

**Module 4:**  
**Using Data in Windows**  
**Forms Applications**

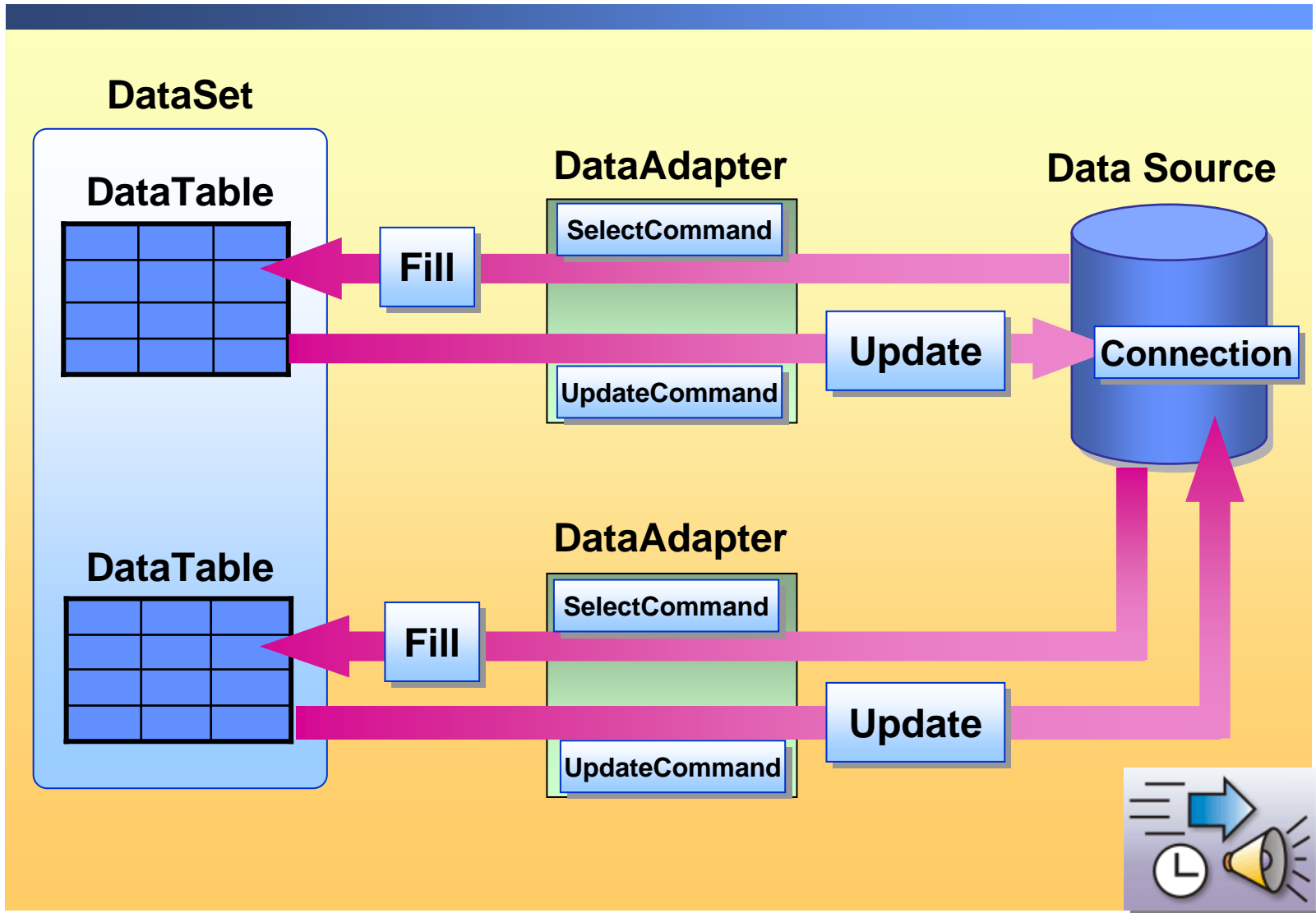
# Overview

- Adding ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application
- Accessing and Modifying Data by Using DataSets
- Binding Data to Controls
- Overview of XML Web Services
- Creating a Simple XML Web Services Client
- Persisting Data

# Lesson: Adding ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application

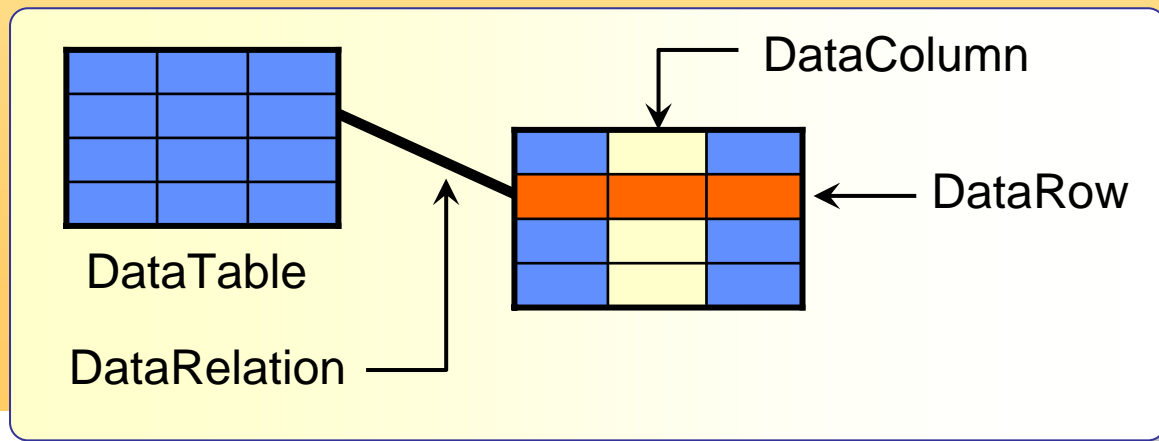
- ADO.NET Objects
- What Is a DataSet?
- What Is a Typed DataSet?
- How to Add ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application
- Practice: Adding ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application

