فسيتم تحويل الزائر الى صفحة تسجيل الدخول.

ان اردت تطبيق التصديق Authentication في موقعك بالصورة التي رسمتها لك، اول خطوة عليك تنفيذها هي فتح ملف التهيئة web.config والبحث عن الوسم authentication والوسم authorization:

```
<configuration>
  <system.web>
    …
    …
    <authentication mode="Windows" />

    <authorization>
        <allow users="*" />
    </authorization>
    …
    …
  </system.web>
</configuration>
```

قم بتغيير وضع التصديق من Windows الى Forms، اكتب بداخله الوسم <forms> واضف به هذه المواصفات:

```
<configuration>
  <system.web>
    …
    …
    <authentication mode="Forms">
        <forms loginUrl="/login.aspx" name="dev4arabs" timeout="10"/>
    </authentication>
    …
    …
  </system.web>
```

اهم مواصفة في الوسم <forms> السابق، هي المواصفة loginUrl والتي تحدد فيها الصفحة التي سيتم توجيه المستخدم لها ان لم يتم التصديق عليها، بالنسبة المواصفات الاخرى فمعظمها لها قيم افتراضية وتقل أهميتها ولكني انصحك دائما باستخدامها، فالمواصفة name تحدد فيها اسم الـ Cookie الذي سيتم اعتباره كتذكرة تصديق وسيتم التحقق منها عند كل طلب لصفحات موقعك، اما المواصفة timeout فهي تفيدك لتحديد الفترة –بالدقائق- الغير نشطة ليتم انهاء جلسة كائن العميل Session وكأنه سجل خروجه.

اخيرا، جميع الزوار يمكنهم الدخول الى صفحات موقعك، وذلك بسبب الوسم الفرعي <allow> والتابع للوسم <authorization>، اذ عليك كتابة الوسم <deny> حتى تمنع الطلبات المتاحة Anonymous Request:

```
<configuration>
  <system.web>
    …
    …
    <authentication mode="Forms">
        …
        …
    </authentication>

    <authorization>
        <deny users="?" />
    </authorization>

    …
    …
  </system.web>
```

## الوسم <credentials>

في مشاريعك الجدية، فان اسماء الاشخاص وكلمات مرورهم سيتم قراءتها وحفظها في ملف قاعدة بيانات، وعليك كتابة كافة الشيفرات الضرورية في صفحة تسجيل الدخول للتحقق من اسم المستخدم وكلمة المرور ومن ثم تسجيل دخوله، اما هنا فسنستخدم الوسم <credentials> لتسهيل العملية وكتابة اسماء الاعضاء وكلمات مرورهم في داخل الوسم الفرعي <forms> التابع للوسم <authentication>:

```
<configuration>
  <system.web>
    …
    …
    <authentication mode="Forms">
        <forms … … …>
            <credentials passwordFormat="Clear">
                <user name="عباس السريع" password="123" />
                <user name="زكريا زعتر" password="456" />
            </credentials>
        </forms>
    </authentication>
    …
    …
  </system.web>
```

PDF Factory Pro trial version www.fastio.com

بالنسبة للمواصفة passwordFormat ففيها تحدد خوارزم التشفير، وبما اننا لا نود استخدام اي خوارزم هنا، فوضعنا القيمة "Clear" لها.

## صفحة تسجيل الدخول Login

والان عليك تصمم صفحة تسجيل الدخول وتحفظها بنفس الاسم الذي حددته في الخاصية loginUrl في الوسم <forms> السابق، قم بتصميم صفحة بإضافة أداتي TextBox لكتابة اسم المستخدم وكلمة المرور (**شكل 21-3**) –لا اعتقد انك بحاجة الى درس في تصميم صفحات تسجيل الدخول!

**شكل 21-3**: صفحة تسجيل دخول.

كما ذكرت سابقا، في مشاريعك الجدية ستعتمد على قاعدة بيانات لقراءة اسم المستخدم وكلمة المرور، وبما اننا اعتمدنا على الوسم <credentials> في الفقرات السابقة، فسنستدعي الطريقة المشتركة Authenticate() والتابعة للفئة System.Web.Security.FormsAuthentication، تتطلب هذه الطريقة اسم المستخدم وكلمة المرور، وستعود بالقيمة True ان تم التحقق منها:

```
Imports System.Web.Security
…
…
…
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    If FormsAuthentication.Authenticate(txtName.Text, _
        txtPassword.Text) Then

        FormsAuthentication.RedirectFromLoginPage(txtName.Text, True)
    Else
        Label1.Text = "خطأ في كلمة المرور"
    End If
End Sub
```

عند طلب اي صفحة من صفحات موقعك، سيتم توجيه الزائر الى صفحة تسجيل الدخول، وان تم التصديق عليه، سيتم نقله الى الصفحة التي طلبها –في اول مرة- باستخدام الطريقة RedirectFromLoginPage() والتي تسند لها القيمة True في وسطتها الثانية ليتم حفظ الـ Cookie في جهاز المستخدم.

# مواضيع متقدمة

سأتحدث في هذا القسم عن مجموعة متفرقة من المواضيع: كالتخزين المؤقت Caching، التصريح عن المتغيرات العامة، حماية الصور وعرضها ديناميكيا، ووحدات Http Modules.

## التخزين المؤقت Caching

القلب النابض لصفحات موقعك المبنية على تقنية ASP.NET ستقوم بتوليد صفحات HTML بعد قراءة بياناتها من قواعد بيانات، مع ذلك فان اغلب صفحات موقعك لن يتم تحديثها بشكل دوري الا كل فترة طويلة، لذلك يفضل **تخزين Caching** صفحات موقعك التي لا تتوقع تحديثها في اجهزة الزوار، حتى يخفف الضغط على جهاز الخادم Server.

يمكنك تخزين الصفحة في جهاز العميل باستخدام الموجه @OutputCache والذي تكتبه في اعلى ملف aspx وترفق معه الفترة –بالثواني- التي تود من تخزين الصفحة في الجهاز، وبعدها سيتم تحديث Refresh الصفحة عندما يزورها المستخدم مرة اخرى –ولا تنسى استخدام المواصفة VaryByParam:

```
<%@ OutputCache Duration="10" VaryByParam="none"%>
```

بعد اضافتك للسطر السابق للصفحة، سيتم تخزين الصفحة في جهاز العميل ولن يتم تحديثها الا كل 10 ثواني، وللتحقق من ذلك اضف أداتي Label و Button واكتب شيئا مثل:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    Label1.Text &= Date.Now & "<br>"
End Sub
```

عند تنفيذ الصفحة، ستلاحظ انك في كل مرة تضغط فيها على الزر لن يتم تحديث الصفحة الا كل عشرة ثواني **(شكل 21-4 أ بالصفحة التالية)**، اما إن ألغيت الموجه @OutputCache فسيتم تحديث الصفحة في كل مرة يطلبها الزائر **(شكل 21-4 ب)**.

بالنسبة للمواصفة VaryByParam ففيها تمكن عملية التخزين المختلفة لنفس الصفحة بحسب القيم المرسلة لها، فمثلا راقب الموجه التالي:

