7

Importing, Linking, and Exporting Using External Data Sources

In Chapter 5, I covered the basics of using ADO and SQL to work with data sources. All the ADO and SQL examples dealt with data stored in tables in a database. However, in today's world of technology, you often work with data and applications in a variety of formats, such as text files and spreadsheets. You may need to import or link data from various sources into your database to avoid having to retype all the information that is already stored electronically in another format. At some point, another application might need the data in your application or you may want to get data out of your application for another reason. In that case, you can export information from your application into another format.

In this chapter, I will explore the various ways to use VBA code to link, import, and export to external data sources. The chapter will cover:

- The difference between linking, importing, and exporting
- Linking, importing, and exporting to external Access databases (MDB files)
- Linking, importing, and exporting to SQL Server databases
- Linking, importing, and exporting to other files such as spreadsheets and text files
- Creating and sending an e-mail programmatically
- The definition of a Web service and how you can use data returned from a Web service

These techniques will enable you to build robust applications that interact with a various applications and formats.

Linking, Importing, and Exporting Overview

Linking to external data means creating a pointer to an external data source that allows you to interact directly with the underlying data. *Importing* external data literally imports a copy of the

data into your application. *Exporting* data refers to the idea of extracting data from your application to an external file or format.

Here are some situations when you should consider linking:

- The data resides on a database server that your application and others can use.
- The data is used by another program that requires the native file format.
- The underlying data needs to be updated on a regular basis in its native format.

Here are some instances when you should consider importing:

- An existing system is being migrated to a new application and the data from the old system will be used in the new application. (In some cases, you may be able to migrate to another system but keep the data on a database server without needing to import the data).
- Numerous data operations must be run against the data from another source. You can obtain performance improvements by importing the data, but the underlying data will be out of sync if you make any changes to the data after it is imported.

Access allows you to link to and import from data sources such as Access (Jet) databases, SQL Server databases, other ODBC databases, Microsoft Sharepoint, XML documents, HTML documents, text files, Microsoft Exchange, Microsoft Outlook, and spreadsheets such as Microsoft Excel and Lotus.

> *Many of the techniques covered in this chapter can also be implemented using menus and wizards in Access. To import or link data using the menus and Wizards, select File ➪ Get External Data and then select either the Import or Link Tables option. You can export data by selecting a particular object (table, for example) in the Database Window, right-clicking, and selecting the Export option from the pop-up box.*

Now that you understand the high-level concept of importing, linking, and exporting, you can jump right in to learning the techniques that will allow you to work with some of these supported data sources.

Create a blank database to use for the examples in this chapter. To do so, select File ➪ New ➪ Blank Database and specify Ch7CodeExamples for the filename and then click the Create button.

Access and Other Databases

You can use the `TransferDatabase` method of the `DoCmd` object to import from, link to, and export data to Access and several other databases, including SQL Server and Oracle. The basic syntax of the `TransferDatabase` method is shown in the following code.

```
DoCmd.TransferDatabase(TransferType, DatabaseType, DatabaseName, ObjectType,
Source, Destination, StructureOnly, StoreLogin)
```

Various parameters are used to specify how the method should execute. The following table explains the use of each parameter.

Parameter	Description
TransferType	Type of transfer to be performed. Valid choices are `acImport` (default), `acLink`, and `acExport`.
DatabaseType	Type of database being used. Access is the default. See the help documentation for a complete list and for the exact syntax for a particular database.
DatabaseName	The full name, including the path, of the database being used.
ObjectType	The type of object that has data you want to work with. The default is `acTable`.
Source	Name of the object whose data you want to work with.
Destination	Name of the object in the destination database.
StructureOnly	Use `True` to work with the structure only and `False` to work with the structure and data. `False` is the default.
StoreLogin	Whether to store the login and password. `False` is the default.

Let's look at an example. Suppose you want to import data from an Access database called SampleDb. The data you want to import is in a table called `Sales`, and you want it to be imported to your current database under the name `tblSales`. You could run the following command from your current Access application.

```
DoCmd.TransferDatabase acImport, "Microsoft Access",_
    "SampleDb.mdb", acTable, "Sales", "tblSales"
```

Here's an example that shows linking to a table called `Sales` in an ODBC database called Wrox.

```
DoCmd.TransferDatabase acLink, "ODBC Database", _
    "ODBC;DSN=DataSourceName;UID=username;PWD=pwd;LANGUAGE=us_english;" _
    & "DATABASE=Wrox", acTable, "Sales", "dboSales"
```

The ODBC data source name can point to any database that ODBC supports, including SQL Server and Oracle, to name a few examples. As with any linking operation, you see the table or tables from the Database Window in Access.

Try It Out — Importing Data from the Sample Northwind Database

Now it's your turn to try this out. Let's import data from the sample Northwind database that comes with Access.