# ADO.NET Objects



# What Is a DataSet?

- Datasets can include multiple DataTables
- Relationships between tables are represented using DataRelations
- Constraints enforce primary and foreign keys
- Use the DataRow and DataColumn to access values in Tables



# What Is a Typed Dataset?

## ■ Typed datasets

- Derive from the base **DataSet** class
- Provide type checking at compile time
- Provide faster access to tables and columns in the dataset
- Generated from XML Schema (.xsd) files by using the XSD.exe tool

## ■ To access tables and columns

- Untyped dataset

```
pubsDataSet.Tables("Titles")
```

- Typed dataset

```
pubsDataSet.Titles
```

# How to Add ADO.NET Objects to and Configure ADO.NET Objects in a Windows Forms Application

- 1** Drag an **OleDbDataAdapter** or **SqlDataAdapter** object from the Toolbox to a form
- 2** Specify connection and SQL command information
- 3** Select the adapter or adapters that will be used to transfer data between the data source and the dataset
- 4** On the **Data** menu, choose **Generate Dataset**
- 5** Select **New** and then specify a name for the new dataset

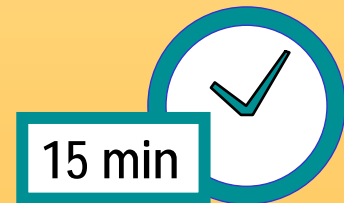
# Practice: Adding ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application



In this practice, you will

- Add and configure a **SqlConnection** object on a Windows Form
- Add and configure a **SQLDataAdapter** control on a Windows Form
- Generate the dataset

Begin reviewing the objectives  
for this practice activity



# Lesson: Accessing and Modifying Data by Using DataSets

- How to Populate a Dataset
- How to Update Data in a Dataset
- How to Update Data to a Data Source
- Practice: Populating and Updating DataSets
- How to Create Database Schema on the Client
- Demonstration: Creating Database Schema by Using the XML Schema Designer
- How to Read and Write XML Data into a DataSet

# How to Populate a Dataset

- Use the DataAdapter object to fill the dataset

```
Dim storesSQLDataAdapter As New SqlConnection.SqlDataAdapter()  
Dim storesSelectSqlCommand As New SqlConnection.SqlCommand()  
storesSelectSqlCommand.CommandText = "SELECT * FROM stores"  
storesSelectSqlCommand.Connection = SqlConnection1  
storesSQLDataAdapter.SelectCommand = storesSelectSqlCommand  
  
storesSQLDataAdapter.Fill(storesDataSet.Tables("Stores"))
```

# How to Update Data in a Dataset

## ■ Adding rows

```
Dim newDataRow As DataRow = _  
    pubsDataSet.Tables("Titles").NewRow  
newDataRow ("title") = "New Book"  
newDataRow ("type") = "business"  
pubsDataSet.Tables("Titles").Rows.Add(newDataRow)
```

## ■ Editing rows

```
changeDataRow.BeginEdit()  
changeDataRow("Title") = changeDataRow("Title").ToString & _  
" 1"  
changeDataRow.EndEdit()
```

## ■ Deleting data

```
Dim deleteDataRow As DataRow = _  
    pubsDataSet.Tables("Titles").Rows(0)  
pubsDataSet.Tables("Titles").Rows.Remove(deleteDataRow)
```

