

# Mecanismos de acceso a datos usando ADO.NET

# En esta sesión

- Veremos conceptos de ADO.NET para acceso a datos, con ejemplos de uso en aplicaciones ASP.NET

# Agenda

- Modelo de objetos de ADO.NET
- Introducción al uso de ADO.NET
- Conectándose a una base de datos
- Accediendo datos mediante DataSets
- Usando múltiples tablas
- Accediendo datos con DataReaders

# Modelo de objetos de ADO.NET

- Qué es ADO.NET?
- Usando Namespaces
- El modelo de objetos de ADO.NET
- Qué es un Dataset?
- Accediendo datos con ADO.NET
- El modelo de objetos del DataAdapter
- Generando un DataSet
- Controles enlazados

# Qué es ADO.NET?

- **ADO.NET Provee un conjunto de clases para trabajar con datos**
- **ADO.NET es:**
  - Una evolución más flexible de ADO
  - Un sistema diseñado para entornos desconectados
- **ADO.NET provee:**
  - Un modelo de programación con soporte de XML
  - Un conjunto de clases, interfaces, estructuras, y enumeraciones que manejan el acceso a datos dentro del .NET Framework

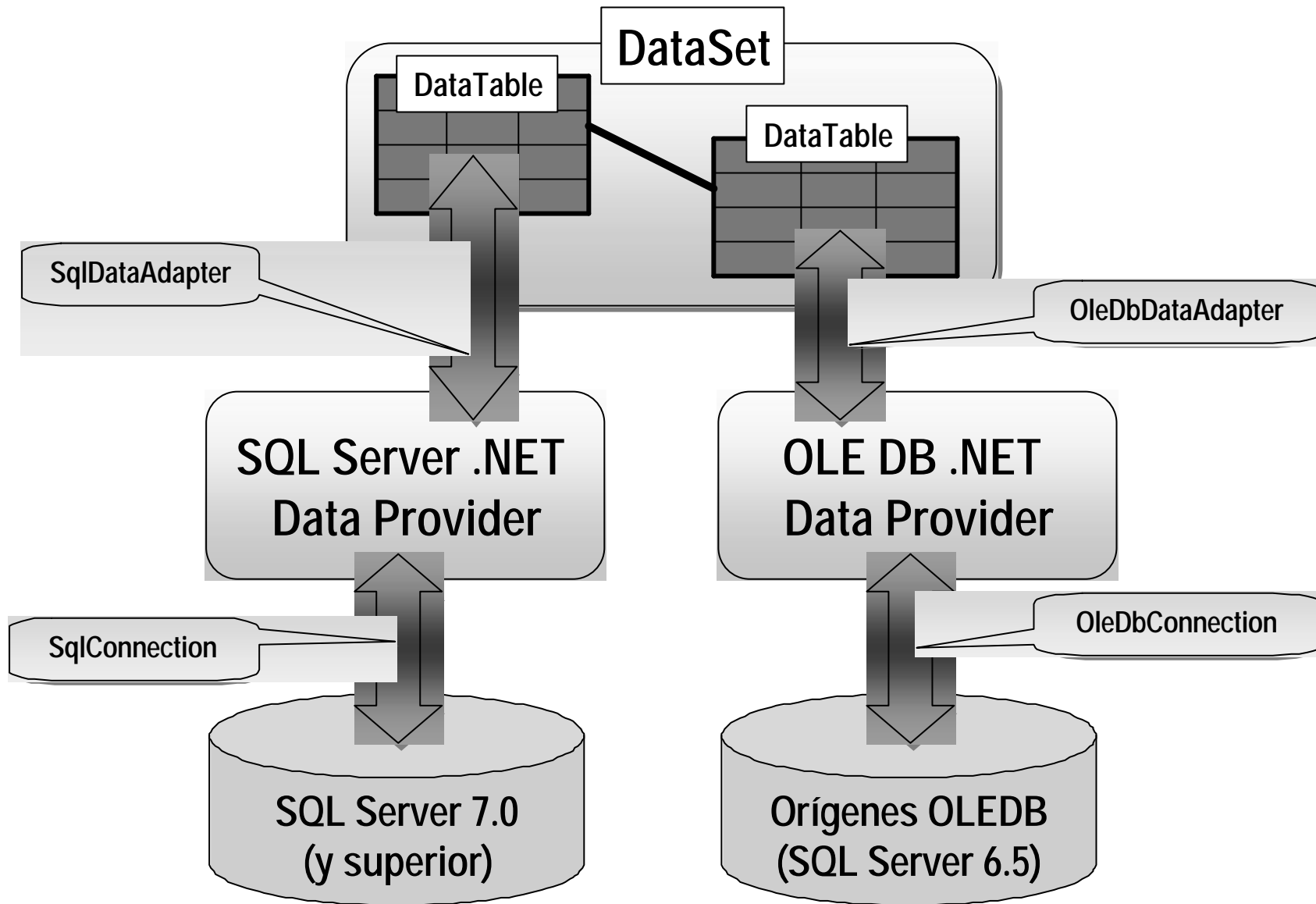
# Usando Namespaces

- Use la instrucción Imports para importar

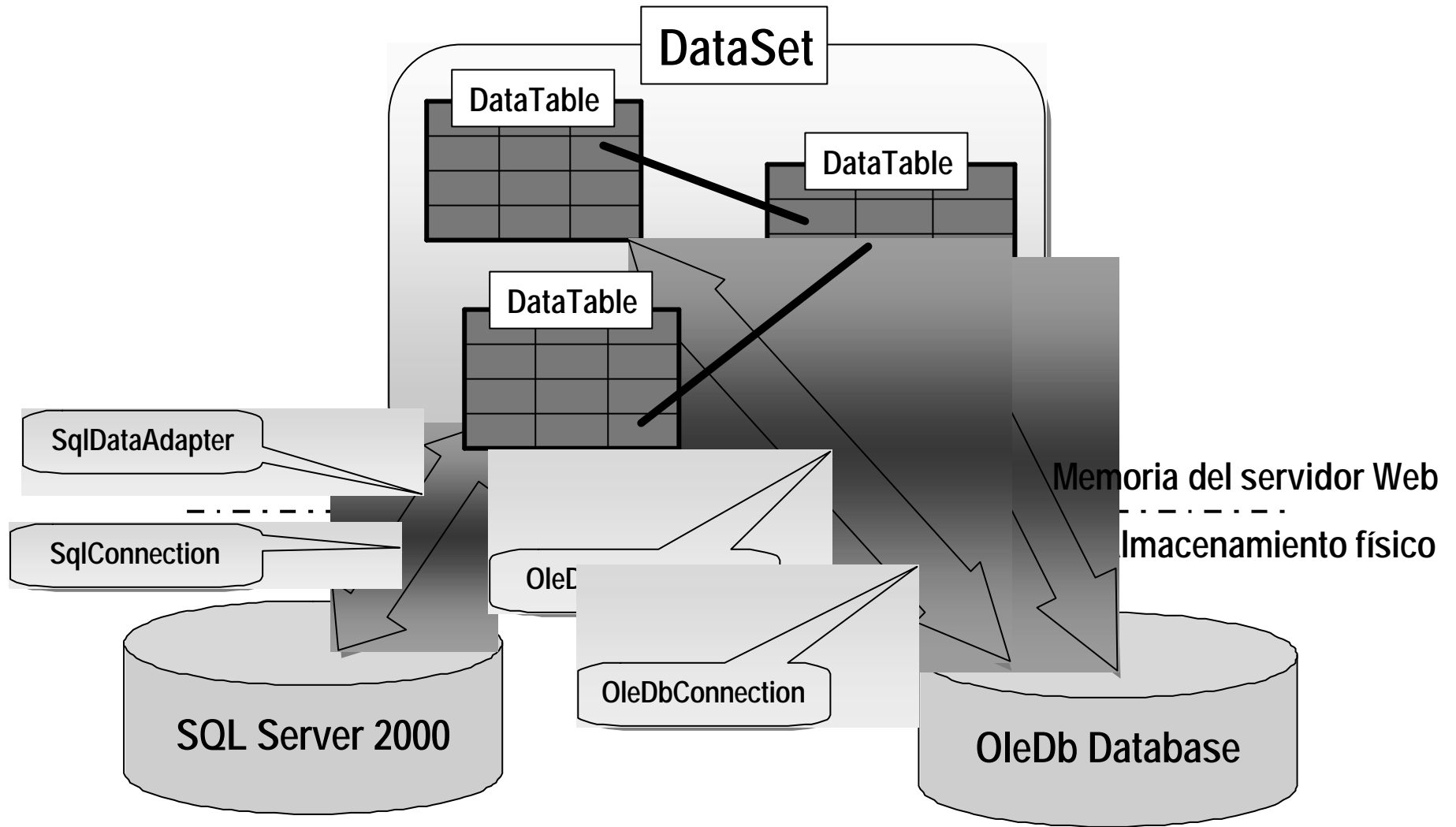
```
Imports System.Data  
Imports System.Data.SqlClient
```