```
<%@ OutputCache Duration="10" VaryByParam="id"%>
```

هب مثلا ان الزائر قد ادخل الروابط التالية للصفحة السابقة:

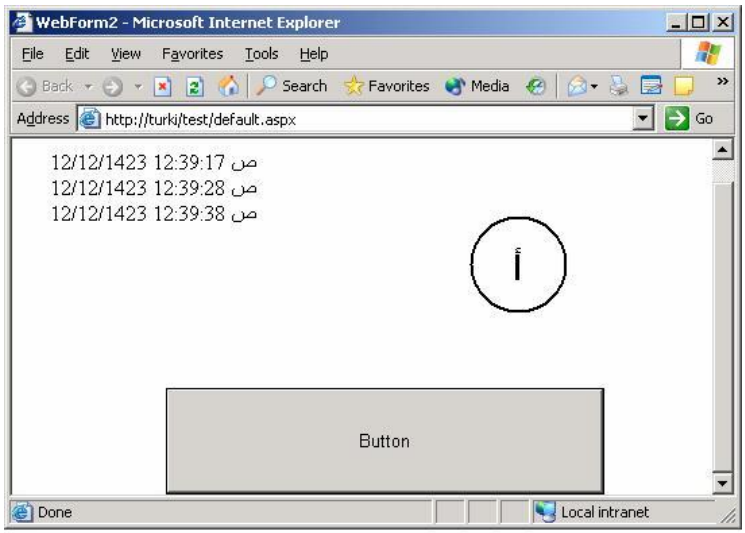
```
http://www.dev4arabs.com/testcache.aspx?id=1
http://www.dev4arabs.com/testcache.aspx?id=1
http://www.dev4arabs.com/testcache.aspx?id=1
http://www.dev4arabs.com/testcache.aspx?id=2
http://www.dev4arabs.com/testcache.aspx?id=2
http://www.dev4arabs.com/testcache.aspx?id=3
```

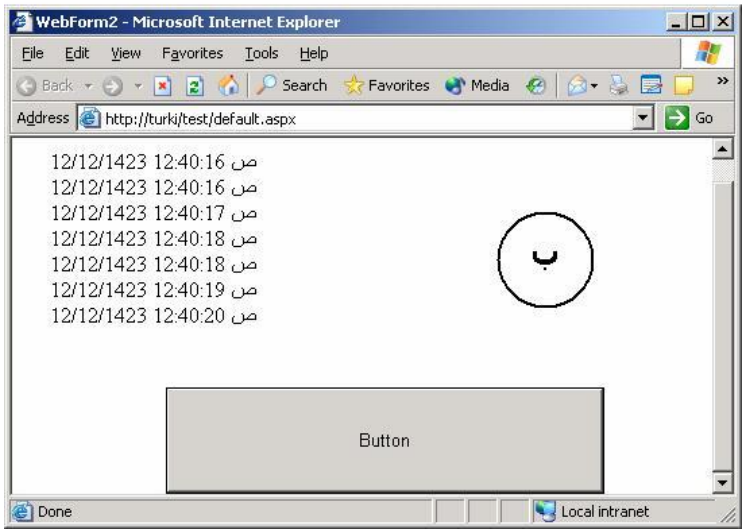
استنادا الى الروابط السابقة، فسيتم تخزين ثلاث نسخ من الصفحة، باختلاف القيم التي تحملها روابطها، ولن يتم تحديثها الا بعد مرور 10 ثواني او كتابة قيمة جديدة في القيمة المرسلة id بالرابط.

**ملاحظة**

الطلبات السابقة أرسلت الصفحة بأسلوب الـ GET، مع ذلك يمكنك تحديد القيم المرسلة بأسلوب الـ POST.

PDF




**شكل 21-4**: الصفحات المخزنة Cached لا يتم تحديثها الا بعد فترة.

اخيرا، يمكنك تحديد مكان تخزين الصفحة، اما في جهاز العميل Client، الخادم Server، او في جهاز آخر -غير الخادم Server- أدى الى توصيل او استقبال الطلب Downstream (يكون في اغلب الاحوال خادم البروكسي Proxy). تستطيع تحديد المكان عن طريق المواصفة Location، وان تجاهلتها فستكون قيمتها Any:

```
<%@ OutputCache … … Location="Downstream"%>
<%@ OutputCache … … Location="Server"%>
<%@ OutputCache … … Location="Client"%>
<%@ OutputCache … … Location="Any"%>
```

## المتغيرات العامة

سابقا في هذا الفصل، تعمدنا استخدام الكائن Application للاحتفاظ بالقيم التي تود الوصول لها من الصفحات المختلفة للموقع، مع ذلك انصحك بشدة بالاعتماد على المتغيرات حيث ان الوصول لها اسرع بمئات المرات، ولن تحتاج الى استخدام دوال التحويل معها، فبدلا من كتابة شيئا مثل:

```
Application("Counter") = CInt(Application("Counter") + 1)
```

يمكنك تعريف وحدة برمجية Module في اي ملف من ملفات موقعك وتعريف متغير بها:

```
Module MainModule
    Public Counter As Integer
End Module
```

وسيتم الوصول لها بالشكل التقليدي من اي صفحة في صفحات موقعك:

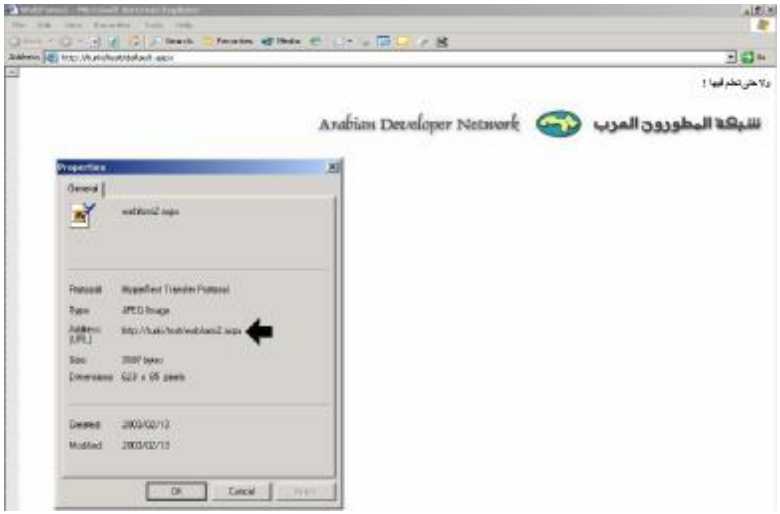
```
Counter += 1
```

## حماية الصور

قد تحمل ملفات موقعك باحد شركات الاستضافة، وكانت هذه الشركة تحدك على **مقدار تدفق بيانات Bandwidth** محدد، واكتشفت لاحقا بان بعض الأشخاص –الغير لبقين- كانوا يستخدمون روابط الصور في موقعك وعرضها في مواقعهم، مما كلفك الكثير من مقدار تدفق البيانات – ومقدار تدفق أموال محفظتك ايضا!

لا دموع بعد اليوم، فالكائن Response يحتوي على الخاصية ContentType والتي تمكنك من تغيير هيئة الصفحة، فصفحات aspx ترسل صفحات HTML بصيغة نصية بشكل ابتدائي، ولكنك تستطيع تحويله الى هيئة صور من نوع JPG:

```
Dim bmp As Bitmap = Bitmap.FromFile("dev4arabs.jpg")
Dim showPic As Boolean = True

' اكتب أي شرط هنا لعرض الصورة
If showPic Then
    Response.ContentType = "image/jpeg"
    bmp.Save(Me.Response.OutputStream, Imaging.ImageFormat.Jpeg)
End If
```

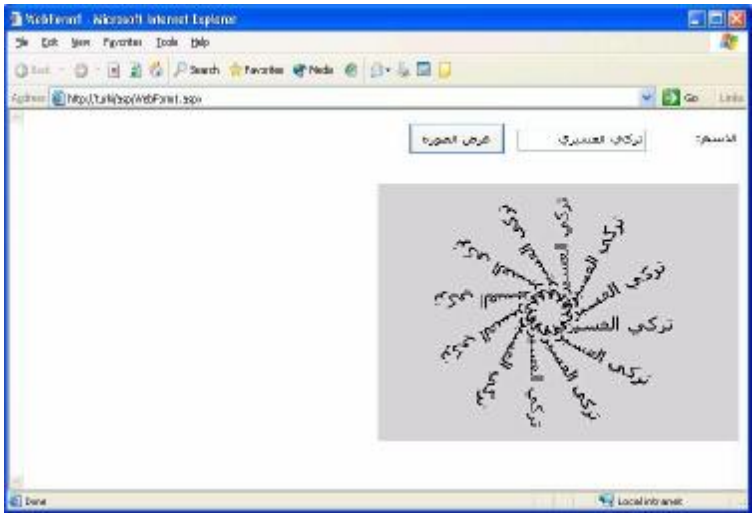



**شكل 21-5**: لن يتمكن احد من استعمال رابط الصورة إلا ان تحقق الشرط.

لاحظ ان رابط الصورة في (**الشكل 21-5**) يشير الى صفحة aspx وليس ملف صـــورة JPG. وبالنسبة لملف الصورة عليك وضعه في مجلد لا يمكن أن يصل اليه الزائر بكتابة رابـــط URL والخاص به.

## التوليد الديناميكي للصور

مهلا مهلا! الا تلاحظ اننا استخدمنا تقنية +GDI في الفقرة السابقة؟ إذا كان هذا لا يعني لك شيئا فما رأيك **(بالشكل 21-6)**:




**شكل 21-6**: التوليد الديناميكي للصور.

الفكرة من الصفحة السابقة هي عملية توليد الصور بشكل ديناميكي –وقت التنفيذ- وذلك بإستخدام فئات +GDI:

```
Dim bmp As New Bitmap(400, 300,
Drawing.Imaging.PixelFormat.Format16bppRgb565)
Dim gr As Graphics = Graphics.FromImage(bmp)
Dim Font As New Font("Tahoma", 16)
Dim text As String = "تركي العسيري"
gr.Clear(Color.LightGray)

' عملية القلب
Dim angle As Single
For angle = 0 To 360 Step 30
    gr.ResetTransform()
    gr.TranslateTransform(200, 150)
    gr.RotateTransform(angle)
    gr.DrawString(text, Font, Brushes.Black, 0, 0)
Next
```

```
Response.ContentType = "image/jpeg"
bmp.Save(Response.OutputStream, Imaging.ImageFormat.Jpeg)

' قتل الكائنات
Font.Dispose()
gr.Dispose()
bmp.Dispose()
```

بعدما عرفتك على أساسيات تطوير صفحات ASP.NET، لم يتبقى لنا الا عرض احد المزايا الجديدة في إطار عمل NET Framework. وهي **خدمات ويب Web Services** عنوان الفصل التالي.

PDF

# خدمات ويب Web Services

اختم معك الكتاب بهذا الفصل والذي يتحدث عن خدمات ويب وطريقة بنائها، ولا تستغرب من قلة عدد الصفحات المخصصة لهذا الفصل، فخدمات ويب ما هي إلا تطبيقات ASP.NET تقليدية، وكل ما تعلمناه في الفصول السابقة يمكنك تطبيقه مع خدمات ويب.

**ملاحظة**

المشاريع من نوع Web Services تدرج مجموعة من مجالات الأسماء التابعة لها، وإن لم تحدد هذا النوع من المشاريع، فقد تحتاج الى استيراد مجال الاسماء التالي:

```
Imports System.Web.Services
```

## مدخلك إلى خدمات ويب

كما ذكرت في الفصل الأول **تعرف على** .NET **Visual Basic**، **خدمة ويب** (تسمى احياناً **XML Web Service**) ما هي إلا برنامج يستقبل طلبات Requests ومن ثم تستجيب لها Response باستخدام بروتوكول HTTP تحت معايير لغة XML القياسية، وبذلك تتمكن ملايين المواقع المنتشرة حول العالم من تبادل البيانات فيما بينها وإنجاز الأعمال المطلوبة.

كمبرمج .NET Visual Basic، لست بحاجة إلى أي خبرة عملية في بروتوكول الاتصال HTTP او لغة الاستعلام XML، فالمشاريع من النوع Web Services تنجزها بنفس الشيفرة البرمجية بلغة .NET Visual Basic.

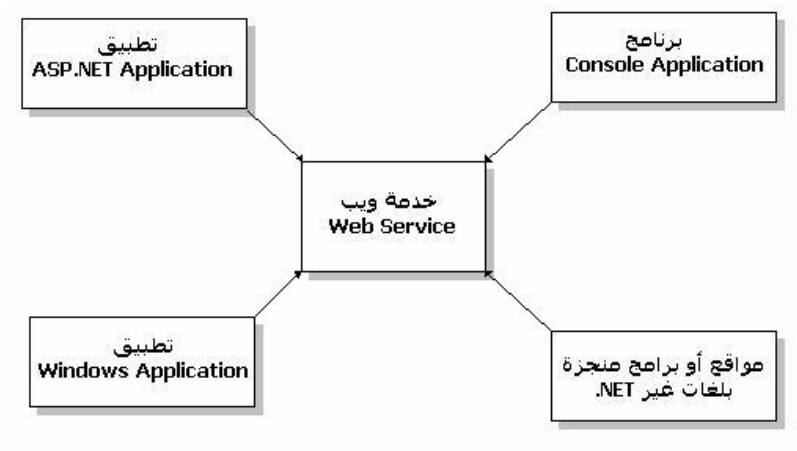
## كيف تعمل خدمة ويب؟

حتى أسهل عليك فكرة خدمات ويب، يمكنك اعتبار خدمة ويب على انها مكتبة DLL تستخدم فئاتها لتنشئ كائنات منها في برامجك، ولكن هذه المكتبة ليس في جهازك الشخصي و انما على موقع تصل اليه بشبكة الانترنت.

يمكنني -مثلا- تطوير فئة باسم VBNETBook تحتوي على طريقة GetUpdates() تعود هذه الطريقة بمصفوفة حرفية تمثل آخر الاخطاء والتحديثات المتعلقة بهذا الكتاب، لتتمكن من استدعائها بـ Visual Basic .NET بنفس الطرق التقليدية:

```
Dim book As New VBNETBook
Dim x As String

For Each x In VBNETBook.GetUpdates()
    ArabicConsole.WriteLine ( x )
Next
```

نستنتج من ذلك، ان خدمات ويب هي الوسيلة المثلى التي تمكن تطبيقات ومواقع .NET من الحصول وتبادل البيانات، دون الحاجة الى أي خبرة مسبقة في بروتوكول الاتصال HTTP ولغة وصف البيانات XML (**شكل 22-1**):




**شكل 22-1**: يمكن استخدام خدمة ويب من أي برنامج او موقع آخر.

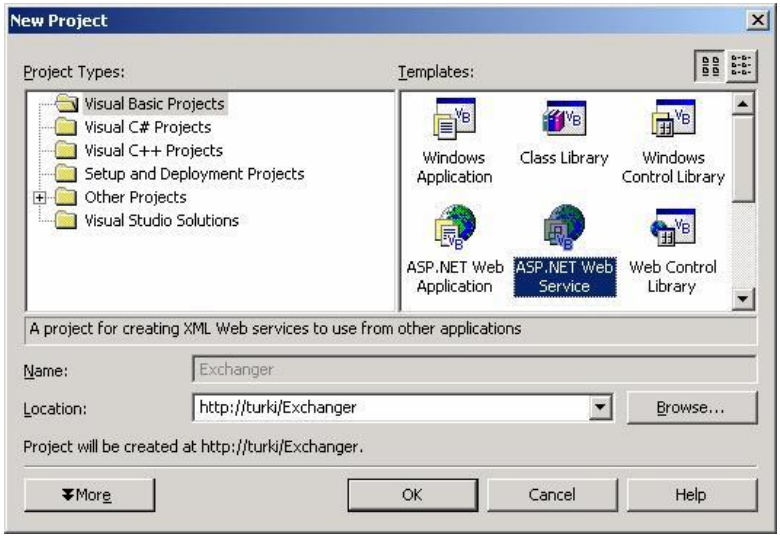
كما تلاحظ في الشكل السابق، يمكن ايضا للمواقع والتطبيقات المنجزة بلغات غيـــر NET. من استخدام خدمات ويب، فهي ليست سوى موقع يرسل ويستقبل الطلبات ببروتوكـــول الاتـــصال HTTP ولغة وصف البيانات XML.

# بناء خدمة ويب

يكفينا من الكلام حول خدمات ويب وما الذي يمكنه ان تقدمه لك، سنقوم في هذا القسم ببنـــاء اول خدمة ويب والتي تقدم لك خدمة تحويل العملات، لترسل لها مبلغ لعملتك المحلية، وستعود بالمقابل لها بالريال السعودي.

## إنشاء المشروع

كما ذكرت سابقا، لا تختلف الفكرة بين خدمات ويب عن المـــشاريع ASP.NET، الا ان بعـــض الملفات والتابعة لمشاريع ASP.NET لن تحتاجها، ولانشاء خدمة ويب جديدة اختر الامر New-Project> من قائمة File، وحدد الرمز ASP.NET Web Service من القوالب، ثم ضع اسم موقعا مناسب لمشروعنا كـــ http://localhost/Exchanger (**شكل 22-2**).




**شكل 22-2**: تحديد مشروع لخدمة ويب ASP.NET Web Service.

**ملاحظة**

لا تنسى تثبيت الخادم Internet Information Server، ان كنت تنوي تجربة خدمة ويب على جهازك الشخصي.

اضغط على الزر OK، ستقوم بيئة التطوير Visual Studio .NET ببناء ملفات مشروعك مشابهه لملفات مشاريع ASP.NET، فيوجد بها الملف web.config والملف global.asax (**شكل 22-3**) واللذان يعملنا بنفس الطريقة في مشاريع ASP.NET التقليدية.

**شكل 22-3**: ملفات مشروع خدمة ويب.

بالنسبة للملف Service1.asmx فهو الملف الرئيسي للخدمة التي ستكتب بها الشفرات، ويمكنك اعتبارها كمكتبة فئات Class Library يتم الاتصال بها عن طريق الانترنت باستخدام بروتوكول HTTP.

غير اسم الملف من Service1.asmx الى اسم مناسب، والسبب ان اسم الملف سيتم استخدامه برمجيا ويمكننا الوصول له لاحقا، ضع الاسم SaudiRiyals.asmx مثلا.

كما هو الحال مع ملفات aspx، الملف asmx تابع لمصمم يمكنك من اضافة وادراج ادوات عليه من صندوق الادوات ToolBox، مع ذلك فان خدمة ويب لا يفترض ان تستخدم ادوات مرئية وذلك لانها لا تعرض واجهة استخدام. يمكنك وضع اداة لا تعرض كالاداة FileSystemWatcher (وهي نسخة على شكل أداة من الفئة System.IO. FileSystemWatcher التي استخدمناها في الفصل السادس عشر **مواضيع متقدمة** لإنشاء خدمة Windows).

## كتابة الشيفرة

الملف SaudiRiyals.asmx يحتوي على فئة بالاسم Service1، وهي تمثل الخدمة التي تصممها، اي -بعبارة اخرى- اي فئة مشتقة وراثيا من الفئة System.Web.Services.WebService تمثل خدمة ويب.

غير اسم الفئة من الخاصية (Name) في نافذة الخصائص الى SaudiRiyals، ثم انقر نقرا مزدوجا على مصمم الخدمة لتفتح نافذة محرر الشيفرة، وقم بتعريف الاجرائين FromEG() و FromBRN():

```
Imports System.Web.Services

<System.Web.Services.WebService(Namespace:="http://tempuri.org/")> _
Public Class SaudiRiyals
    Inherits System.Web.Services.WebService

#Region " Web Services Designer Generated Code "
    …
    …
#End Region

    <WebMethod(Description:="تحويل المبلغ من الجنيه المصري الى الريال السعودي")>
    Function FromEG(ByVal amount As Decimal) As Decimal
        Return amount * 0.9
    End Function

    <WebMethod(Description:="تحويل المبلغ من الدينار البحريني الى الريال " _
    & "السعودي")>
    Function FromBRN(ByVal amount As Decimal) As Decimal
        Return amount * 10
    End Function