# How to Update Data to a Data Source

- Explicitly specifying the updates

```
Dim insertTitlesCommand As New SqlConnection.SqlCommand _
("Insert titles (title_id, title, type) values " & _
"@title_id,@title,@type)")

insertTitlesCommand.Parameters.Add("@title_id", _
SqlConnectionType.VarChar, 6, "title_id")

insertTitlesCommand.Parameters.Add("@title", _
SqlConnectionType.VarChar, 80, "title")

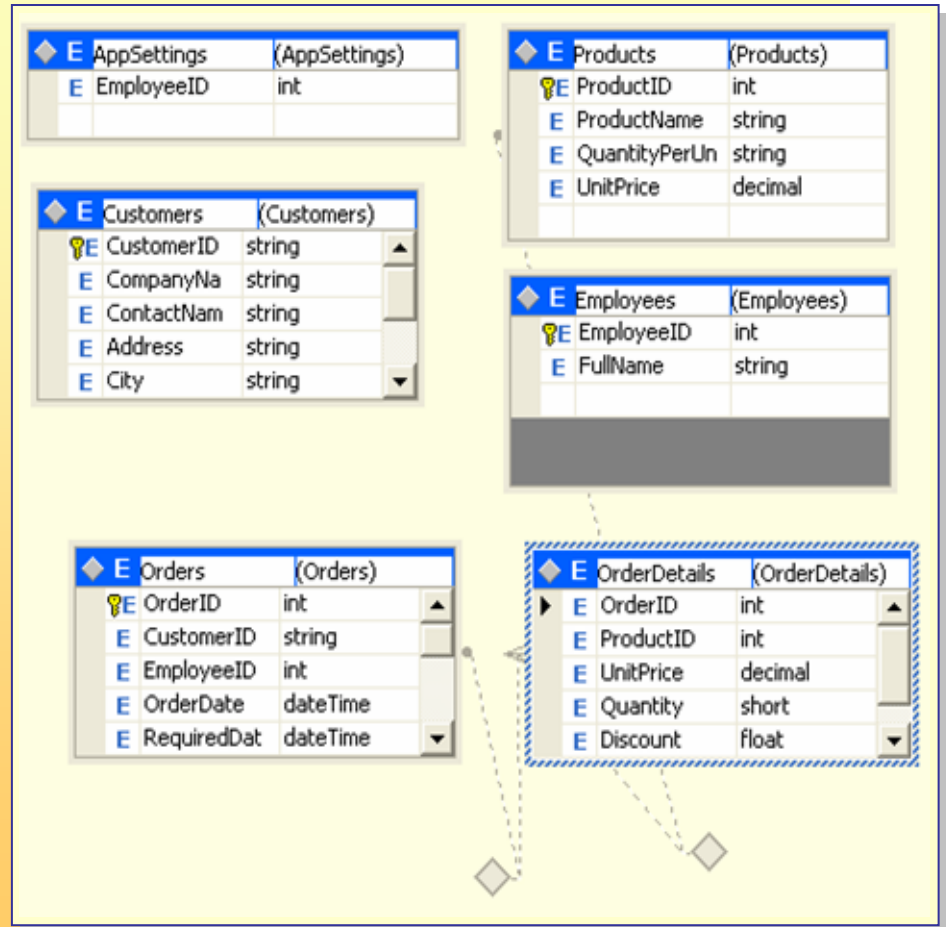
insertTitlesCommand.Parameters.Add("@type", _
SqlConnectionType.Char, 12, "type")

titlesSQLDataAdapter.InsertCommand = insertTitlesCommand
titlesSQLDataAdapter.Update(pubsDataSet, "titles")
```

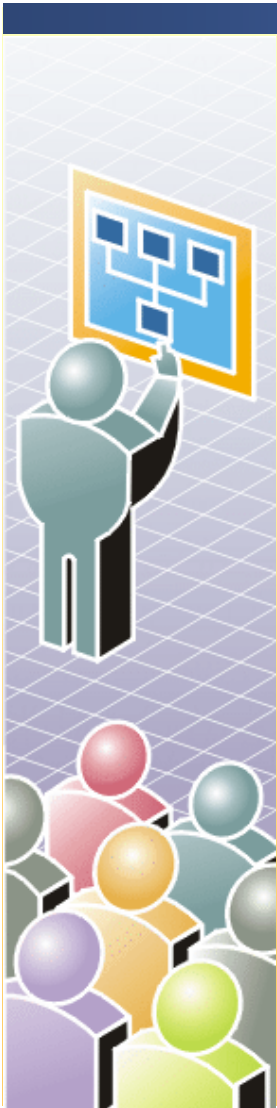


# How to Create Database Schema on the Client

- XML Schema files (.xsd) enforce data integrity on the client
- Use the XML Designer to create and modify XML Schema files
  1. Determine the schema design
  2. On the **Project** menu, click **Add New Item**
  3. Add the schema
  4. Create the schema



# Demonstration: Creating Database Schema by Using the XML Schema Designer



In this demonstration, you will see how to create a `DataRelation` between tables in a `DataSet`

# How to Read and Write XML Data into a DataSet

- Use ReadXML to load data from a file or stream

```
purchaseDataSet.ReadXml _  
    ("C:\sampledata\PurchaseData.xml", _  
     XmlReadMode.IgnoreSchema)
```

- Write data and schema information from a DataSet to a file or stream by using the WriteXML method

```
("C:\sampledata\CurrentOrders.xml", _  
     XmlWriteMode.IgnoreSchema)
```

# Lesson: Binding Data to Controls

- How to Perform Simple Binding by Using the DataBindings Property
- How to Perform Complex Data Binding by Using the DataBound Windows Forms Controls
- Practice: Binding Controls to Data
- How to Maintain the Currency of a Control by Using CurrencyManager
- Demonstration: Maintaining the Currency of a Control by Using CurrencyManager
- How to Format and Parse Data Bound Values
- Practice: Formatting Data Bound Controls

# How to Perform Simple Binding by Using the DataBindings Property

To use the DataBindings Collection to bind a control to a data source, set the DataBinding property of the control to the data source

Property of the control to which data is bound

```
txtCustomerAddress.DataBindings.Add("Text",  
    dsNorthwindData1.Customers, "Address")  
txtCustomerCity.DataBindings.Add("Text",  
    dsNorthwindData1.Customers, "City")
```

