

Module 8: Using ADO.NET

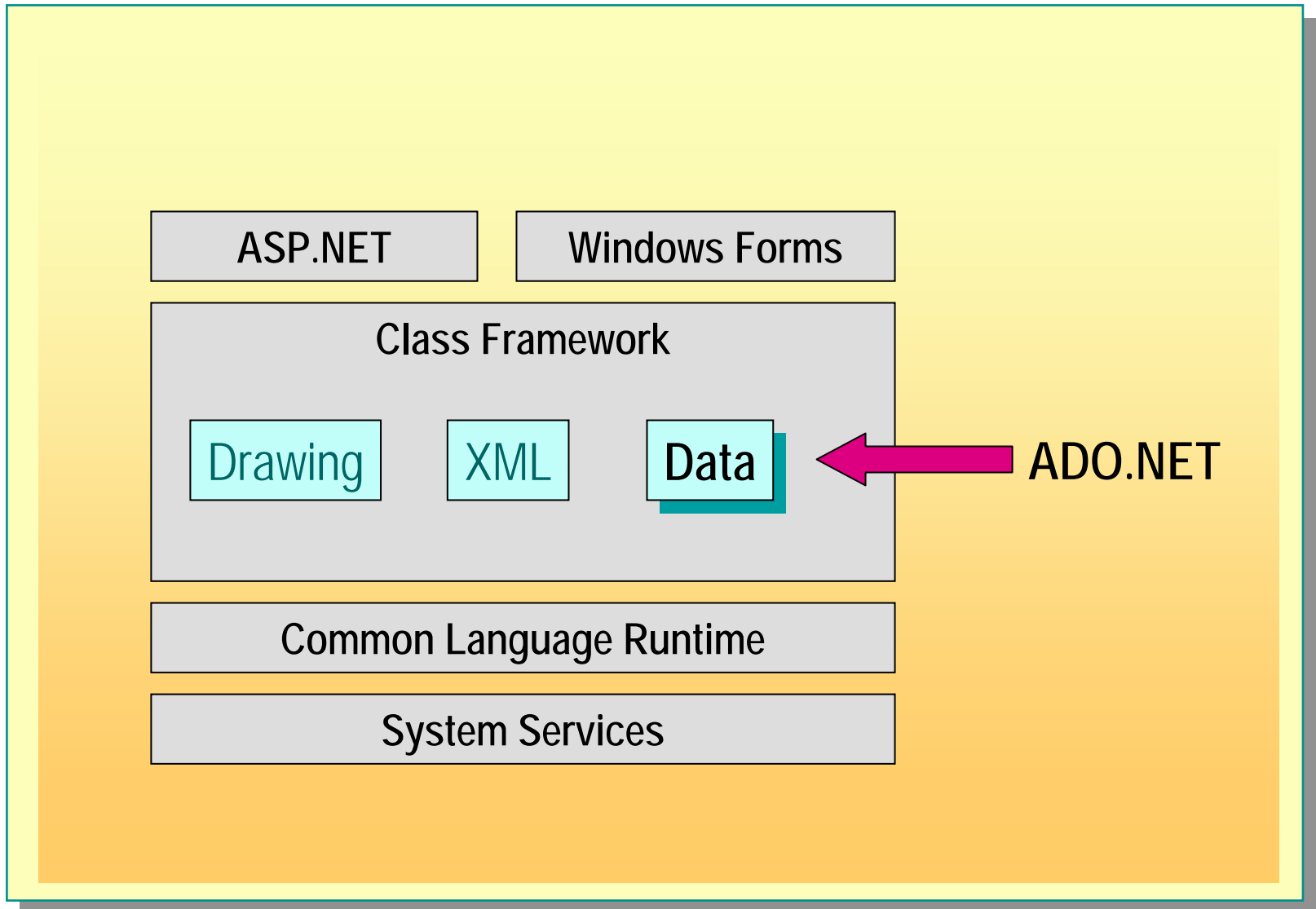
Overview

- ADO.NET Overview
- .NET Data Providers
- The DataSet Object
- Data Designers and Data Binding
- XML Integration

◆ ADO.NET Overview

- Introduction to ADO.NET
- Benefits of ADO.NET

Introduction to ADO.NET



Benefits of ADO.NET

- Similar to ADO
- Designed for disconnected data
- Intrinsic to the .NET Framework
- Supports XML

◆ .NET Data Providers

- Using the Connection Object
- Using the Command Object
- Using the Command Object with Stored Procedures
- Using the DataReader Object
- Using the DataAdapter Object

Using the Connection Object

- Connecting from a Web application
 - Add a line to Web.config file

```
<identity impersonate="true" userName="Karen"  
password="Password"/>
```

■ SqlConnection

```
Dim conSQL As New SqlClient.SqlConnection( )  
conSQL.ConnectionString = "Integrated Security=True;" & _  
    "Data Source=localhost;Initial Catalog=Pubs;"  
conSQL.Open( )
```

Using the Command Object

- Two ways to create a Command:

- Command constructor
- CreateCommand method

- Four ways to execute a Command:

- ExecuteReader
- ExecuteScalar
- ExecuteNonQuery
- ExecuteXMLReader

```
Dim commSQL As New SqlConnection( )  
commSQL.Connection = conSQL  
commSQL.CommandText = "Select Count(*) from Authors"  
MessageBox.Show(commSQL.ExecuteScalar( ).ToString)
```

Using the Command Object with Stored Procedures

1. Create a Command object
2. Set the CommandType to StoredProcedure
3. Set the CommandText property
4. Use the Add method to create and set parameters
5. Use the ParameterDirection property
6. Call ExecuteReader
7. Use records, and then close DataReader
8. Access output and return parameters

Using the DataReader Object

- Reading data

```
Dim commSQL As New SqlConnection.SqlCommand( )
commSQL.Connection = conSQL
commSQL.CommandText = "Select au_lname, au_fname from authors"
Dim datRead As SqlConnection.SqlDataReader
datRead = commSQL.ExecuteReader( )
Do Until datRead.Read = False
    MessageBox.Show(datRead.GetString(1) & " " _
        & datRead.GetString(0))
Loop
datRead.Close( )
```

- Retrieving data
- Returning multiple result sets

Using the DataAdapter Object

- Used as a link between data source and cached tables

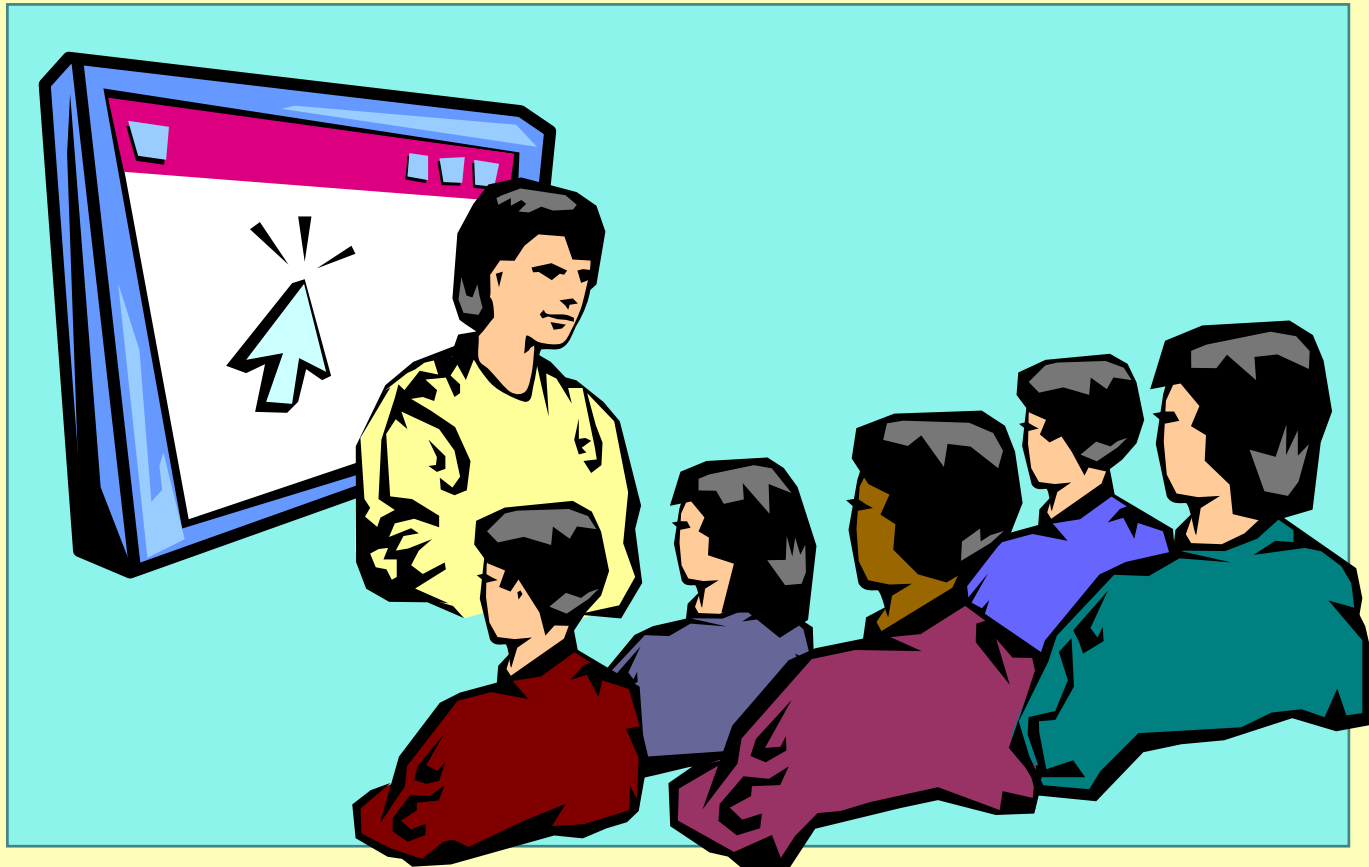
```
Dim adaptSQL As New SqlConnection.SqlDataAdapter( _  
    "Select * from authors", conSQL)
```

```
Dim datPubs As New DataSet( )  
adaptSQL.Fill(datPubs, "NewTable")
```

```
' Manipulate the data locally
```

```
adaptSQL.Update (datPubs, "NewTable")
```