End Class
```

الاجرائين FromEG() و FromBRN() هما الاجرائين الرئيسين للخدمـــة، الاول يقـــوم بتحويل المبلغ المرسل بالجنيه المصري الى الريال السعودي، والثاني من الدينار البحريني، وكلا القيم من النوع Decimal.

**ملاحظة**

انا مبرمج ولست مصرفيا، ولـم اعمـل فـي القطاعـات الماليـة ابـدا فـي حياتي، لذلك تجاهل النسب الخاطئة والمستخدمة في عملية الـضرب لتحويل المبلغ فالغرض هو التوضيح فقط.

بالنسبة للمواصفة WebService المسطورة في اعلى الفئة والمواصفة WebMethod في اعلى الاجرائين، فاعرف–بشكل مبسط- ان الاولى تجعل الفئة فئة خدمة ويـــب Web Service Class، والثاني تجعل الإجراء قابل للوصول من خلال شبكة الانترنت.

## اختبار الخدمة من المتصفح

بمجرد كتابة الشيفرة السابقة وتعريف الاجرائين، فان الخدمة SaudiRiyals أصـــبحت جـــاهزة، ويمكنك اختبارها عن طريق المتصفح Internet Explorer® وذلك بتنفيذ البرنامج والضغط على المفتاح [F5] **(شكل 22-4)**.

PDF

**شكل 22-4**: اختبار الخدمة من المتصفحة Internet Explorer®.

تلاحظ في اعلى شريط العنوان بالمتصفح، بان رابط URL يشير الى ملف الخدمة SaudiRiyals.asmx، والذي تم تعريف الفئة SaudiRiyals فيه، كما يظهر لك في اعلاه اسم الخدمة والاجرائين FromBRN() و FromEG() اللذان عرفناهما للتو، بالاضافة الى نفس الوصف الذي كتبناه في المواصفه WebMethod لكلا الاجرائين.

كيف تم الحصول على هذه المعلومات؟ والجواب قام محرك ASP.NET باستخدام فئات الانعكاس Reflection Classes لعرض الطرق التي تحتويها الخدمة، كما تفعل تماما بيئة التطوير Visual Studio .NET بعد كتابتك لنقطة بعد اسم الكائن ليتم عرض جميع طرقه وخصائصه.

**ملاحظة**

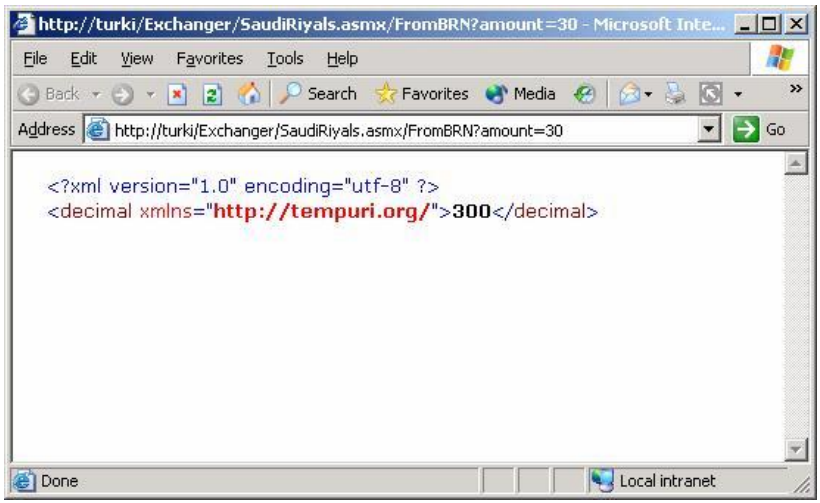
ان كان الملف يحتوي على اكثر من فئة لخدمة ويب، سيتم عرض الفئة الاولى فقط في المتصفح Internet Explorer.

تطبيقيا، لن يتم الوصول الى هذه الخدمة بهذه الطريقة، اذ ان المبرمجين سيصلون الى طرقها من داخل شيفراتهم المصدرية، ولكن الميزة التي يقدمها إطار عمل NET. Framework في عرض الخدمة على المتصفح ستفيدك كثيرا، وستسهل عليك تجربة واختبار الخدمة قبل اعتمادها ونشرها في كل ارجاء الارض عن طريق شبكة الانترنت.

دعنا الان نقوم بتجربة لاستدعاء احد الاجراءات، اضغط على الرابط FromBRN ليتم نقلك الى صفحة تمكنك من اسناد قيم لوسيطات الإجراء (**شكل 22-5**).

**شكل 22-5**: كتابة وسيطات الإجراء ()FromBRN.

ضع في عين الاعتبار، ان ليس كل القيم يمكنك اسنادها من خلال المتصفح لتجربة الخدمة، فمثلا قيم الكائنات Object والمعرفة من الفئات Classes على وجه التحديد لن تستطيع تجربتها وارسالها الى الاجراء كما في الشكل السابق، ولكن معظم انواع البيانات الابتدائية Primitive Types (كـ Integer، Long، String، Decimal، ...الخ) يمكنك تجربتها، ادخل القيمة 30 للوسيطة amount واضغط على الزر Invoke لتظهر لك القيمة التي يعود بها الاجراء (**شكل 22-6 بالصفحة المقابلة**).




**شكل 22-6**: عاد الإجراء بالقيمة 300 بعد تحويل العملة للمبلغ المرسل.

من الشكل السابق يتضح لنا ان الخدمة تعمل بنجاح ويمكنك اعتمادها، جرب اختبار الاجراء الاخر FromEG() وتاكد بنفسك نتيجة عملية التحويل، وستعود القيمة بالصيغة XML، لانه –كما ذكرت لك في بداية هذا الفصل- ان الغرض الرئيسي من خدمات ويب هو تبادل البيانات بين المواقع المختلفة بصيغة XML.

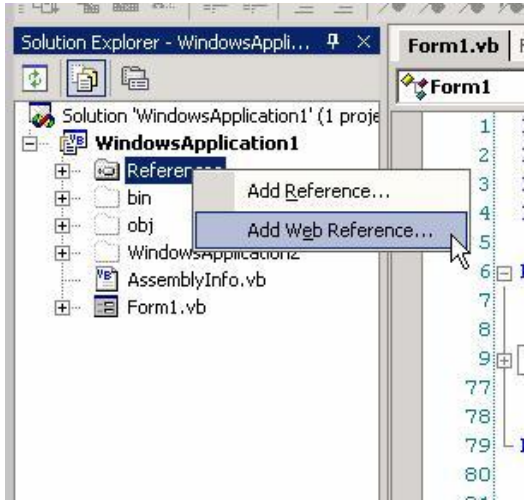
## استخدام الخدمة

لو عدنا الى تعريف خدمة ويب في مقدمة هذا الفصل، ستذكر ان خدمة ويب ما هي الا برنامج يستقبل الطلبات Requests ويعود بالاستجابات Responses بصيغة XML تحت بروتوكول HTTP، وهذا يعني اي برنامج او عملية يستخدم البروتوكول HTTP يمكن له استخدام الخدمة، مما يعني انه لا يشترط ان يكون العميل يستخدم برنامج منجز باحد لغات NET.، ليس هذا فقط بل لا يشترط ايضا ان يكون تطبيق تحت بيئة Windows، فلو كنت من مبرجي Macintosh، Linux، او حتى Unix وكنت على دراية كافية ببروتوكول الاتصال HTTP ولغة XML، يمكنك استخدام الخدمة دون اي مشاكل.

اما في هذا الكتاب سأريك كيف تستخدم الخدمة من برنامجك المكتوب بلغة Visual Basic NET.، ويمكنك استخدام الخدمة من اي نوع من المشاريع كـ Windows Service،

ASP.NET Application، Class Library، وغيرها وفي الخطوات التالية سنجرب الخدمة من مشروع Windows Application.

أنشئ مشروع من النوع Windows Application، وكما تفعل مع مكتبات الفئات التي تدرجها من نافذة المراجع، سنقوم بإدراج مرجع لخدمة ويب التي صممناها للتو، وحتى يمكنك اضافة مرجع لهذه الخدمة، انقر بزر الفأرة الايمن على رمز المشروع في نافذة مستكشف الحل Soulution Explorer واختر هذه المرة الامر Add Web Refrence -وليس Add Reference (**شكل 22-7**).

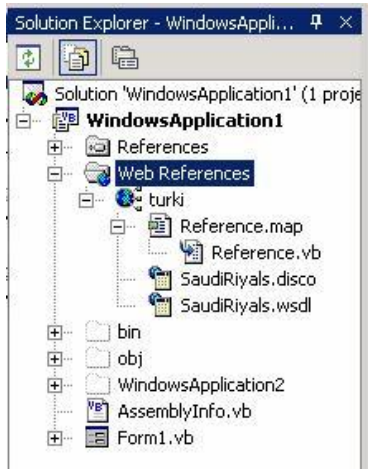



**شكل 22-7**: اضافة مرجع لخدمة ويب.

بمجرد اختيارك للامر السابق، سيظهر لك صندوق حوار Add Web Reference يمكنك من البحث عن الخدمات في شبكة الانترنت، في اعلى صندوق الحوار اكتب المسار الرابط الكامل للملف asmx الذي يحتوي على الخدمة (سيكون http://localhost/Exchanger/SaudiRiyals.asmx ان اتبعت الخطوات السابقة) ثم اضغط على المفتاح [Enter]، وبعدما يتم الاتصال بالخدمة ستعرض لك أعضائها في جهة اليسار من صندوق الحوار (**شكل 22-8**).

**شكل 22-8**: صندوق حوار Add Web Reference.

ان تم اصطياد الخدمة بالشكل الصحيح، فاضغط على الزر Add Reference في اسفل صندوق الحوار، سيتم الاتصال بالخدمة ومن ثم اضافتها الى خانة مراجع الخـــدمات Web References في نافذة مستكشف الحل Solution Explorer (**شكل 22-9**).




**شكل 22-9**: تم إضافة مرجع لخدمة ويب SaudiRiyals.

**ملاحظة**

لن تظهر لك كافة الملفات (**بالشكل** 22-9) الا ان حددت الاختيار Show All files في اعلى نافذة مستكشف الحل Solution Explorer.

والان تم اضافة مرجع الى هذه الخدمة، وان كنت لا تصدق ما تراه قم باضافة ادوات TextBox وزر Button، واكتب هذه الشيفرة في الحدث Click للزر:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click

    Dim test As New LocalHost.SaudiRiyals()

    txtEG.Text = CDec(test.FromEG(txtSaudi.Text))
    txtBRN.Text = CDec(test.FromBRN(txtSaudi.Text))
End Sub
```