Table from the data source

Column in the table

# How to Perform Complex Data Binding by Using the DataBound Windows Forms Controls

- **Complex data binding**
  - Bind a control property to a data table
  - Use with combo boxes, list boxes, data grids
  - Can bind controls to DataSets, Arrays, and ArrayLists
- **Complex databinding at design time**
  - Set the **DataSource** and **DataMember** properties
- **Complex databinding at run time**

```
DataGrid1.DataSource = productsDataSet  
DataGrid1.DataMember = "Products"
```

# Practice: Binding Controls to Data



## In this practice, you will

- Configure the **SQLConnection** control to connect to the database
- Bind controls to columns in the dataset at design time
- Bind controls to columns in the dataset at run time

Begin reviewing the objectives  
for this practice activity



# How to Maintain the Currency of a Control by Using CurrencyManager

Blue	Yellow	Blue
Orange	Orange	Orange
Blue	Yellow	Blue
Blue	Yellow	Blue

Data Source 1

Currency Manager1

TextBox1

TextBox2

Blue	Yellow	Blue
Orange	Orange	Orange
Blue	Yellow	Blue
Blue	Yellow	Blue

Currency Manager2

Blue	Blue	Blue
Blue	Blue	Blue
Blue	Blue	Blue
Blue	Blue	Blue

Datagrid

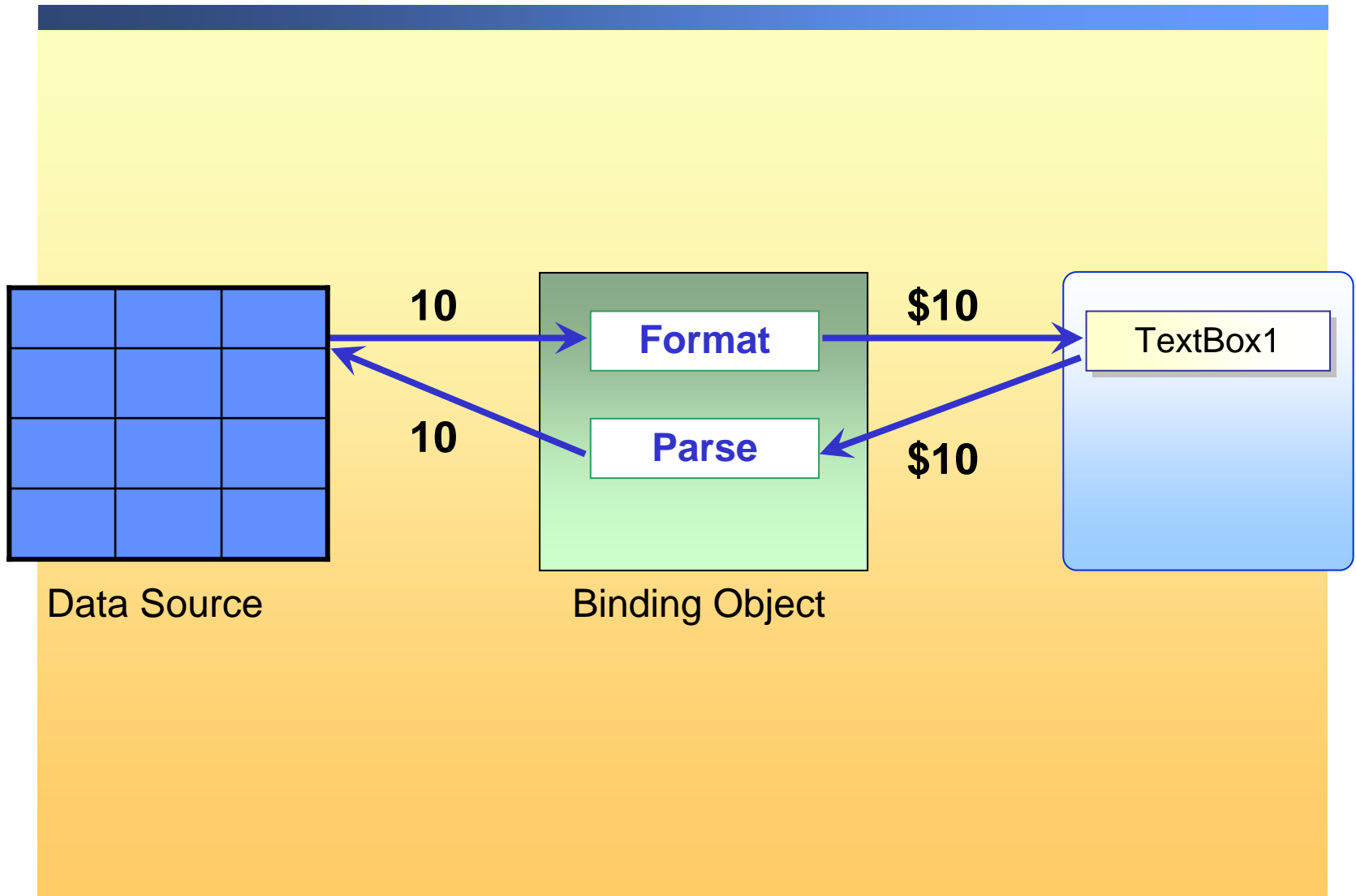
```
Dim cm As CurrencyManager  
cm = Me.BindingContext(pubsDataSet, "Authors")  
cm.Position += 1
```