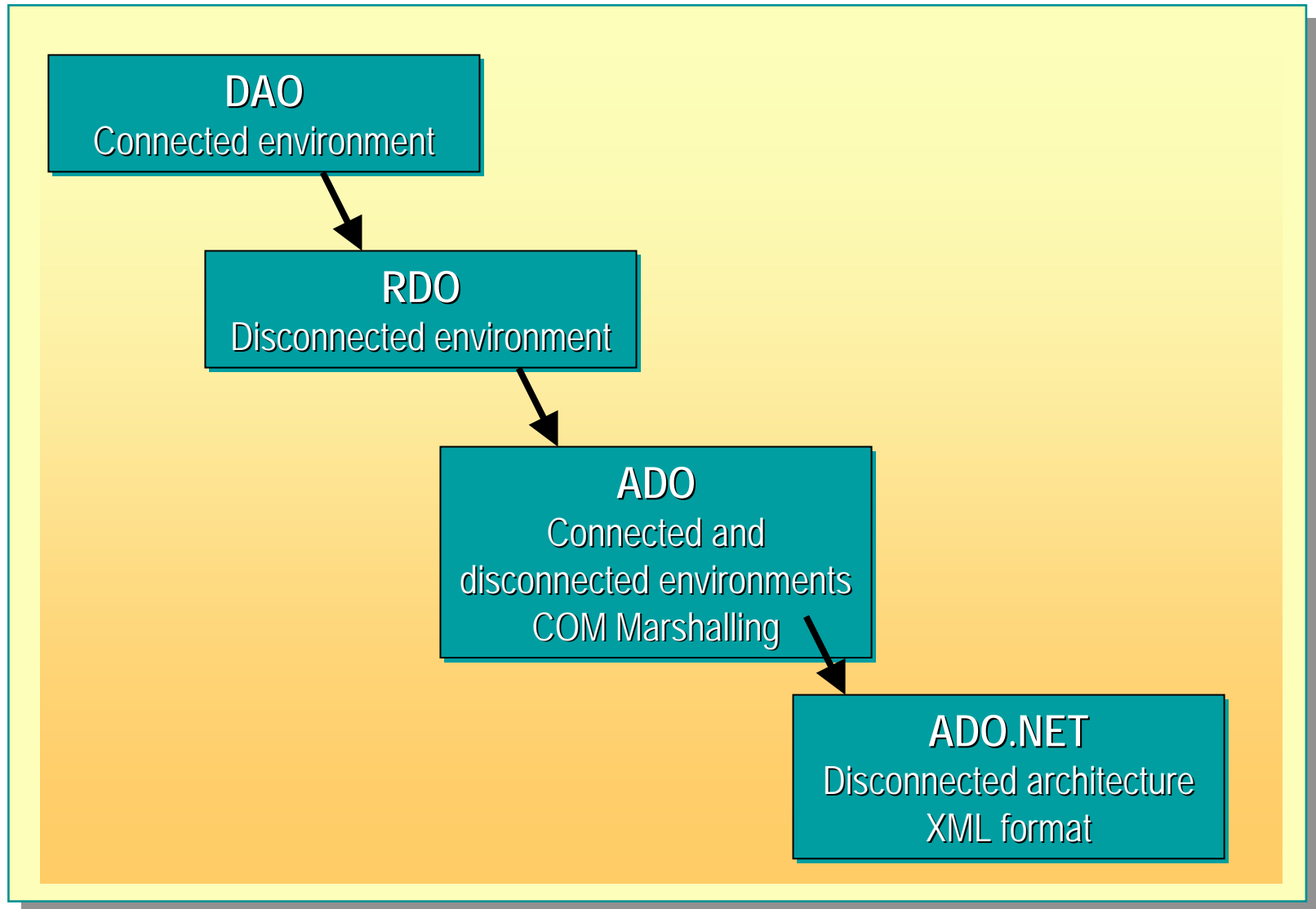
Demonstration: Retrieving Data Using ADO.NET