1. Insert a new module into your Ch7CodeExamples database.
2. Add the following code to the module.

```
Sub TestTransferDatabase()

'import from Northwind
```

```
DoCmd.TransferDatabase acImport, "Microsoft Access", _
    "C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb", _
acTable, "Employees", "tblEmployees"

End Sub
```

3. Modify the preceding path to the location on your hard drive where Northwind.mdb is located. If you do not have the sample Northwind database installed, change the previous parameters to reference the Access database that you do have.
4. From the Immediate Window in the Visual Basic Editor, type `TestTransferDatabase` and press Enter to run the procedure.
5. Open the Database Window and you should see a screen similar to Figure 7.1.

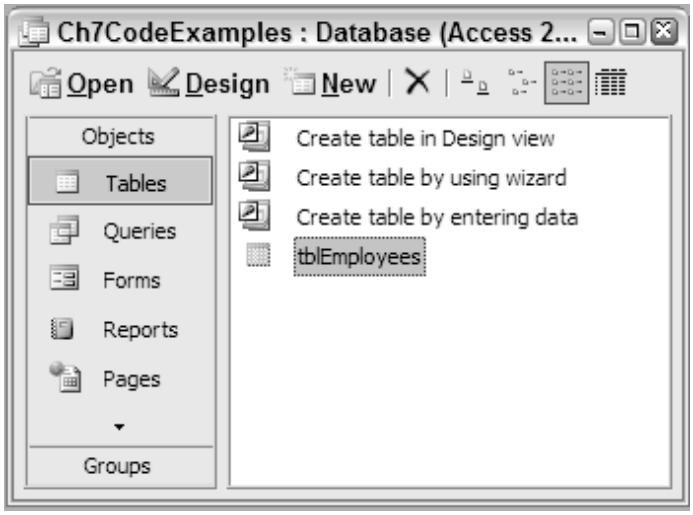

Figure 7.1

How It Works

In this example, you used the `TransferDatabase` method to import data from the Northwind sample database. The parameters of the `TransferDatabase` method specified the various bits of information Access needed to perform the import.

```
Sub TestTransferDatabase()

'import from Northwind
DoCmd.TransferDatabase acImport, "Microsoft Access", _
    "C:\Program Files\Microsoft Office\OFFICE11\SAMPLES\Northwind.mdb", _
            acTable, "Employees", "tblEmployees"

End Sub
```

After you ran the procedure , you should have noticed in the Database Window that the new table was inserted into your database.

Transferring Complete SQL Server Database

The `TransferSQLDatabase` method allows you to transfer an entire SQL Server database to another database. In effect, this method imports the entire SQL Server database into your Access database. Here is the basic syntax.

```
DoCmd.TransferSQLDatabase(Server, Database, UseTrustedConnection, Login,
Password, TransferCopyData)
```

Various parameters are used to specify how the method should execute. The following table explains the use of each parameter.

Parameter	Description
Server	Name of the SQL Server.
Database	Name of new database on specified SQL Server.
UseTrustedConnection	`True` if account has Administrator privileges, `False` otherwise and must specify `Login` and `Password`.
Login	Login name. Ignored if `UseTrustedConnection` is `True`.
Password	Login password. Ignored if `UseTrustedConnection` is `True`.
TransferCopyData	Use `True` to work with the data and schema and `False` to work with the schema only.

For example, to transfer the entire contents of a database called Pubs to the current database, you can use a command similar to the following.

```
DoCmd.TransferSQLDatabase _
    Server:="ServerName", _
    Database:="Pubs", _
    UseTrustedConnection:=True, _
    TransferCopyData:=False
```

Spreadsheets

The `TransferSpreadsheet` method is very similar to the `TransferDatabase` method in that it enables you to import, link, and export, only in this case it deals with spreadsheets. The syntax is shown in the following code.

```
DoCmd.TransferSpreadsheet(TransferType, SpreadsheetType, TableName,
FileName, HasFieldNames, Range, UseOA)
```

Various parameters are used to specify how the method should execute. The following table explains the use of each parameter.

Parameter	Description
TransferType	Type of transfer to be performed. Valid choices are `acImport` (default), `acLink`, and `acExport`.
SpreadsheetType	Type of spreadsheet. The default is `acSpreadsheetTypeExcel8`. See the help documentation for a complete list and explanation.
TableName	String expression that contains a table name or a SQL statement if you want to export data based on a SQL statement.
FileName	Filename and path of your spreadsheet.
HasFieldNames	Use `True` to use the first row of the spreadsheet as field names and `False` to treat the first row as data. `False` is the default.
Range	Valid range of cells or named range in the spreadsheet that you want to import from. Leave blank to import an entire spreadsheet. Using with Export will cause an error.
UseOA	Optional variant.