- Los namespaces usados en ADO.NET incluyen:
  - **System.Data**
  - **System.Data.SqlClient**
  - **System.Data.OleDb**

# El modelo de objetos de ADO.NET

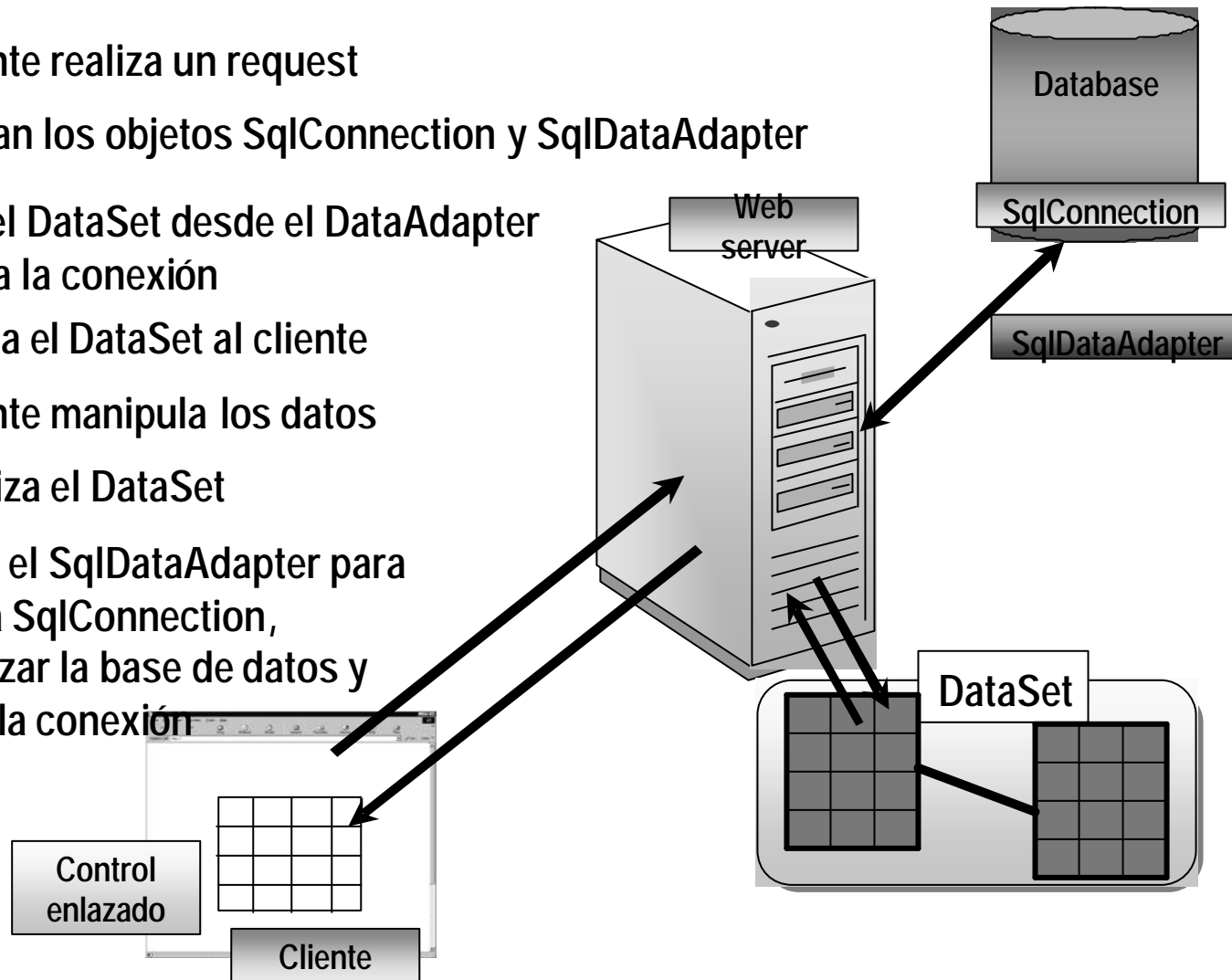


# Qué es un Dataset?

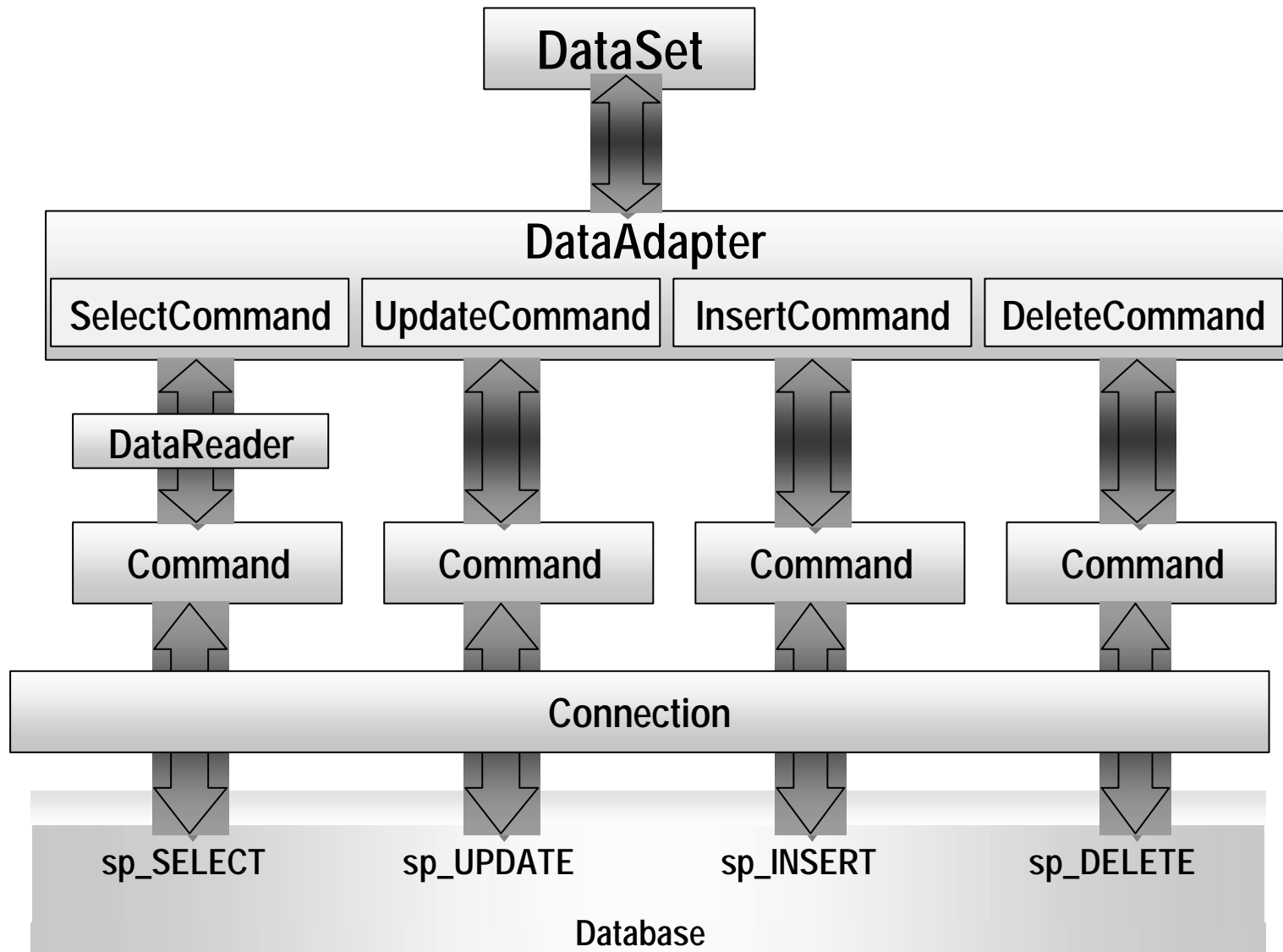


# Accediendo datos con ADO.NET

- 1 El cliente realiza un request
- 2 Se crean los objetos SqlConnection y SqlDataAdapter
- 3 Llena el DataSet desde el DataAdapter y cierra la conexión
- 4 Retorna el DataSet al cliente
- 5 El cliente manipula los datos
- 6 Actualiza el DataSet
- 7 Se usa el SqlDataAdapter para abrir la SqlConnection, actualizar la base de datos y cerrar la conexión



# El modelo de objetos del DataAdapter



# Generando un DataSet

- Se puede generar un dataset...
  - ...a través de la interface de usuario...
    - Crear un dataset que permite acceder datos como un objeto
  - ...a través de código...

- `Dim ds As New DataSet()`

`DataAdapter(s)`

`DataAdapter1.Fill(ds)`

`DataAdapter2.Fill(ds)`

# Controles enlazados

- Son controles que se conectan a un origen de datos y muestran la información
- Entre ellos se encuentran:
  - DropDownList
  - ListBox
  - CheckBoxList
  - RadioButtonList
  - DataGrid
  - DataList
  - Repeater

# Controles enlazados (continuación)

- Indicar las propiedades

Propiedad	Descripción
<b>DataSource</b>	▪ El <b>DataSet</b> que contiene los datos
<b>DataMember</b>	▪ El <b>DataTable</b> dentro del DataSet
<b>DataTextField</b>	▪ El campo del <b>DataTable</b> que se visualizará
<b>DataValueField</b>	▪ El campo del <b>DataTable</b> que contiene el valor del ítem seleccionado

- 

```
DataAdapter1. Fill(ds)  
lstEmployees. DataBind()
```