جرب تنفيذ البرنامج، وقم بإدخال المبلغ بالريال السعودي في اداة TextBox المناسبة، ومن ثم اضغط على الزر Button لترى نتائج تحويل المبلغ الى العملات الاخرى، وهذا يعني عملية الاتصال بالخدمة تمت بنجاح واستخدمت بالشكل المطلوب (**شكل 22-10**).

**شكل 22-10**: استخدام خدمة ويب في مشروع Windows Application.

## تحديث الخدمة

عندما تجري أي تعديل على خدمة ويب –خاصة في مرحلة التجربة- عليك تحديثها في برنامج العميل دائما (وهو تطبيق Windows Application كما في المثال السابق) حتى تظهر التعديلات ويتم تنفيذ الخدمة بالشكل الصحيح. لعمل ذلك، انقر بزر الفأرة الايمن على رمز الخدمة في نافذة مستكشف الحل (**شكل 22-9 بالصفحة السابقة**)، واختر الامر Update Web Reference من القائمة المنبثقة.

الصفحة 735 هي نهاية الفصل الثاني والعشرون **خدمات ويب Web Services**، وفي نفس الوقت نهاية الجزء الخامس **برمجة ويب**، والتي تمثل نهاية كتاب **برمجة إطار عمل NET. باستخدام Visual Basic .NET**. اسأل الله العظيم بأسمائه الحسنى وصفاته العلى أن يجعله من العلم الذي ينتفع به، انه سميع مجيب الدعوات.

PDF

# لغة وصف البيانات XML

انتشرت لغة وصف البيانات eXtensible Markup Language انتشارا لا مثيل له بين كافة التطبيقات التجارية على اوسع نطاقها، ولو تمعنت قليلا في إطار عمل NET. Framework لاكتشفت ان البنية التحتية لاغلب ادواتها وفئاتها تعتمد XML كلغة قياسية لوصف البيانات. يمكنك عرض مواقع الانترنت المختلفة للحصول على مراجع وصيغ استخدامات لغة وصف البيانات XML، ولكني في هذا الملحق اعرض لك مقرر سريع للغة وصف البيانات XML، أخاطب به أولئك المبرمجين الذي لم يتشرفوا بمعرفتها لا من قريب ولا من بعيد.

## استخدام وسوم HTML

ان كنت من مصممي مواقع ويب المعتمدة على صفحات HTML فقد تعتقد ان استخدام وسوم HTML Tags كافي لتبادل البيانات بين المواقع، ولكن الحقيقة غير ذلك، والسبب ان وسوم HTML غرضها تنسيق Formating البيانات على المتصفحات، وهي لا تمثل البيانات بشكلها الاستقلالي.

فمثلا، لو وجدت وسوم HTML التالية في احد الصفحات:

```
<p>تركي العسيري</p>
<p>99</p>
```

ماذا تفهم من هذه البيانات؟ لاشئ والسبب قد يبدو منطقيا ان علمت ان وسوم HTML لا تشرح ولا توضح البيانات وانما تقوم بعرضها فقط، فقد يقوم احد الاشخاص باضا فت الوسوم السابقة في موقعه بهذا الشكل:

```
<p><b>المبرمج: </b></p> <p>تركي العسيري</p>
<p><b>العمر: </b></p> <p>99</p>
```

PDF

وقد يقوم اخر بتفسيرها بهذا الشكل:

```
<p><b>الطباخ: </b></p> <p>تركي العسيري</p>
<p><b>الوزن: </b></p> <p>99</p>
```

كيف ستقوم بتبادل البيانات مع التطبيقات والمواقع المختلفة ان لم تكن انت اصلا لديك اي خلفية عن الماهية الحقيقية للبيانات وتركيبتها الشكلية، وقد يصل الامر ايضا لحصولك على المعلومات الغير صحيحة، فمثلا ان تبادل بياناتك مع موقع يحتوي على الوسوم السابقة، قد تحصل على معلومات لا تعني لك شيئا:

```
تركي العسيري</b> :الاسم<b></p> <p></b>: خ
```