Now we'll walk through an example of how you might use the `TransferSpreadsheet` method to export data to a spreadsheet. Suppose you want to export the contents of the `Employees` table you just imported from Northwind into a spreadsheet so you can e-mail or send it to a colleague. The following code will create a new spreadsheet called `Employees.xls` in the temp directory.

```
DoCmd.TransferSpreadsheet acExport, acSpreadsheetTypeExcel9, _
    "tblEmployees", "C:\Temp\Employees.xls"
```

An example of the spreadsheet created from the preceding command might look like the one shown in Figure 7.2.

You can also use ADO to select, insert, update, and delete the underlying data in most of the data sources in this chapter by specifying the correct ADO provider (in this case Excel).

Text Files

The `TransferText` method allows you to import from, link to, and export to text files. It has the following syntax.

```
DoCmd.TransferText(TransferType, SpecificationName, TableName, FileName,
HasFieldNames, HTMLTableName, CodePage)
```

As you would expect, various parameters can be used to specify how the method should execute. These parameters are similar to the `TransferDatabase` and `TransferSpreadsheet` methods you have already seen. The following table explains the use of each parameter.

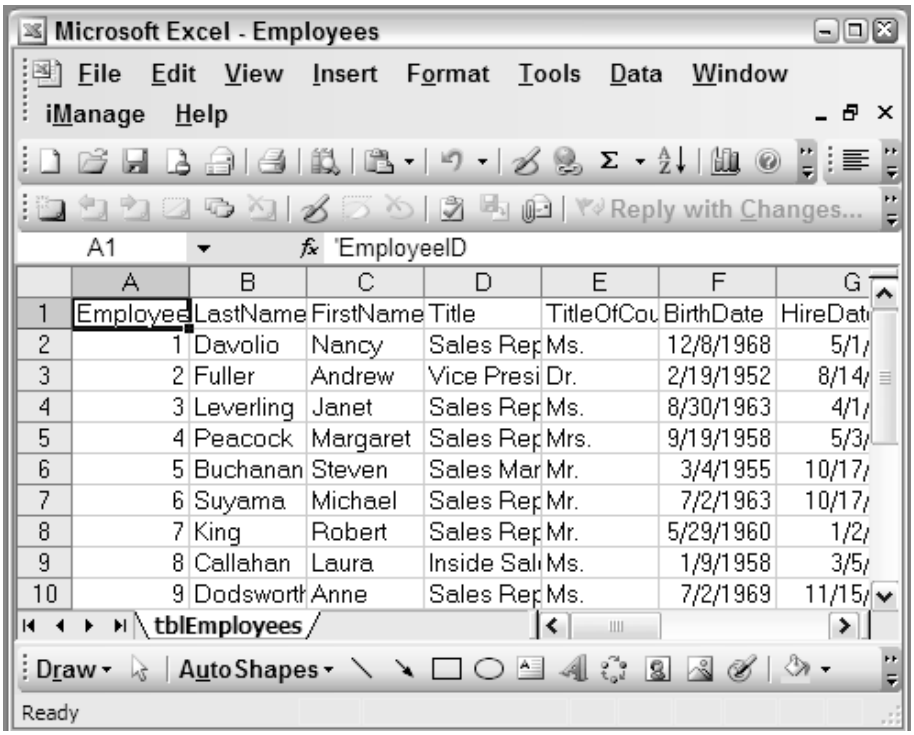

Figure 7.2

Parameter	Description
TransferType	Type of transfer to be performed. The default is `acImportDelim`. See the help documentation for a complete list and explanation.
SpecificationName	Name of import or export specification you have saved in the current database. This argument can be left blank for delimited text files.
TableName	String expression that contains a table name you want to work with or a query you want to export.
FileName	Filename and path of the text file you want to work with.
HasFieldNames	Use `True` to use the contents of first row of the spreadsheet as field names and `False` to treat the first row as data. `False` is the default.
HTMLTableName	Use with `acImportHTML` or `acLinkHTML`. Name of table or list in HTML file you want to work with. If blank, the first one is assumed.
CodePage	Long value indicating the character set of the code page.

Next, we'll jump right into importing data from a text file.

Try It Out Importing Data from a Text File

Now, you import data from a text file into a new table, called `tblEmails`, in your Access database.