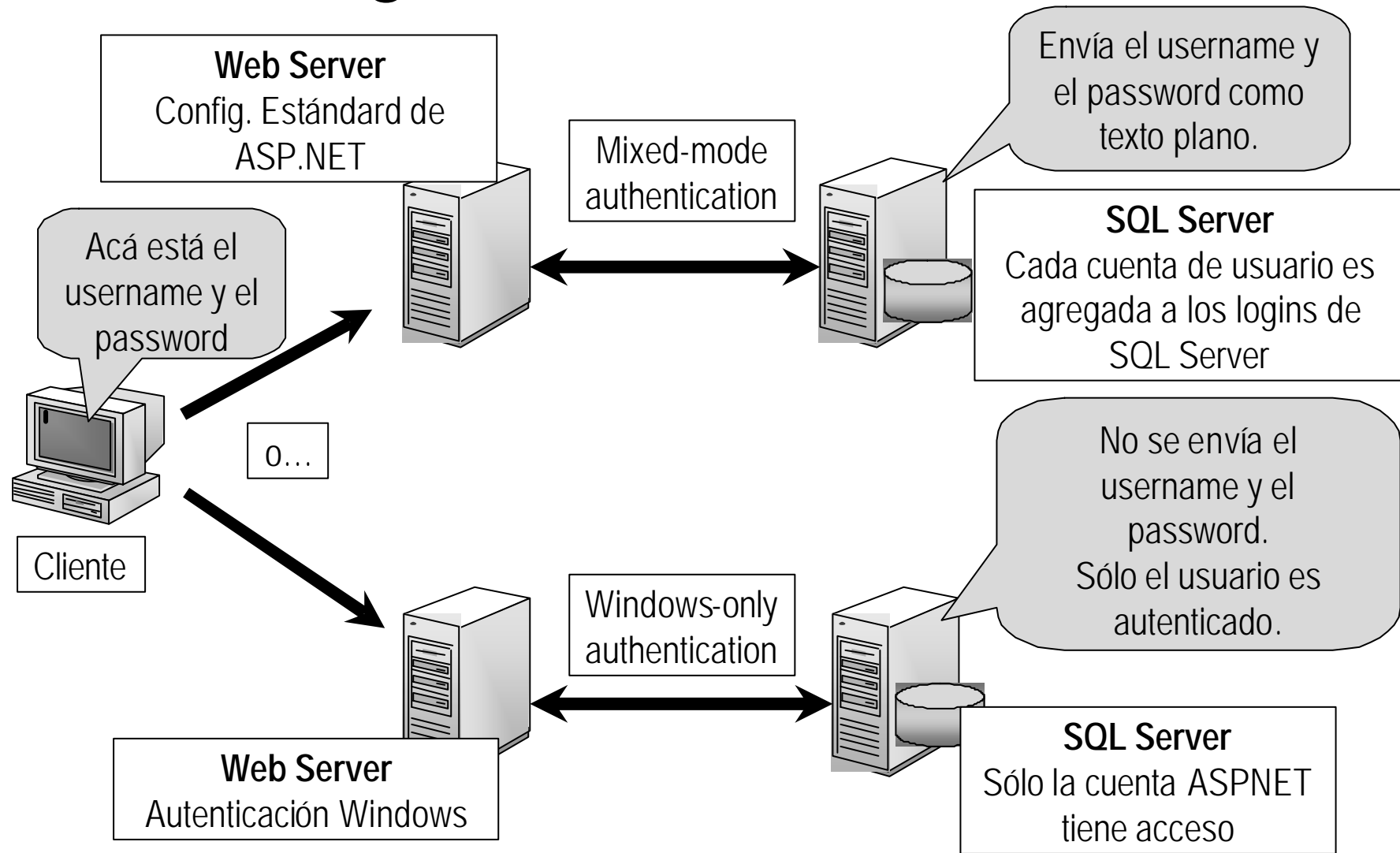
# DataSets vs. DataReaders

<b>DataSet</b>	<b>DataReader</b>
Lectura/Escritura	Read-only
Múltiples tablas de diferentes orígenes	Instrucción SQL sobre una sola base de datos
Desconectado	Conectado
Enlazado a muchos controles	Enlazado a un sólo control
Forward y backward	Forward only
Acceso lento	Acceso rápido
Soportado por las herramientas de Visual Studio .NET	Codificación manual

# Conectándose a una base de datos

- Seguridad en SQL Server
- Creando la conexión

# Seguridad en SQL Server



# Creando la conexión

- Uso de SqlConnection

```
Dim strConn As String = _  
    "data source=localhost; " & _  
    "initial catalog=northwind; " & _  
    "integrated security=true"  
Dim conn As New SqlConnection(strConn)
```

- Indicando parámetros del connection string

- Connection timeout
- Data source
- Initial catalog
- Integrated security
- Password
- Persist security info
- Provider
- User ID

# Accediendo datos mediante DataSets



- Creación de un DataSet
- Uso del DataView
- Enlazando un DataSet a un Control
- Control de errores

# Creación de un DataSet

- Crear y llenar el DataSet con DataTables
  - El método **Fill** ejecuta el **SelectCommand**

```
Dim ds As New DataSet()  
da.Fill(ds, "Authors")
```

- Acceso a un DataTable

```
ds.Tables("Authors").Rows.Count
```

```
Dim r As DataRow  
Dim str As String  
For Each r in ds.Tables("Authors").Rows  
    str &= r(2)  
    str &= r("au_lname")  
Next
```

# Uso del DataView

- El DataView puede ser personalizado para presentar un subconjunto de información
- La propiedad DefaultView retorna la vista por defecto del DataTable

```
Dim dv As DataView  
dv = ds.Tables("Authors").DefaultView
```

- Los DataViews personalizados retornan un subconjunto de un DataTable

```
Dim dv As DataView  
dv = New DataView (ds.Tables("Authors"))  
dv.RowFilter = "state = 'CA' "
```

# Enlazando un DataSet a un Control

- Crear el control

```
<asp:DataGrid id="dg" runat="server" />
```

- Enlazar a un DataSet o DataView

```
dg.DataSource = ds  
dg.DataMember = "Authors"  
dg.DataBind()
```