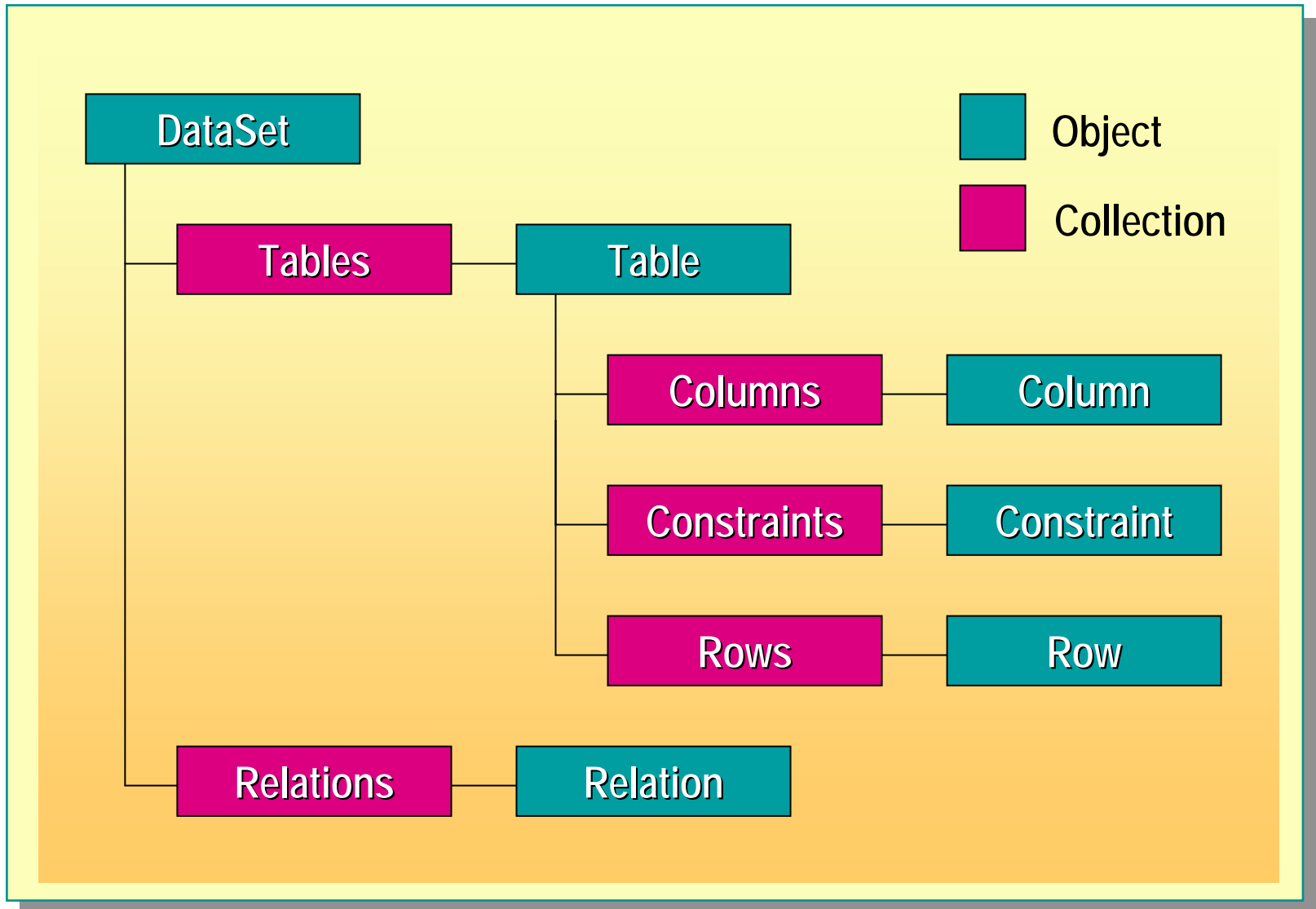
◆ The DataSet Object

- Disconnected Data Review
- The DataSet Object
- Populating DataSets
- Using Relationships in DataSets
- Using Constraints
- Updating Data in the DataSet
- Updating Data at the Source

Disconnected Data Review



The DataSet Object



Populating DataSets

■ Populating DataSets from an RDBMS

```
Dim adaptSQL As New SqlConnection.SqlDataAdapter( _  
    "Select * from authors", conSQL)
```

```
Dim datPubs As DataSet = New DataSet( )  
adaptSQL.Fill(datPubs, "NewTable")
```

■ Programmatically creating DataSets

```
Dim datPubs As New DataSet( )  
Dim tblAuthors As DataTable = New DataTable("authors")  
tblAuthors.Columns.Add("AuthorID", System.Type.GetType( _  
    ("System.Int32")))
```

Using Relationships in DataSets

■ Creating relationships

```
Dim relPubsTitle As New DataRelation("PubsTitles", _  
    datPubs.Tables("Publishers").Columns("pub_id"), _  
    datPubs.Tables("Titles").Columns("pub_id"))  
datPubs.Relations.Add(relPubsTitle)
```

■ Accessing related data

```
Dim PubRow, TitleRow As DataRow, TitleRows( ) As DataRow  
  
PubRow = datPubs.Tables("Publishers").Rows(0)  
TitleRows = PubRow.GetChildRows("PubsTitles")
```