1. Create a text file as shown in Figure 7.3 and save it in C:\temp.

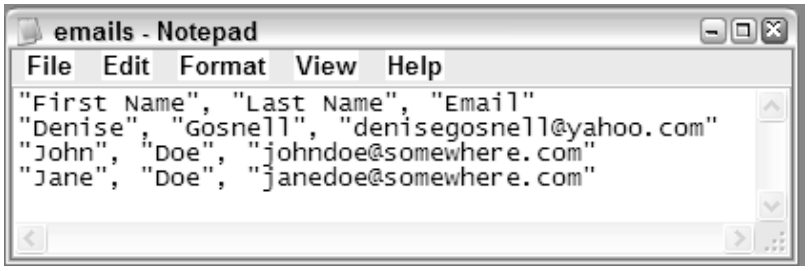

Figure 7.3

2. Add the following procedure to the module in your database.

```
Sub TestTransferText()

DoCmd.TransferText acImportDelim, , _
    "tblEmails", "C:\Temp\emails.txt", True

End Sub
```

3. Run the procedure from the Immediate Window in Visual Basic Editor.
4. Return to the database window and you should see a screen similar to that shown in Figure 7.4.

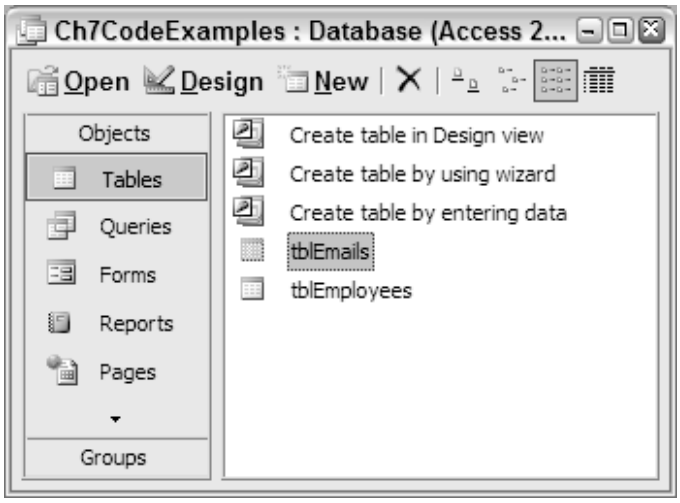

Figure 7.4

How It Works

First, you created a text file that contained comma-delimited records. You then created a procedure to import the comma-delimited file to your database.

```
Sub TestTransferText()

DoCmd.TransferText acImportDelim, , _
    "tblEmails", "C:\Temp\emails.txt", True

End Sub
```

The `TransferText` method imported the comma-delimited file into a new table called `tblEmails`, as shown in Figure 7.4. Note that the parameter for the `SpecificationName` was left blank because it is not required when working with delimited files.

XML Files

XML stands for eXtensible Markup Language. You have likely heard the XML buzzword, but you may not know what XML files really are. XML is a syntax that enables systems to create simple text documents with various tags that identify how the text should be interpreted. At the end of this section you will create an XML document from a table so you can see what one looks like.

The idea behind XML is to give various types of operating systems on different platforms a meaningful way of communicating with one another. As the use of XML has grown in popularity, the need to write Access applications that can import and export to XML text files has increased. Recent versions of Microsoft products incorporate extended XML functionality, including the `ImportXML` and `ExportXML` methods that Access provides to enable users to import from and export to XML databases. You will now look at each of these in turn.

The syntax for the `ImportXML` method is:

```
Application.ImportXML(DataSource, ImportOptions)
```

The `DataSource` is the name and path of the XML file to import. The `ImportOptions` parameter can be `acStructureAndData` (default), `acAppendData`, or `acStructureOnly`. Thus, to import an XML document into a table in your Access database, you might use the following code:

```
Application.ImportXML "employees.xml", acStructureAndData
```

The `ExportXML` method allows you to export data in your Access database to XML files to exchange data with other applications. Here is the syntax.

```
Application.ExportXML(ObjectType, DataSource, DataTarget, SchemaTarget,
PresentationTarget, ImageTarget, Encoding, OtherFlags, FilterCriteria,
AdditionalData)
```

The following table explains what the various parameters of the `ExportXML` object are used for.

Parameter	Description
ObjectType	The type of Access object to export.
DataSource	The name of the Access object to export.
DataTarget	The file name and path for the exported data.
SchemaTarget	The file name and path for the exported schema.
PresentationTarget	The file name and path for the exported target information.
ImageTarget	The path for exported images.
Encoding	The text encoding to use.
OtherFlags	Additional flags that can be used.
FilterCriteria	Subset of records to be exported.
AdditionalData	Additional tables to export.

I said earlier that I would provide a sample XML file so you could see what it looks like. Well, it's now time to use the `ExportXML` method to export one of your tables to XML so you can see how it works.

Suppose you have the following procedure:

```
Sub TestExportXML()

Application.ExportXML acExportTable, "tblEmployees", _
    "c:\Temp\Employees.xml", _
    "c:\Temp\EmployeesSchema.xml"

End Sub
```

The procedure uses the `ExportXML` method to export the `Employees` table in your database to an XML file called `Employees.xml`. After you run the preceding procedure, you create the XML file that looks similar to the XML file shown in Figure 7.5.