# Control de errores

- La conexión no puede abrirse
  - El connection string es inválido
  - El servidor o la base de datos no se encuentra
  - Falló el login
- El DataAdapter no puede crear el DataSet
  - Sintáxis SQL inválida
  - Nombre de tabla o campo inválidos

# Usando múltiples tablas

- Almacenamiento de Múltiples Tablas
- Creando relaciones
- Navegar entre tablas relacionadas a través de código

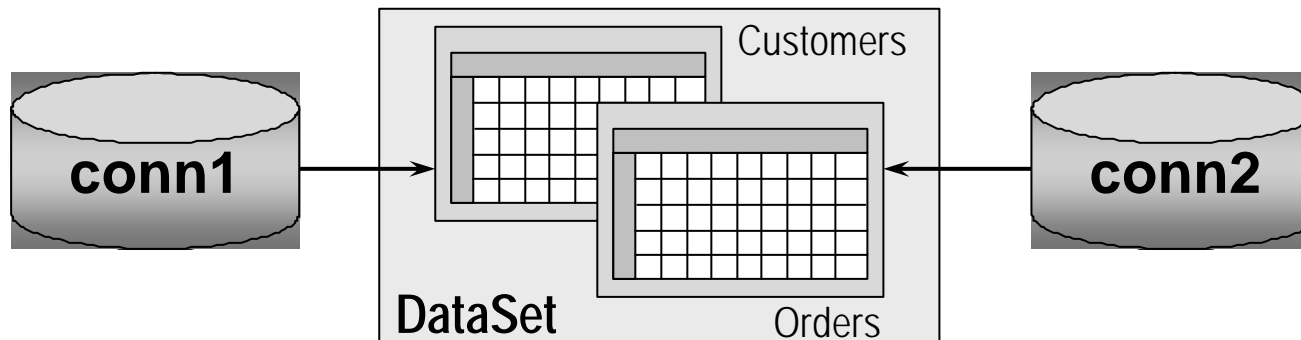
# Almacenamiento de Múltiples Tablas

- Agregar la primer tabla

```
daCustomers = New SqlDataAdapter _  
    ("select * from Customers", conn1)  
daCustomers.Fill(ds, "Customers")
```

- Agregar las siguientes tablas

```
daOrders = New SqlDataAdapter _  
    ("select * from Orders", conn2)  
daOrders.Fill(ds, "Orders")
```



# Creando relaciones

- Identificar la columna padre

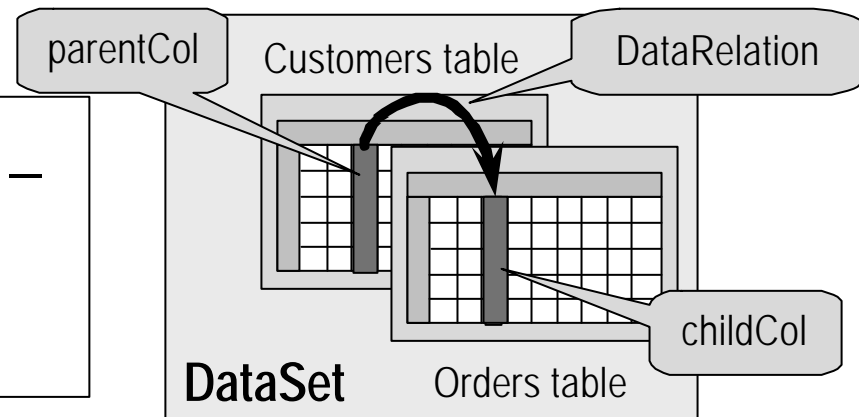
```
Dim parentCol As DataColumn = _  
    ds.Tables("Customers").Columns("CustomerID")
```

- Identificar la columna hijo

```
Dim childCol As DataColumn = _  
    ds.Tables("Orders").Columns("CustomerID")
```