ويا ليت المشكلة تقف عند النقطة السابقة، فلو قرأت صفحة تحتوي على وسوم HTML وطلبتك تجريد نصوصها ومعلومات، سيكون الامر مقدور عليه –في قليل من الأحيان، فمثلا الوسم التالي:

```
<p>تركي العسيري</p>
```

يمكنك تجريده بحذف الثلاث الحروف الأولى والأربع الأخيرة من الوسم، لتصبح النتيجة:

```
تركي العسيري
```

تخيل مثلا ان قام صاحب الموقع قام بتعديل الخط في واجهة موقعه، وكتب شيئا مثل:

```
<p><font face="Tahoma" size="4">تركي العسيري</font></p>
```

فلو نفذت نفس البرنامج السابق لتجريد الاسم، فستكون النتيجة:

```
<font face="Tahoma" size="4">تركي العسيري</font>
```

واعتقد ان الاسم السابق ستصعب قراءته فما بالك بنطقه! اما لو لم يكن التعديل طفيف كما في الحالة بالصفحة بالمقابلة:

PDF

```
<table border="1" cellpadding="0" cellspacing="0" style="border-
collapse: collapse" bordercolor="#111111" width="100%" id="AutoNumber
2"><tr><td width="100%">      <table border="1" cellpadding="0" cellsp
acing="0" style="border-collapse: collapse" bordercolor="#111111" wid
th="100%" id="AutoNumber3"><tr><td width="100%"><table border="1"
cellpadding="0" cellspacing="0" style="border-collapse: collapse"
bordercolor="#111111" width="100%" id="AutoNumber4"><tr><td width=
"100%"> <table border="1" cellpadding="0" cellspacing="0" style
="bordercollapse:collapse"bordercolor="#111111"width="100%"id="AutoNu
mber5"><tr><td width="100%">  <table border="1" cellpadding="0"
cellspa
cing="0" style="border-collapse: collapse" bordercolor="#111111" width
="100%" id="AutoNumber6"><tr><td width="100" height="51"bordercolordar
k="#008000" bgcolor="#00FF00" align="center"><h1 dir="rtl" align="cent
er"><font face="Tahoma" size="4"color="#FFFFFF">تركي</font></h1></td><
/tr><tr><td width="100" height="51" bordercolordark="#008000" bgcolor
="#00FF00"align="center"><b><fontface="Tahoma"size="4"color="#FFFFFF">
العسيري</font></b></td></tr><tr><td>width="100%"> </td></tr></table
>
</td></tr></table></td></tr></table></td></tr></table></td></tr>
</table>
```

فعملية تجريد الاسم تركي العسيري ستتطلب كتابة شيفرة برمجية لمستعرض Browser كامل!!!

# تمثيل البيانات بـ XML

اما مع لغة وصف البيانات XML، فإننا نتحدث عن عملية وصف بصيغة قياسية، فلو كان لـــدينا الجدول التالي:

| جدول الطلاب | | |
|---|---|---|
| المعرف | الاسم | العمر |
| 1 | عباس السريع | 99 |
| 2 | برعي ابو جبهة | 88 |
| 3 | شرشبيل بن جوزة | 77 |

يمكننا وصفه نصيا ونقول:

جدول الطلاب

| | | |
|---|---|---|
| المعرف: 1 | الاسم: عباس السريع | العمر: 99 |
| المعرف: 2 | الاسم: برعي ابو جبهة | العمر: 88 |
| المعرف: 3 | الاسم: شرشبيل بن جوزة | العمر: 77 |

الوصف السابق كان باللغة العربية، وهي لغة لا يتقنها سوى نسبة قليلة من البشر، لذلك سنقوم بوصفه بلغة XML وهي صيغة عالمية متوافق معها جميع التطبيقات التي تستخدمها، ونكتب شيئا مثل:

```
<?xml version="1.0" encoding="utf-8" ?>
<table>
        <record>
                <ID>1</ID>
                <name>عباس السريع</name>
                <age>99</age>
        </record>
        <record>
                <ID>2</ID>
                < name >برعي ابو جبهة</name>
                <age>88</age>
        </record>
        <record>
                <ID>3</ID>
                <name>شرشبيل بن جوزة</name>
                <age>77</age>
        </record>
</table>
```

بالنسبة للسطر الاول من الشيفرة السابقة، فهو يمثل توقيع ملف XML، وكما تفعل مع صفحات HTML والتي توقعها بالوسم <HTML> في اعلى الصفحة. الكلمة version هو رقم اصدار لغة XML، اما encoding فتمثل صفحة المحارف المستخدمة في الملف، يمكنك نسخ الشيفرة السابقة وحفظها في ملف نصي بالامتداد test.xml لتتمكن من استخدامه مع كافة التطبيقات والمواقع التي تعتمد لغة XML لوصف البيانات (عليك استخدام نفس صفحة المحارف لحظة حفظ الملف).

PDF Converter

# العناصر Elements

كما رأيت في الفقرة السابقة، لغة وصف البيانات XML لا تملك كلمات محجوزة خاصة بها، وكل شيء يأتي من عندك، فهي مجرد نسق معين تتهجه لوصف البيانات، الوسوم <xxx> التي استخدمناها تسمى **العناصر XML Elements**.

على عكس وسوم HTML، فان عناصر XML حساسة لحالة الأحرف case-sensitive، فهي تفرق بين الحروف الكبيرة Capital والصغيرة Small. المزيد ايضا، عند انشاء عنصر من عناصر XML عليك اغلاقه دائما بكتابة نفس اسم العنصر مسبقا بالرمز /:

```
<test>
    …
</test>
```

مع ذلك، ان كان العنصر لا يحتوي على قيمة، فيمكنك اغلاقه بهذه الطريقة:

```
<test />
```

اخيرا، لا تنسى ان العناصر يمكن ان تكون متداخلة:

```
<aa>
        <bb>
                <cc />
        </bb>
</aa>
```

# المواصفات Attributes

**المواصفات Attributes** هي خصائص وقيم إضافية تسندها داخل العناصر، فالعنصر التالي يحتوي على مواصفتين هما X و Y، قيمة الاولى 10 والثانية 20 (يمكن ان تكون حروف ايضا):

```
<myelement x="10" y="20">value</myelemnt>
```

يمكنك الاستفادة من المواصفات في وصف أكثر للعنصر، فمثلا يمكنا إضافة المواصفة PrimaryKey لعناصر الجدول السابق لنبين بذلك ان العنصر الحالي هو مفتاح ابتدائي:

PDF

```
<?xml version="1.0" encoding="utf-8" ?>
<table>
        <record>
                <ID Primarykey="True">1</ID>
                <name>عباس السريع</name>
                <age>99</age>
        </record>
        <record>
                <ID Primarykey="True">2</ID>
                < name >برعي ابو جبهة</name>
                <age>88</age>
        </record>
        <record>
                <ID Primarykey="True">3</ID>
                <name>شرشبيل بن جوزة</name>
                <age>77</age>
        </record>
</table>
```

## التعليقات Comments

اما **التعليقات Comments** فهي معلومات إضافية تكتب في ملف XML غرضها شرح العناصر ولا تؤثر بأي شكل من الاشكال، يمكنك كتابتها داخل التركيب <-- و --!>:

```
<test>

        <!--
                هذا تعليق لا يؤثر بشئ
        -->
</test>
```

## خاتمة

في هذا المحلق عرضت لك لغة وصف البيانات XML والتي عبارة عن لا شيء سوى مجموعة من التنسيقات تستخدمها لوصف البيانات، تذكر أن هذه التنسيقات هي صيغ قياسية للغـــة XML. توجد بعض التنسيقات الإضافية يمكنك البحث عنها في مواقع الانترنت.