Notice in Figure 7.5 how the tags describe the data in detailed ways. This is a more detailed and structured way of organizing and describing data than HTML, which is just designed for displaying data.

That's all it takes to export data from your Access application to send to another system. You're now ready to learn how to send an e-mail from VBA code.

E-mails and Outlook

One way you can send an e-mail from VBA code is using the `SendObject` method, as shown below.

```
DoCmd.SendObject(ObjectType, ObjectName, OutputFormat, To, Cc, Bcc, Subject,
MessageText, EditMessage, TemplateFile)
```

Figure 7.5

The `ObjectType`, `ObjectName`, and `OutputFormat` parameters are used to specify a file created from the database to include as an attachment. Remember that earlier I said you exported the `tblEmployees` table to Excel so you could e-mail it to a co-worker. The `SendObject` method allows you to attach certain database objects in one of a variety of formats as part of the e-mail. Thus, to generate a new e-mail that also attaches the `tblEmployees` table as an Excel attachment, you could use something similar to the following:

```
'Send the Employees file
DoCmd.SendObject acSendTable, "tblEmployees", acFormatXLS, _
    "someone@yahoo.com", , , "Employee List", "For your review.", False
```

If you do not want to send an attachment, but just want to send an e-mail telling me how much you like the book so far, you can use the following command. Please do this—I would love to get this test e-mail from you!

```
'Send the author of this book an email
DoCmd.SendObject acSendNoObject, , , "denisegosnell@yahoo.com", , , _
      "This is cool!", _
      "I just sent an email from VBA. Really am enjoying your book.", False
```

If you want to learn more about controlling Outlook from your VBA applications, consult Chapter 10, where I cover automation with various Office programs such as Outlook. The `SendObject` method I just discussed is not specific to any e-mail program.

Other Ways to Export Data

Yet another way to export data can be implemented using the `OutputTo` method. The `OutputTo` method can export data to various formats: ASP, DAP, HTML, IIS, RTF, SNP, TXT, and XLS. The syntax for `OutputTo` is shown here.

```
DoCmd.OutputTo(ObjectType, ObjectName, OutputFormat, OutputFile, AutoStart,
TemplateFile, Encoding)
```

Please consult Chapter 8 for more information on the `OutputTo` method, as the entire chapter is dedicated to creating reports and Web-enabled output, including those created with the `OutputTo` method.

Using Data from Web Services

So far in this chapter, you have learned how to interact with the most common external data sources in your application. A common topic of discussion in the high-tech sector these days is Web services. *Web services* are reusable components that are based on standard Internet protocols. They enable systems on different platforms to talk to each other. In the simplest terms, I like to describe a Web service as some procedure or function that someone has made available over Internet protocols so you can use it for free or for a cost. Web services can be called (*consumed*) by Web and non-Web-based applications as long as the application can communicate using standard Internet protocols. Web services return a response in XML format that can then be used appropriately in the consuming program.

Although Web services communicate using Internet protocols, you can still use Web services for private purposes within a company. For example, some companies are better integrating legacy mainframe and other applications into their current environment by *exposing* parts of the old application as a Web service. After the old application is made accessible using a Web service, it can be called from other platforms that, traditionally, could not communicate as easily. Although I could write a whole chapter or even a book on Web services alone, I want you, at least, to learn the basic concept of what a Web service is. I also want to give you a simple example of how you can use someone else's Web service in your Access applications. The rest of this chapter will show you how to locate and consume Web services from Access.

Before you can consume a Web service, you must download and install the Microsoft Office 2003 Web Services Toolkit. At the time of this book's creation, the easiest way to obtain the toolkit is to go to: `http://msdn.microsoft.com/office/downloads/toolsutils/default.aspx` and then click the link to the Microsoft Office 2003 Web Services Toolkit 2.01.

If you cannot locate the toolkit at the preceding link, you can go to the downloads area of Microsoft.com: `http://www.microsoft.com/downloads`. Then, select the Office and Home Application Download Category in the left navigation pane and then run the following search: Select Office System for Product/Technology and type `Web Services Toolkit` in the Keywords text box. Click the Search button. You will see a link to the Microsoft Office 2003 Web Services Toolkit 2.01.

You can then follow the instructions on the page to download and run the setup program or just launch the setup directly from the Web site. Follow the on-screen instructions to complete the installation.

After you have the Web Services Toolkit installed, you can use Web services from your Access applications.

Try It Out Adding a Reference to a Web Service

Now, let's confirm that the Web Services Toolkit installation was successful and then add a reference to your Access project to an existing Web service.