# Demonstration: Maintaining the Currency of a Control by Using CurrencyManager



In this demonstration, you will see how to maintain the currency of a control by using CurrencyManager

# How to Format and Parse Data Bound Values



# Practice: Formatting Data Bound Controls



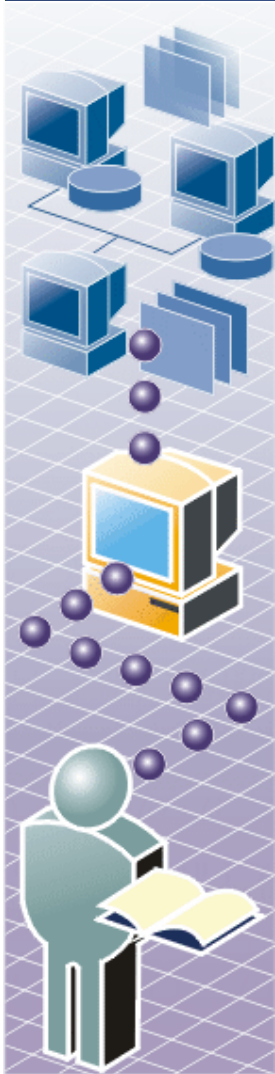
In this practice, you will

- Configure the **SqlConnection** control to connect to the database
- Create an event procedure for the **Parse** event
- Create an event delegate for the **Parse** event

Begin reviewing the objectives  
for this practice activity

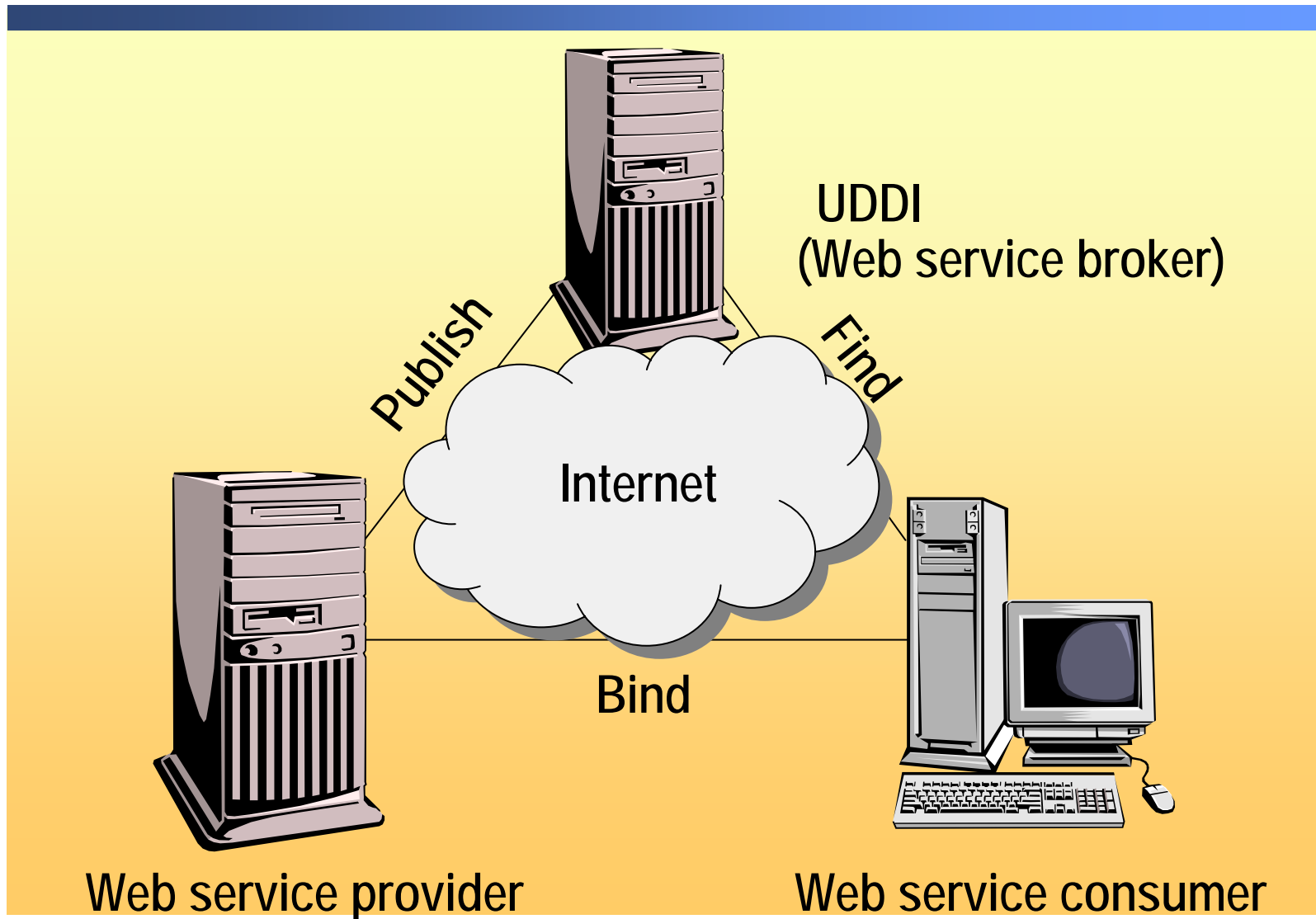


# Lab 4.1: Accessing Data by Using ADO .NET



- Exercise 1: Generating and Populating DataSets
- Exercise 2: Modifying a DataSet
- Exercise 3: Updating a DataSet to a Data Source
- Exercise 4: Binding and Formatting Data in Controls