PDF Factory Pro trial version www.fastio.com

# لغة الاستعلام SQL

اذا اردت ان تخاطب قاعدة بيانات، فانك لن تتحدث معها باللغة الصينية وانما بلغة موحدة خاصة بقواعد البيانات هي لغة الاستعلام المركبة Structured Query Language (SQL). وهي لغة قياسية مدعومة في جميع الهيئات المختلفة لقواعد البيانات. عشرات الكتب المنتشرة هنا وهناك مختصة في لغة SQL ومئات المواقع –بما فيها مكتبة MSDN- توفر لك مراجع شاملة للغة SQL، ومن الصعب علي شرح جميع اوامر وعبارات لغة SQL، لذلك يعرض لك هذا الملحق مقرر سريع للغة الاستعلام SQL تخاطب به أولئك المبرمجين الذين لم يحصل لهم شرف ممارسة واستخدام لغة SQL.

## تصنيف أوامر لغة الاستعلام

بامكاننا تصنيف اوامر وعبارات لغة SQL الى صنفين، الاول عبارات **لغة تعريف البيانات Data Definition Language (DDF)** والثاني هي **عبارات لغة صيانة البيانات Data Manipulation Language (DML)**. اوامر DDF هي أوامر وعبارات خاصة ببنية وتركيبة قاعدة البيانات، فهي تمكنك من إنشاء الجداول Tables وتعرف الحقول Fields وغيرها، اما عبارات DML فهي اقرب الى الاستعلام عن البيانات في السجلات Records وإضافة وحذف سجلات اخرى، والفقرات التالية تختص ببعض اوامر DML فقط، فهي اكثر استخداما من اوامر DDF والتي يفضل المبرمجون استخدام نظام قاعدة البيانات لتعريف بنية وتركيبة القاعدة عوضا عن اوامر وعبارات DDF.

## الامر SELECT

يعتبر الامر SELECT بلا شك هو اكثر اوامر SQL استخداما والذي يعود بمجموعة من السجلات تحددها في نفس الامر. المثال التالي يعود بجميع السجلات الموجودة في الجدول "بيانات الموظفين" مع جميع الحقول التابعة له:

```
SELECT * FROM [بيانات الموظفين]
```

بامكانك تحديد حقول معينة لزيادة سرعة الاستعلام، فالمثال التالي يعود بجميع الـسجلات الموجودة في نفس الجدول مع تحديد حقل الاسم وتاريخ الميلاد فقط:

```
SELECT [الاسم], [تاريخ الميلاد] FROM [بيانات الموظفين]
```

واذا اردت استخلاص سجلات معينة توافق شرط معين استخدام العبارة WHERE، فالمثال التالي سيعود بجميع السجلات التي تكون فيها جنسية الموظف "سعودي":

```
SELECT * FROM [بيانات الموظفين]
WHERE
        [الجنسية] = 'سعودي'
```

بامكانك استخدام ادوات الربط AND، OR وادوات المقارنة >، <، LIKE ...الخ:

```
SELECT * FROM [بيانات الموظفين]
WHERE
        [الجنسية] = 'سعودي'
        AND [الاسم] LIKE '%ت'
        OR [الراتب] < 9000
```

واذا اردت تحديد مجال قيم معين فاستخدم المعامل BETWEEN:

```
SELECT * FROM [بيانات الموظفين]
WHERE
        [الراتب] BETWEEN 2000 AND 10000
```

او مجموعة قيم باستخدام المعامل IN:

```
SELECT * FROM [بيانات الموظفين]
WHERE
        [الجنسية] IN ('عراقي', 'سعودي', 'مصري')
```

المزيد ايضا، تستطيع فرز (ترتيب) السجلات بشكل تصاعدي باستخدام العبارة ORDER BY:

```
SELECT * FROM [بيانات الموظفين] ORDER BY [الاسم]
```

او تنازلي باستخدام الكلمة المحجوزة DESC:

```
SELECT * FROM [بيانات الموظفين] ORDER BY [الاسم] DESC
```

# الامر INSERT INTO

يمكنك الامر INSERT INTO من اضافة سجلات جديدة الى الجدول المحدد:

```
INSERT INTO [بيانات الموظفين]
        ( [الاسم], [الجنسية] )
VALUES
        ( 'تركي العسيري', 'سعودي' )
```

# الامر UPDATE

تستخدم الامر UPDATE لتحرير قيمة حقل في سجل معين تحـــدده فـــي العبـــارة WHERE، فالجملة التالية ستقوم بتعيين القيمة 10000 في حقل الراتب التابع للسجل الذي معرفه 32421:

```
UPDATE [بيانات الموظفين] SET
        [الراتب] = 10000
WHERE
        [المعرف] = 123
```

ضع في عين الاعتبار ان التعديل قد يشمل مجموعة سجلات توافق الشرط الموجود في العبـــارة WHERE، فالجملة التالية ستقوم بزيادة جميع رواتب الجنسية "سعودي" الى الضعف:

```
UPDATE [بيانات الموظفين] SET
        [الراتب] = [الراتب] * 2
WHERE
        [الجنسية] = 'سعودي'
```

وان لم تكتب شرط باستخدام العبارة WHERE، فان جميع السجلات سيتم تعديلها:

```
UPDATE [بيانات الموظفين] SET
        [الراتب] = 0
```

PDF

# الامر DELETE

من الواضح ان الامر DELETE لا يقوم بعملية نسخ للسجلات وانما حذفها:

```
DELETE FROM [بيانات الموظفين]
```

في العادة لن تحذف الا عدد معين من السجلات الذي يوافق جملة شرطية باستخدام WHERE:

```
DELETE FROM [بيانات الموظفين]
WHERE
        [العمر] < 18
```

# خاتمة

في هذا الملحق عرضت مقرر سريع للغة الاستعلام SQL وهي ذات معايير ومواصفات موحدة عالميا ANSI-92 مدعومة في اغلب نظم ادارة قواعد البيانات ومزودات .NET Data Providers والتي تستخدم مع كائنات ADO.NET. تذكر أنني تحدثت عن DML وتجاهلت اوامر DDF والتي يمكنك الاستغناء عنها واستخدام نظام إدارة قواعد البيانات لانشاء الجداول وتعريف الحقول.

PDF

PDF