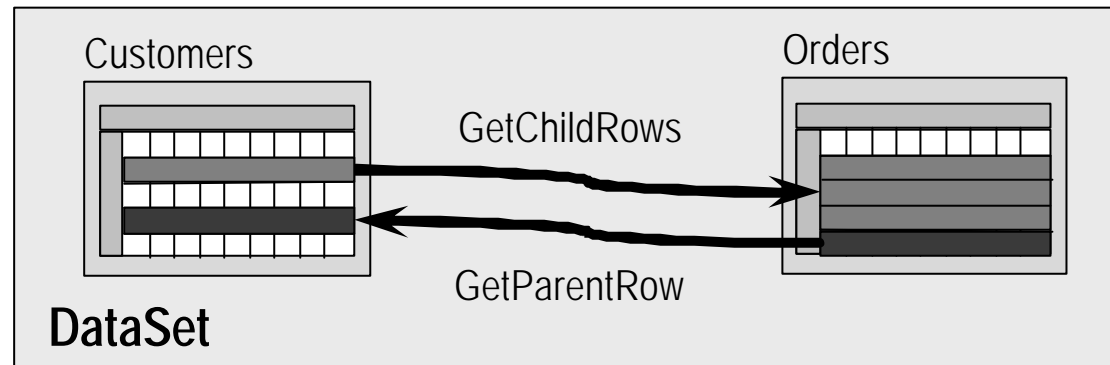
- Crear el DataRelation

```
Dim dr As New DataRelation _  
    ("name", parentCol, _  
    childCol)  
ds.DataRelations.Add(dr)
```



# Navegar entre tablas relacionadas a través de código

```
ds.Tables(index).Rows(index).GetChildRows("relation")  
ds.Tables(index).Rows(index).GetParentRow("relation")
```



# Accediendo datos con DataReaders

- Qué es un DataReader?
- Creando un DataReader
- Leer datos desde un DataReader
- Enlazar un DataReader a un control

# Qué es un DataReader?

- Forward-only, read-only
- Acceso rápido a los datos
- Conectado al origen
- La conexión la maneja usted mismo
- Los datos se manejan por código o a través de controles enlazados
- Usa pocos recursos

# Creando un DataReader

- Para usar un DataReader:
  - 1- Crear y abrir una conexión a una base de datos
  - 2- Crear el objeto **Command**
  - 3- Crear el **DataReader** desde el objeto **Command**
  - 4- Invocar el método **ExecuteReader**
  - 5- Usar el objeto **DataReader**
  - 6- Cerrar el objeto **DataReader**
  - 6- Cerrar el objeto **Connection**
- 7- Usar control de errores  
Try...Catch...Finally

# Leer datos desde un DataReader

- Invocar Read para cada registro
  - Retorna **False** cuando no hay más registros
- Acceso a los campos
  - Por la posición o el nombre del campo
  - Funciones Get, tienen mayor performance

```
Do While myReader.Read()  
    str &= myReader(1)  
    str &= myReader("field")  
    str &= myReader.GetDateTime(2)  
Loop
```

- Cerrar el DataReader
- Cerrar la conexión

# Enlazar un DataReader a un control

- Crear el control

```
<asp:DataGrid id="dgAuthors" runat="server" />
```

- Enlazar al DataReader

```
dgAuthors.DataSource = dr  
dgAuthors.DataBind()
```

# Data Access App Block (DAAP)

- Componente con código optimizado de acceso a datos SQL Server
- Facilita la programación de
  - Llamadas a stored procedures
  - Ejecución de comandos de texto
- Tiene una guía para la implementación del layer de acceso a datos usando ADO.NET
  - Prácticas de buena performance
  - Prácticas de manejo de recursos

# Ejemplo

**[Visual Basic]**

```
Imports Microsoft.ApplicationBlocks.Data
```

```
...
```

```
Dim ds As DataSet = SqlHelper.ExecuteDataset( _  
    connectionString, _  
    CommandType.StoredProcedure, _  
    "getProductsByCategory", _  
    new SqlParameter("@CategoryID", categoryID))
```

**[C#]**

```
using Microsoft.ApplicationBlocks.Data;
```

```
...
```

```
DataSet ds = SqlHelper.ExecuteDataset( connectionString,  
    CommandType.StoredProcedure, "getProductsByCategory",  
    new SqlParameter("@CategoryID", categoryID));
```

# Revisión

- Modelo de objetos de ADO.NET
- Introducción al uso de ADO.NET
- Conectándose a una base de datos
- Accediendo datos mediante DataSets
- Usando múltiples tablas
- Accediendo datos con DataReaders