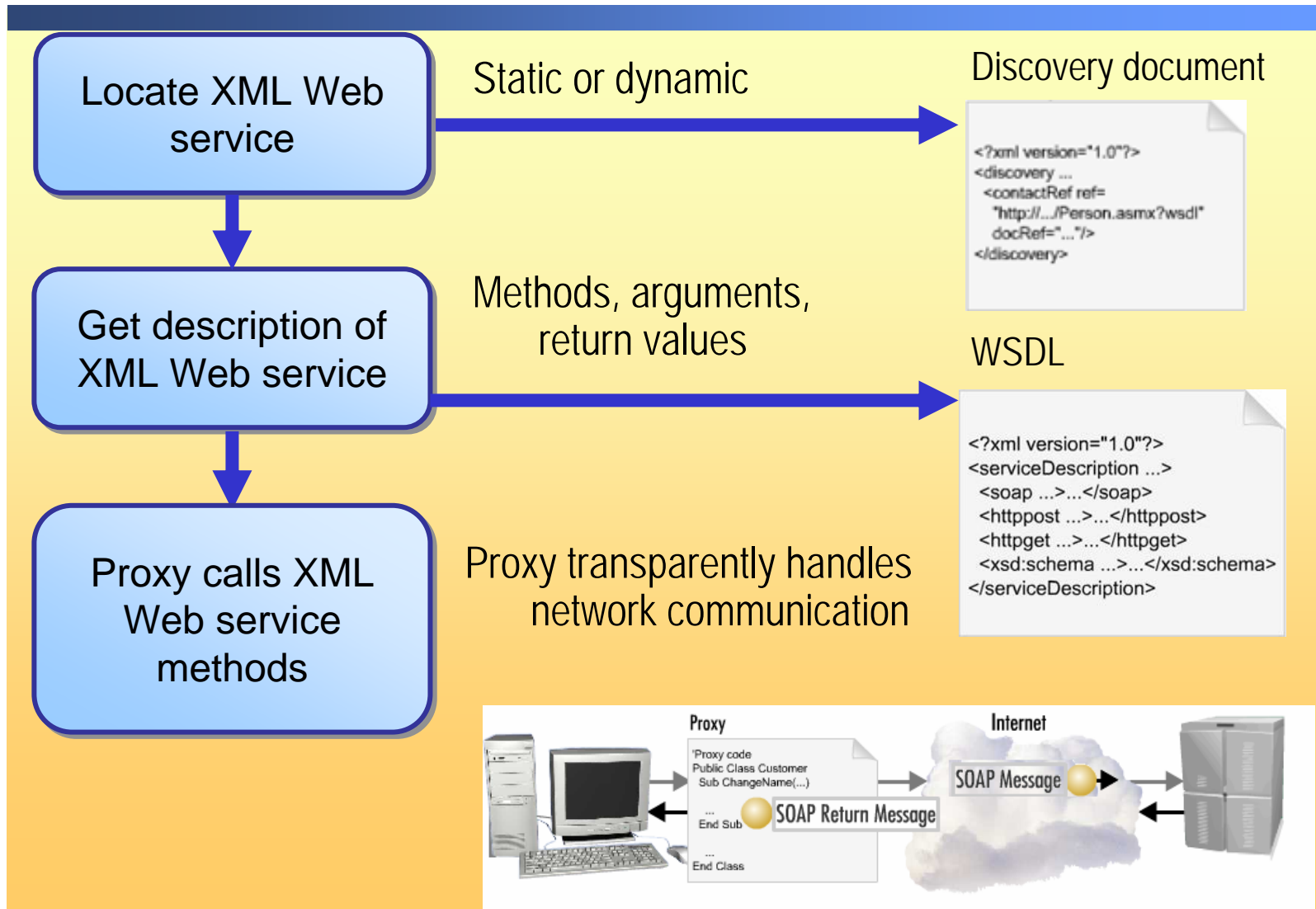
# Lesson: Overview of XML Web Services



# What Are XML Web Services?

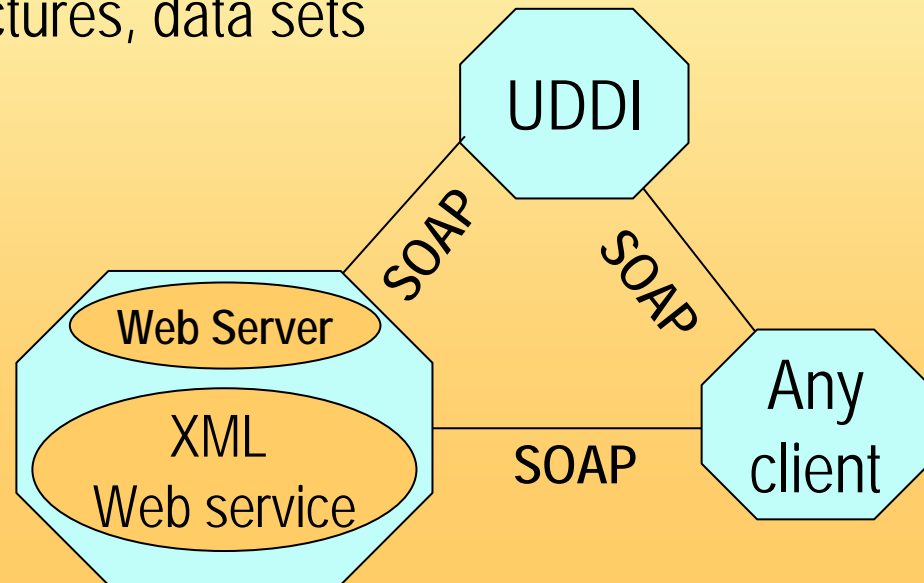
- An XML Web service is a URL-addressable set of functions that is exposed over a network to serve as a building block for creating distributed applications
  - Based on Internet technologies, such as HTTP, XML and SOAP
  - Building blocks
- Basic elements
  - XML Web service provider
  - XML Web service consumer