Using Constraints

- Creating new constraints
 - ForeignKeyConstraints
 - UniqueConstraints
- Using existing constraints

```
adaptSQL = New SqlClient.SqlDataAdapter("Select title_id" _  
    & ", title, type, price from titles", conSQL)  
adaptSQL.FillSchema(datPubs, schematype.Source, "Titles")  
adaptSQL.Fill(datPubs, "Titles")  
'Edit some data  
adaptSQL.Fill(datPubs, "Titles")
```

Updating Data in the DataSet

■ Adding rows

```
Dim drNewRow As DataRow = datPubs.Tables("Titles").NewRow  
'Populate columns  
datPubs.Tables("Titles").Rows.Add(drNewRow)
```

■ Editing rows

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

■ Deleting data

```
datPubs.Tables("Titles").Rows.Remove(drDelRow)
```

Updating Data at the Source

■ Explicitly specifying the updates

```
Dim comm As New SqlCommand("Insert titles" & _  
"(title_id, title, type) values(@t_id,@title,@type)")  
comm.Parameters.Add("@t_id", SqlDbType.VarChar, 6, "title_id")  
comm.Parameters.Add("@title", SqlDbType.VarChar, 80, "title")  
comm.Parameters.Add("@type", SqlDbType.Char, 12, "type")  
adaptSQL.InsertCommand = comm  
adaptSQL.Update(datPubs, "titles")
```

■ Automatically generating the updates

```
Dim sqlCommBuild As New SqlCommandBuilder(adaptSQL)  
MsgBox(sqlCommBuild.GetInsertCommand.CommandText)  
adaptSQL.Update(datPubs, "titles")
```

Practice: Using DataSets



◆ Data Designers and Data Binding

- Designing DataSets
- Data Form Wizard
- Data Binding in Windows Forms
- Data Binding in Web Forms

Designing DataSets

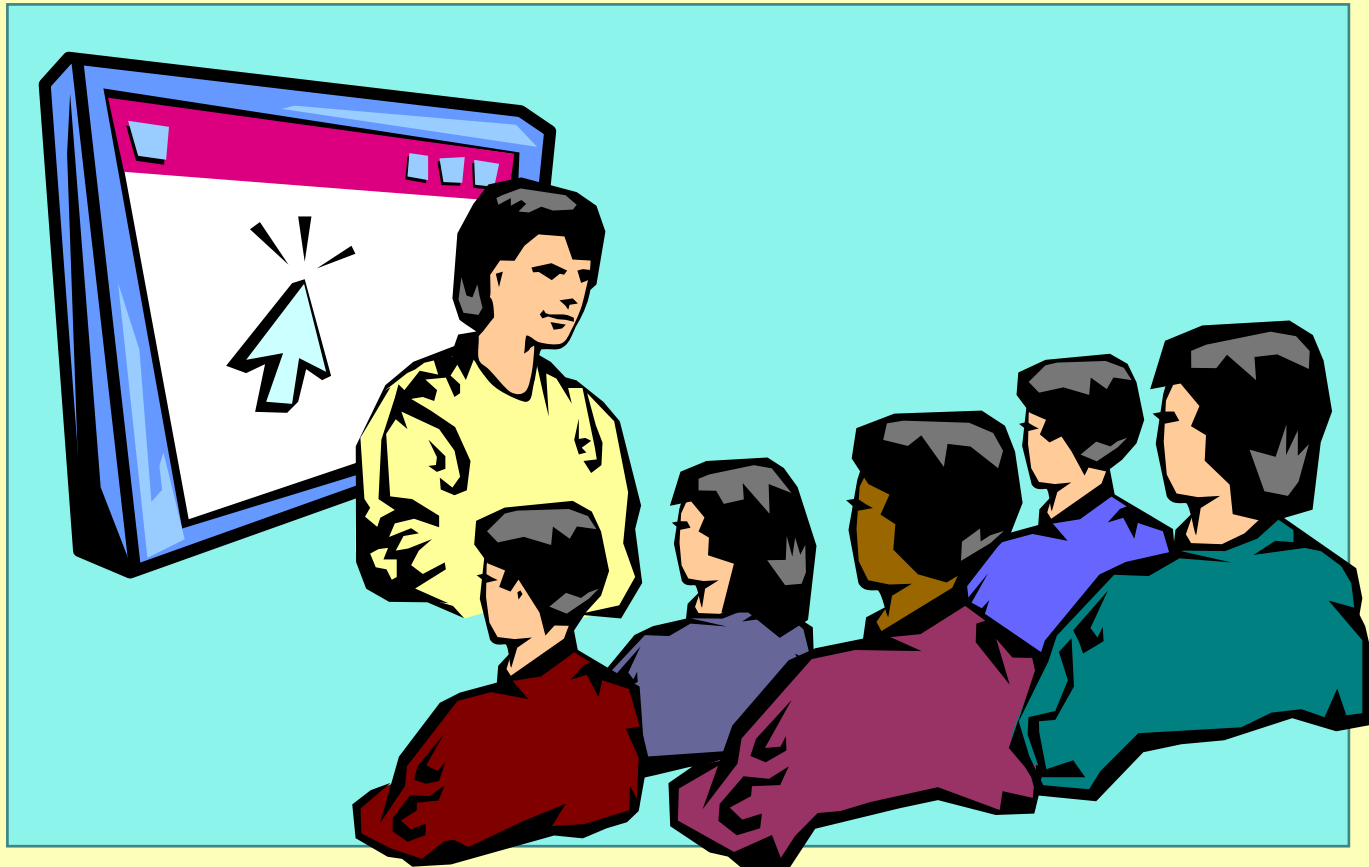
- **DataAdapter Configuration Wizard**
 - Generates a **DataAdapter** object in the **InitializeComponent** procedure for use in your code
- **Generate DataSet Tool**
 - Generates a **DataSet** based on data from an existing **DataAdapter**