1. Confirm that the Web Services Toolkit installation was successful. To do so, open the Visual Basic Editor and select Tools to confirm that you have a Web Services References option as shown in Figure 7.6.

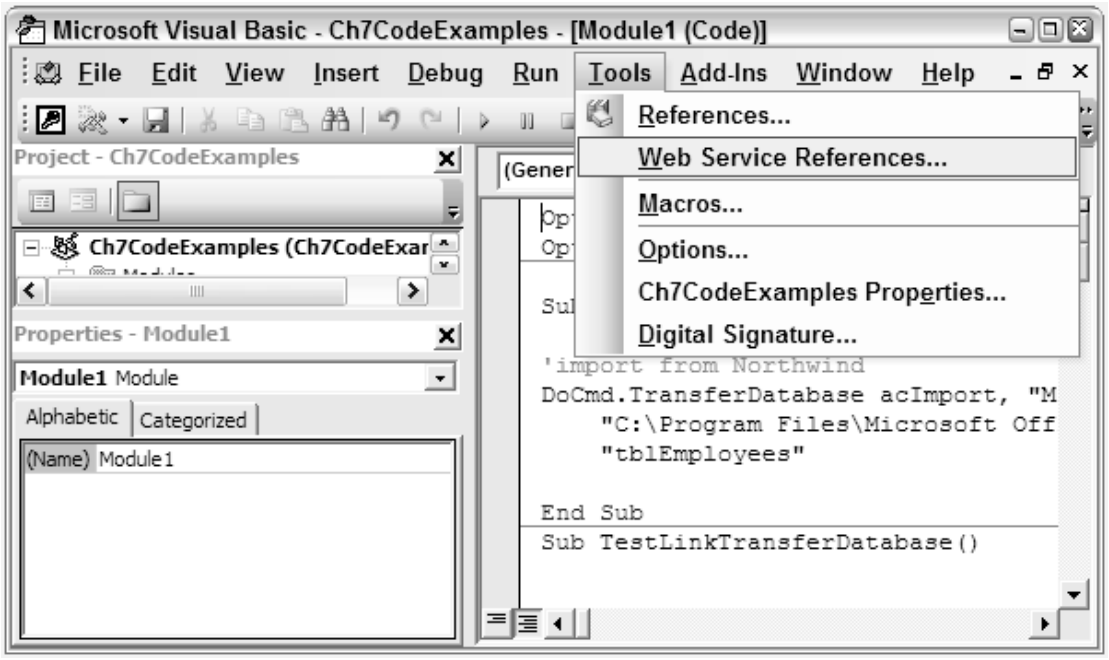

Figure 7.6

2. Select Tools ➪ Web Services References. Select the WebServiceURL option and type the following URL into the box: `http://terraserver-usa.com/TerraService2.asmx`. Click the Search button. A screen similar to that shown in Figure 7.7 will be displayed.

   You will need a connection to the Internet or the Web services examples shown here will not function.

3. Select the TerraService Web services in the upper-right pane so that a check box appears. Also, expand the list so you can see the procedures available, including the procedures shown in Figure 7.8.