  - XML Web service broker

# How Do XML Web Services Work?



# What is SOAP?

- SOAP is a lightweight XML-based protocol for exchange of information in decentralized, distributed environments
- Provides enhanced features such as the ability to
  - Pass by reference
  - Pass objects, structures, data sets



# What is WSDL?

- **WSDL**

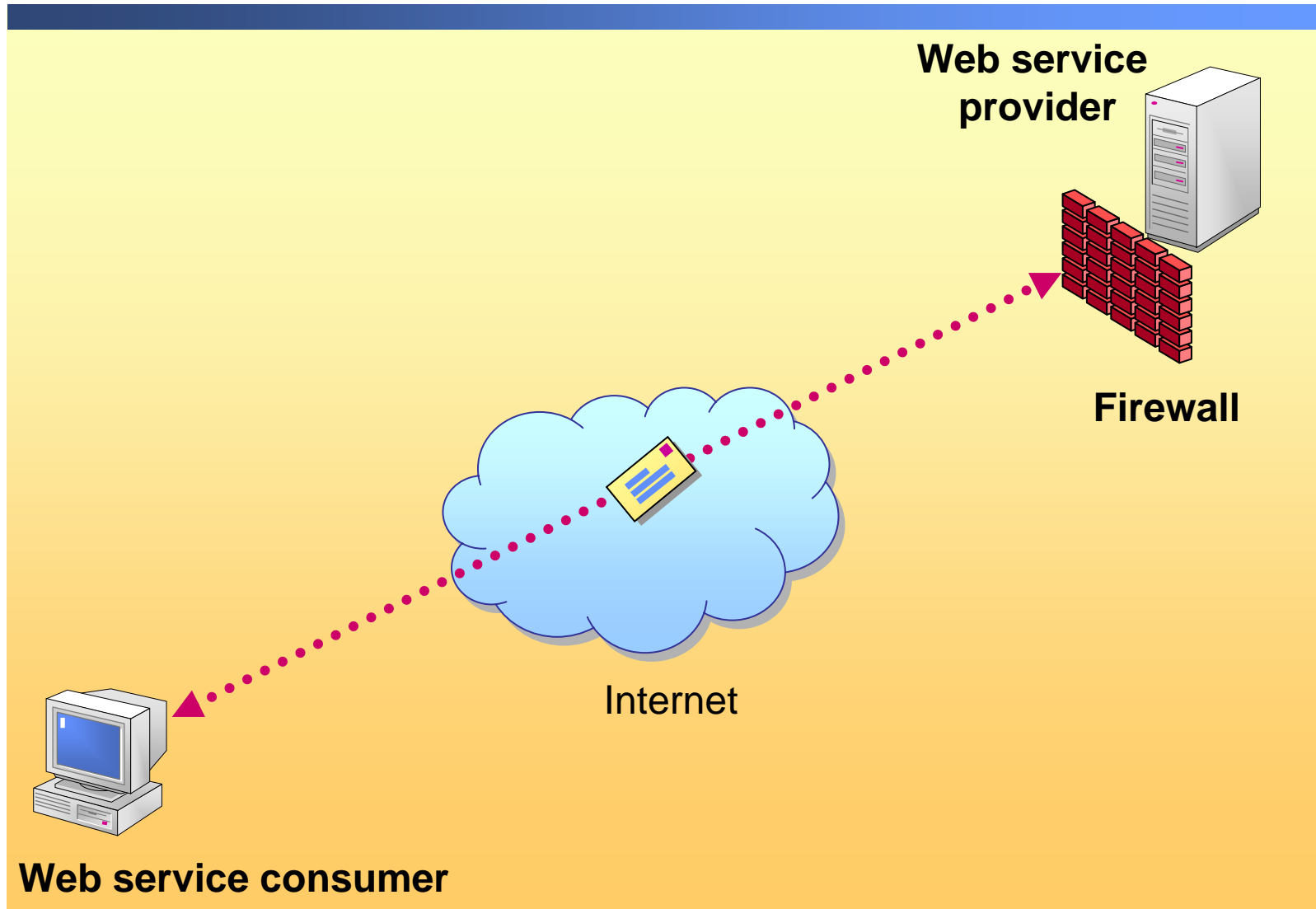
An XML grammar used for describing a Web service in terms of the messages it accepts and generates