Data Form Wizard

- **Information required:**

- Name of **DataSet**
- Connection to be used
- Which tables or views, and which columns within them
- How to display the data
- Which buttons to create

Demonstration: Using the Data Form Wizard



Data Binding in Windows Forms

■ Simple binding

```
da = New SqlConnection.SqlDataAdapter("Select au_lname, " & _  
    "au_fname from authors", sqlconn)  
da.Fill(ds, "authors")  
TextBox1.DataBindings.Add("Text", _  
    ds.Tables("authors"), "au_fname")
```

■ Complex binding

```
da = New SqlConnection.SqlDataAdapter("Select au_lname, " & _  
    "au_fname from authors", sqlconn)  
da.Fill(ds, "authors")  
DataGrid1.DataSource = ds.Tables("authors")
```

Data Binding in Web Forms

- Use impersonation

```
<identity impersonate="true" userName="Karen"  
password="Password"/>
```

- Binding to read-only data

```
Dim sqlComm As New SqlConnection.SqlCommand("Select * from " & _  
    "authors", sqlconn)  
Dim sqlReader As SqlConnection.SqlDataReader  
sqlReader = sqlComm.ExecuteReader  
DataGrid1.DataSource( ) = sqlReader  
DataGrid1.DataBind( )
```

◆ XML Integration

- Why Use Schemas?
- Describing XML Structure
- Creating Schemas
- Using XML Data and Schemas in ADO.NET
- DataSets and XmlDocumentDocuments

Why Use Schemas?

- Define format of data
- Use for validity checking
- Advantages over DTDs
 - XML syntax
 - Reusable types
 - Grouping

Describing XML Structure

- Schemas can describe:
 - Elements in the document
 - Attributes in the document
 - Element and attribute relationships
 - Data types
 - The order of the elements
 - Which elements are optional

Creating Schemas

- Creating schemas from existing XML documents
- Creating schemas from databases
- Working with schemas
- Validating XML documents against schemas

Using XML Data and Schemas in ADO.NET

■ Loading XML data into a DataSet

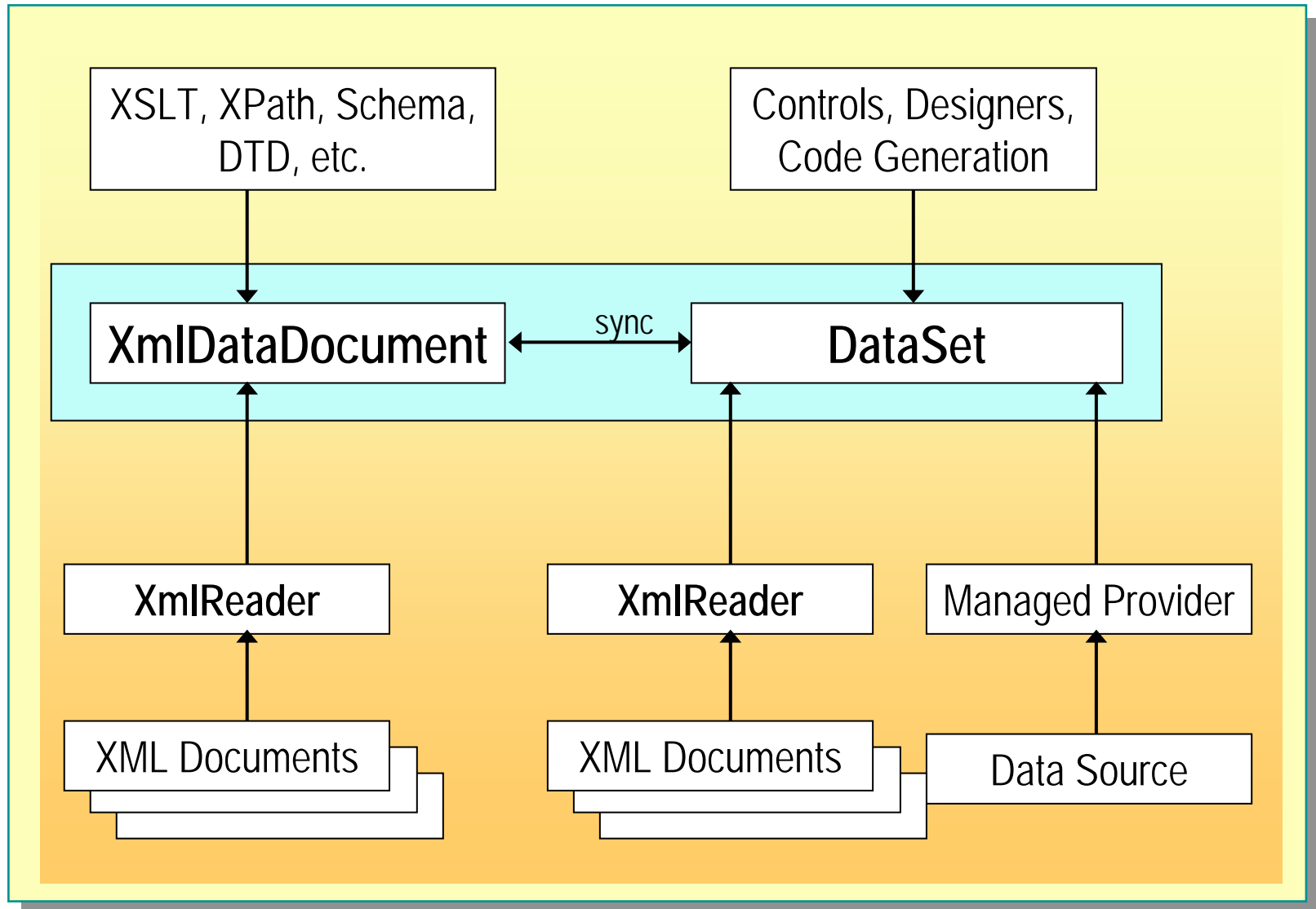
```
Dim datXML As DataSet = New DataSet()  
datXML.ReadXml("c:\publishers.xml")  
MessageBox.Show(datXML.Tables(0).Rows(0)(0).ToString)
```