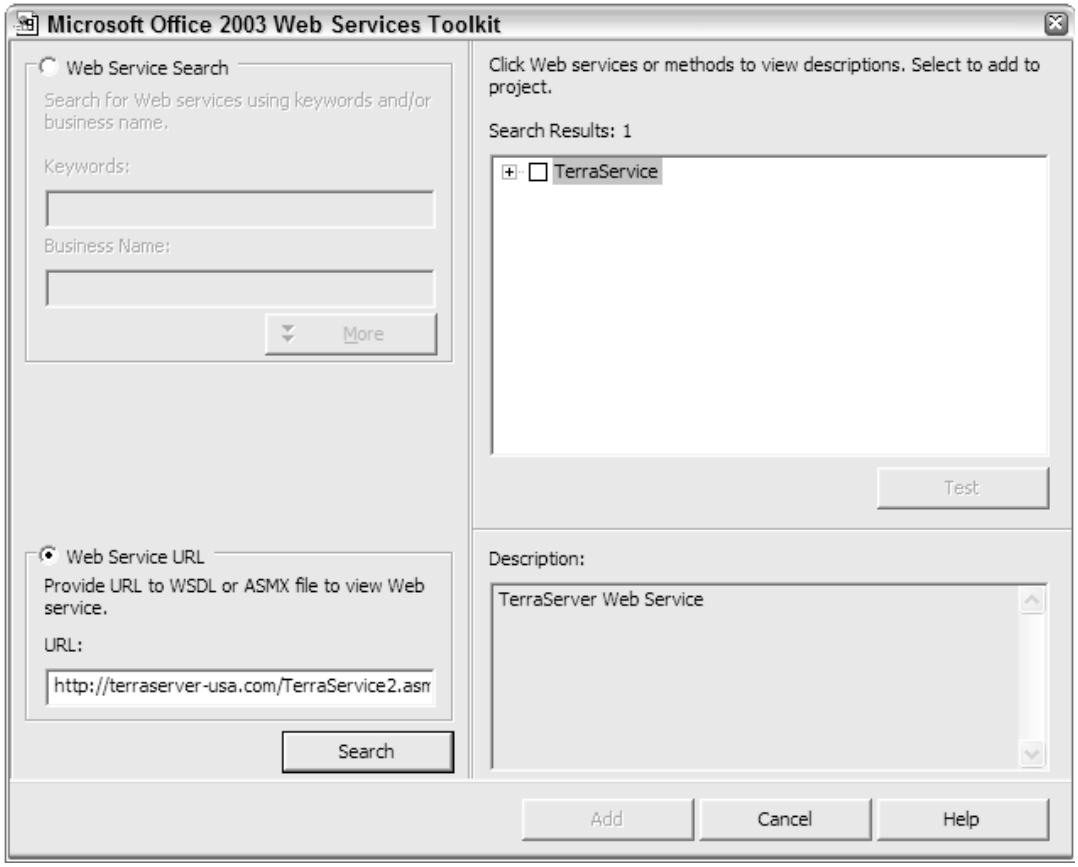

Figure 7.7

4. Select the Add button to add a reference to the Terra Web Service to your form.
5. When you return to the Visual Basic Editor, you can see that several class modules were added to your project, as shown in Figure 7.9.

How It Works

To add a reference to a Web service, you first selected the Tools ➪ Web Services References option. Because you're planning to use a service located at `http://terraserver-usa.com/TerraService2.asmx`, you selected the Web Service URL option and specified the Terra Server path in the URL field of Figure 7.7. You also could have searched for available Web services using certain keywords or specified the location of another Web service.

You then selected the Terra Server Web service and clicked the Add button. Several class modules were added to the database to make the procedures you saw in Figure 7.8 available to your Access project.

You can add a reference to any Web service that is available on your network over the Internet, or on your local computer, by navigating to its location. For example, Microsoft, Amazon, Google and others have created Web services that you can use. More information about these Web services can be found at:

Figure 7.8

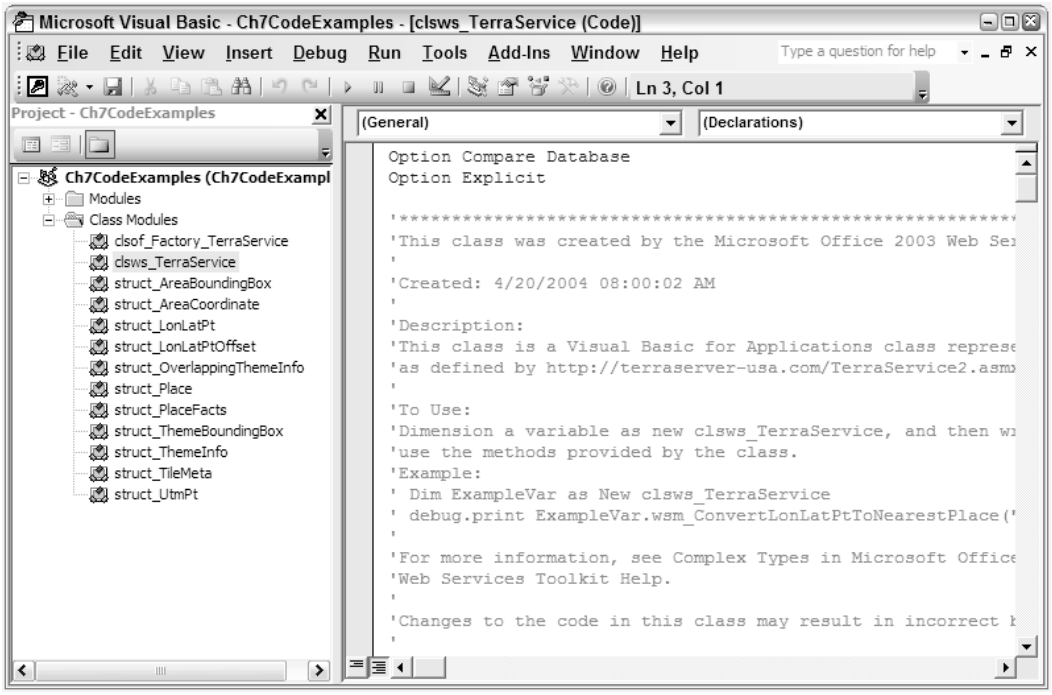

Figure 7.9

`http://msdn.microsoft.com/webservices/building/livewebservices/`. Many of these Web services, although they are free, require that you obtain an ID. Other companies may charge for use of Web services they offer.