- **WSDL document**

Defines the types used in the operations (methods) of a Web service and the documents that are exchanged for each operation

[CodeExample](#)

# Lesson: Creating a Simple XML Web Services Client



# How to Locate an XML Web service

- **Static Discovery**
  - Provide explicit URL
  - Disco file
- **Dynamic Discovery**
  - UDDI
  - Vsdisco file

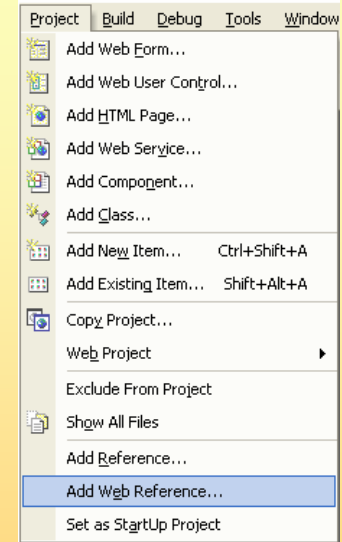
# How to Access an XML Web Service

## 1 Generate the proxy

- Add Web reference
  - Or -
- Run WSDL.exe

## 2 Instantiate the proxy

## 3 Call XML Web service methods



```
' Inside a button click event
Dim WS As New ExpenseReportWebService()
Dim ds As DataSet = WS.GetReportsForEmployee(username, _
    passwordToken, 0, 10, totalReports)
```

Instantiate the proxy

Call XML Web service method

# How to Test an XML Web Services Client

- 1** Build the client application
- 2** Call the XML Web service installed on <http://localhost>
- 3** Handle exceptions thrown by the XML Web service

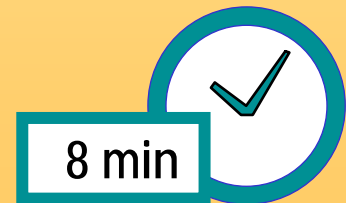
# Practice: Calling an XML Web Service



## In this practice, you will

- Add a call to an XML Web service to a client application
- Use a provided XML Web service to test your application

Begin reviewing the objectives  
for this practice activity



# Lesson: Persisting Data

- How to Persist Data in Files
- How to Serialize Objects
- How to Use Isolated Storage
- How to Persist Application Settings

# How to Persist Data in Files

- Use readers and writers for persisted data

Class	Description
BinaryReader and BinaryWriter	These classes read and write primitive types as binary values in a specific encoding to and from a stream.
StreamReader and StreamWriter	The implementations of these classes are designed for character input and output.
StringReader and StringWriter	The implementations of these classes are designed for string input and output.

# How to Serialize Objects

- Binary serialization

- 1 Add the **Serializable** attribute

- 2 Serialize the object to a file

- 3 Deserialize the file into an object

- XML serialization

- 1 Serialize the object to a file

- 2 Deserialize the file into an object

# How to Use Isolated Storage

- For scenarios that require higher security, such as the Internet, you should read and write data to isolated storage
- To access isolated storage

**1** Open the store

**2** Create a stream for reading or writing files in the store

**3** Close the stream and store

[CodeExample](#)

# How to Persist Application Settings

## ■ Choose a technique

- Use a DataSet object

Good for tabular or relational data

- Use reader/writer objects

Complete control, but developer must write and maintain more code

- Use serialization

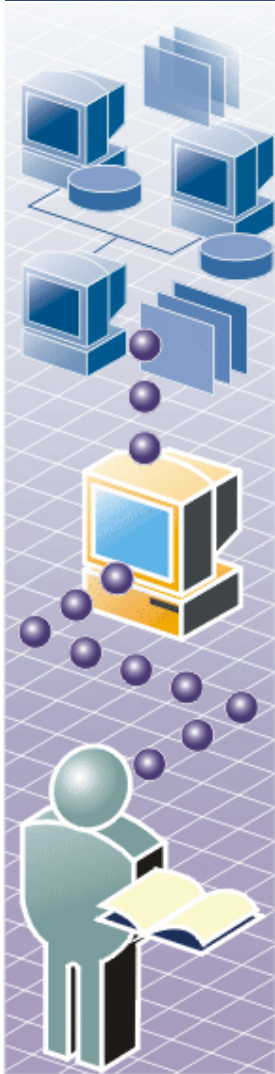
Good choice when application stores state in objects

## ■ Choose a storage location

- The file system

- Isolated storage

# Lab 4.2: Calling an XML Web Service



- Exercise 1: Calling an XML Web Service

# Review

- Adding ADO.NET Objects to and Configuring ADO.NET Objects in a Windows Forms Application
- Accessing and Modifying Data by Using DataSets
- Binding Data to Controls
- Overview of XML Web Services
- Creating a Simple XML Web Services Client
- Persisting Data