■ Using a typed DataSet

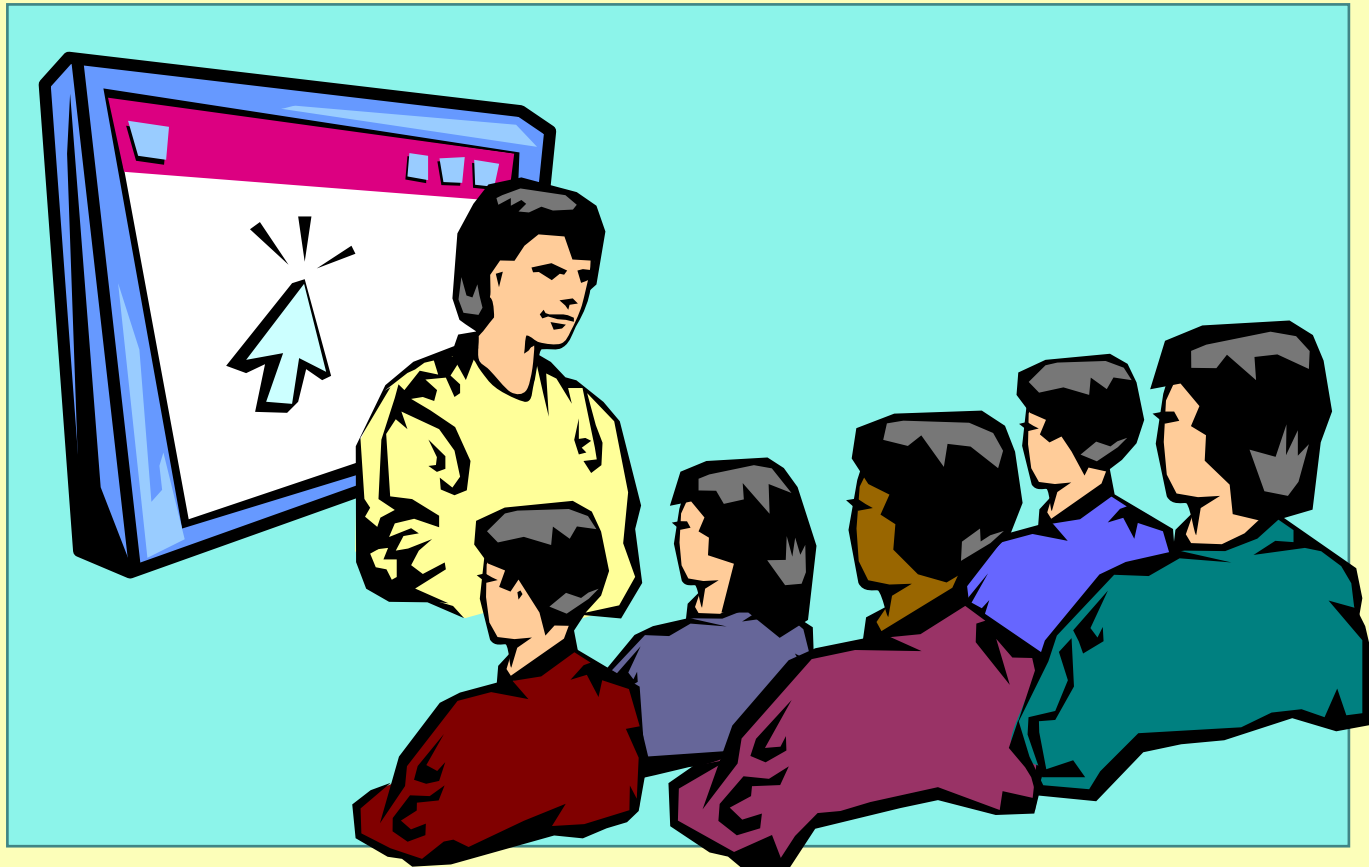
- Increases performance
- Simplifies coding

```
MessageBox.Show(pubs.Publishers(0).pub_id)
```