Now that you have a reference to the Terra Server Web service, you can consume one of its procedures from within your application.

Try It Out Consuming the GetPlaceFacts Method

In this example, you will call one of the procedures in the Terra Server Web service. The Web service procedure you will call accepts latitude and longitude values and returns the city that is located at those coordinates.

1. Add the following procedure to your class module.

```
Sub TestWebService()

'declare a new instance of the web service
Dim ts As clsws_TerraService
Set ts = New clsws_TerraService

'declare a structure to hold the latitude and longitude values
Dim objLonLatPt As New struct_LonLatPt

'declare a variable to store the result from the web service
Dim strResult As String

'assign the latitude and longitude values
objLonLatPt.Lat = "37.7875671"
objLonLatPt.Lon = "-122.4276"

'Call the web service to return the place for that latitude
'and longitude
strResult = ts.wsm_ConvertLonLatPtToNearestPlace(objLonLatPt)

'display the result
MsgBox "The city at that latitude and longitude is: " & strResult

End Sub
```

2. Run the procedure from the Immediate Window by typing `TestWebService` and pressing Enter. You should receive a message similar to Figure 7.10.

Figure 7.10

How It Works

To call the Web service from your application, you created the `TestWebService` procedure.

```
Sub TestWebService()
```

You declare a new instance of the Web service, just as you would declare any other object.

```
'declare a new instance of the web service
Dim ts As clsws_TerraService
Set ts = New clsws_TerraService
```

You declare a structure variable to store the latitude and longitude values. You created the structure definition in the code that was generated when adding the Web service, so all you had to do was declare the variable.

```
'declare a structure to hold the latitude and longitude values
Dim objLonLatPt As New struct_LonLatPt
```

Next, you declared a local string variable to store the result of the Web service.

```
'declare a variable to store the result from the web service
Dim strResult As String
```

You also set the latitude and longitude parameters of the `objLonLatPt` that will be passed to the Web service as parameters.

```
'assign the latitude and longitude values
objLonLatPt.Lat = "37.7875671"
objLonLatPt.Lon = "-122.4276"
```

This line below actually calls the Web service on the Terra Server over the Internet and returns the result into the `strResult` variable. If you do not have an Internet connection, this line will raise an error.

```
'Call the web service to return the place for that latitude
'and longitude
strResult = ts.wsm_ConvertLonLatPtToNearestPlace(objLonLatPt)
```

As you typed the line of code, you should have noticed that the tooltip displayed to show you what parameters the Web service required. Figure 7.11 show an example of this.

The last line of code displays the result that was retrieved from the Web service.

```
'display the result
MsgBox "The city at that latitude and longitude is: " & strResult

End Sub
```

When running the procedure from the Immediate Window, you receive a dialog box like Figure 7.10 that indicates that San Francisco, California, USA is located at those coordinates. It is really amazing that you have just executed a remote procedure that was located on someone else's computer and used the result in the application. Hopefully, you realize that it is almost as easy to call a Web service as it is to call any other procedure. The only difference is that you have to locate and add a reference to the Web service and familiarize yourself with the parameters it expects.

```
Sub TestWebService()

'declare a new instance of the web service
Dim ts As clsws_TerraService
Set ts = New clsws_TerraService

'declare a structure to hold the latitude and longitude values
Dim objLonLatPt As New struct_LonLatPt

'declare a variable to store the result from the web service
Dim strResult As String

'assign the latitude and longitude values
objLonLatPt.Lat = "37.7875671"
objLonLatPt.Lon = "-122.4276"

'Call the web service to return the place for that latitude
'and longitude
strResult = ts.wsm_ConvertLonLatPtToNearestPlace(objLonLatPt)
                wsm_ConvertLonLatPtToNearestPlace(ByVal obj_point As struct_LonLatPt) As String
'display the result
MsgBox "The city at that latitude and longitude is: " & strResult


End Sub
```

Figure 7.11

Summary

In today's world of technology, it has become commonplace for companies to expect systems to communicate with each other in ways that were not possible just a decade ago. In this chapter, you explored ways to work with external data to facilitate communications between various systems. You learned VBA techniques for linking, importing, and exporting data to various sources, such as databases, spreadsheets, and text files. You also learned how to handle more advanced scenarios, such as sending e-mails programmatically and consuming Web services from your application. Web services allow applications on various platforms to exchange data upon request and have become an increasingly popular communication technique between systems.

Chapter 8 will build on this chapter by explaining how to export data in your application to reports and Web-enabled output.

Exercises

1. What are some reasons you would choose linking versus importing?
2. What is the difference between the `TransferDatabase` and `TransferSQLDatabase` methods?
3. What is a Web service and how do you use one?