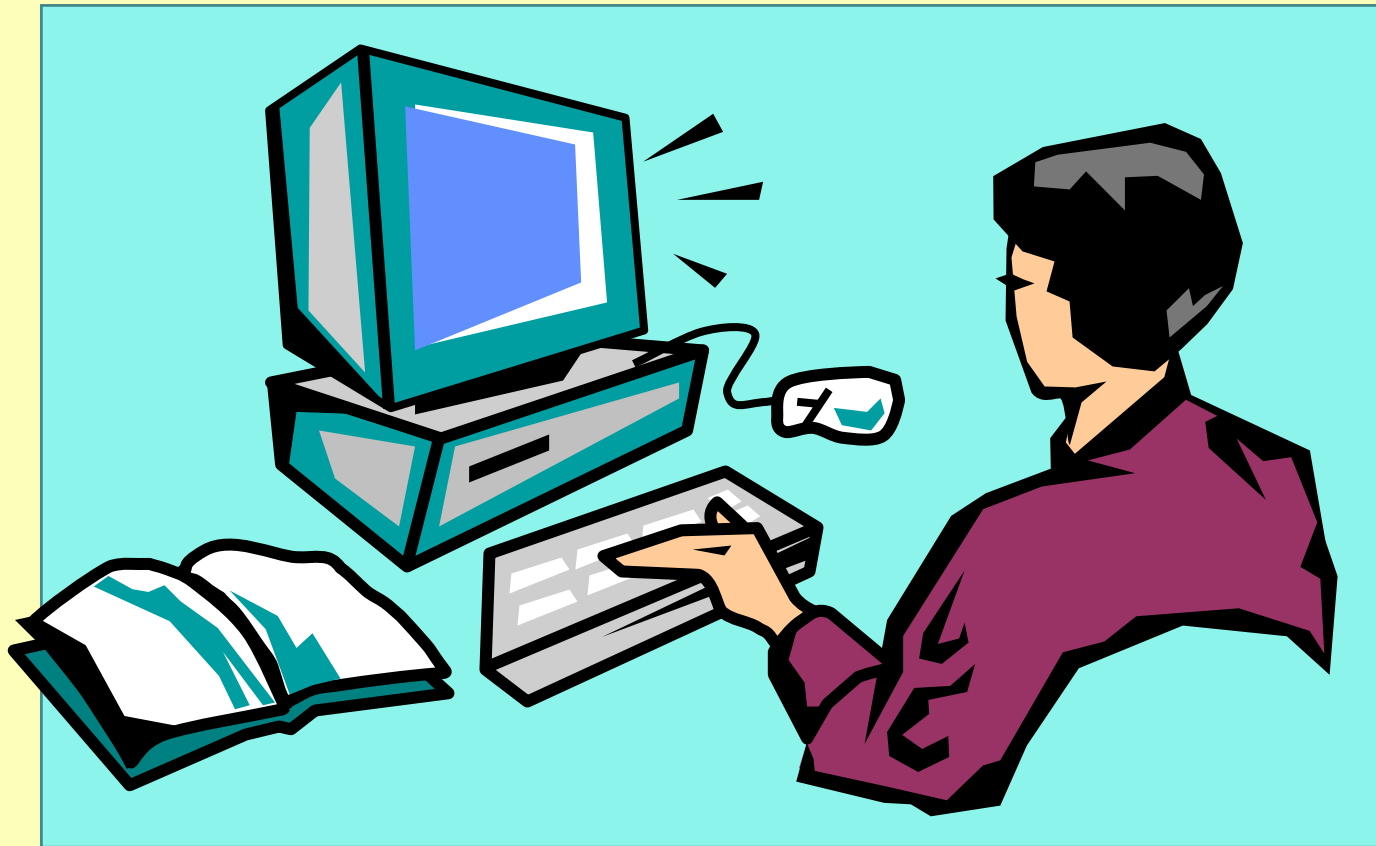
DataSets and XmlDocument Documents



Demonstration: Using XML Schemas



Lab 8.1: Creating Applications That Use ADO.NET



Review

- ADO.NET Overview
- .NET Data Providers
- The DataSet Object
- Data Designers and Data Binding
- XML Integration