



# **Module 2: Connecting to Data Sources**

# Overview

- Choosing a .NET Data Provider
- Defining a Connection
- Managing a Connection
- Handling Connection Exceptions
- Connection Pooling

# Lesson: Choosing a .NET Data Provider

- What Are .NET Data Providers?
- The .NET Data Provider Classes
- Which .NET Data Provider Should You Use?

# What Are .NET Data Providers?

## ■ Definition

- A .NET data provider is a set of classes that you use to connect to a data source, and retrieve and update data

## ■ Types of .NET data providers

- SQL Server .NET Data Provider
- OLE DB .NET Data Provider
- ODBC .NET Data Provider
- Others

# The .NET Data Provider Classes

- **XxxConnection** – for example, **SqlConnection**
  - XxxTransaction – for example, **SqlTransaction**
  - XxxException – for example, **SqlException**
  - XxxError – for example, **SqlError**
- **XxxCommand** – for example, **SqlCommand**
  - XxxParameter – for example, **SqlParameter**
- **XxxDataReader** – for example, **SqlDataReader**
- **XxxDataAdapter** – for example, **SqlDataAdapter**
- **XxxPermission** – for example, **SqlClientPermission**

# Which .NET Data Provider Should You Use?

- **SQL Server .NET Data Provider**
  - SQL Server version 7.0 or later
- **OLE DB .NET Data Provider**
  - SQL Server 6.5, Microsoft Access, Oracle, other data sources with OLE DB providers
- **ODBC .NET Data Provider**
  - Legacy data sources that only have ODBC drivers
- **Guidelines for choosing a .NET data provider**

# Lesson: Defining a Connection

- Database Security
- What Is a Connection String?
- How to Set a Connection String

# Database Security

## Using SQL Server security

### ■ Windows Authentication

- Secure validation and encryption
- Auditing
- Password expiration and minimum length
- Account lockout

### ■ Mixed Mode (Windows Authentication and SQL Server authentication)

- Primarily for backward compatibility

# What Is a Connection String?

- A connection string defines the parameters required to make a connection to a data source
- Connection string parameters
  - Provider (OLE DB only)
  - Data Source
  - Initial Catalog
  - Integrated Security
  - User ID/Password
  - Persist Security Info

# How to Set a Connection String

- You can set the `ConnectionString` property only when the connection is closed
- To reset a connection string, you must close and reopen the connection
- Microsoft Access connection

```
Dim cnNorthwind As New _  
    System.Data.OleDb.OleDbConnection()  
cnNorthwind.ConnectionString = _  
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _  
    "Data Source=\Samples\Northwind.mdb;"
```

- Practice

# Lesson: Managing a Connection

- Opening and Closing a Connection
- Handling Connection Events

# Opening and Closing a Connection

- Opening and closing connections explicitly
  - Open and Close methods
- Opening and closing connections implicitly
  - Data adapters can open and close connections automatically when needed
- Using the Dispose method
  - Removes the connection from the connection pool

# Handling Connection Events

- Connection events

- StateChange and InfoMessage

- StateChangeEventArgs class

- Provides data for the state change event of a .NET data provider
- CurrentState and OriginalState properties are read-only

- Visual Basic code sample

```
Private Sub cnNorthwind_StateChange( _  
    ByVal sender As Object, _  
    ByVal e As System.Data.StateChangeEventArgs _  
) Handles cnNorthwind.StateChange
```

- Practice

# Lesson: Handling Connection Exceptions

- What Is Structured Exception Handling?
- How to Handle SqlExceptions
- How to Handle the InfoMessage Event

# Handling Exceptions

- Syntax of the Try... Catch... Finally statement
- Handling multiple exception types
- What are generic exceptions?
- Practice

# How to Handle SqlExceptions

- Write the code to execute inside a Try block
- Write a Catch statement for each specific exception that you want to catch
  - System.Data.SqlClient.SqlException
  - Errors collection
  - SqlError properties (Class, Number)
- Write a generic Catch statement for all other exceptions
- Write a Finally statement to run the code no matter what happens
- End the exception handler with an End Try block
- Practice

[Visual Basic Example](#)

# How to Handle the InfoMessage Event

- The InfoMessage event is triggered when a SQL Server raises an error of severity level 1 - 10
  - These are not fatal errors
  - InfoMessageEventArgs
  - Errors collection
  - SqlError
- Generated by the T-SQL Print statement
  - Useful for debugging stored procedures
- Practice

# Lesson: Connection Pooling

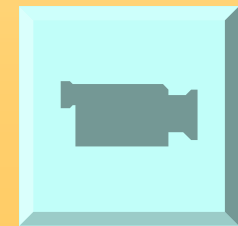
- What Is Connection Pooling?
- Controlling OLE DB Connection Pooling
- Controlling SQL Server Connection Pooling
- Demonstration: Monitoring SQL Server Activity

# What Is Connection Pooling?

- Connection pooling is the process of keeping connections active and pooled so that they can be efficiently reused.
- How connection pooling works
- Example of connection pooling

# Multimedia: How SQL Server Connection Pooling Works

- This animation describes how connection pooling works with Microsoft SQL Server 2000



# Controlling OLE DB Connection Pooling

- Enabled by default
- To disable OLE DB connection pooling:

```
Dim cnNorthwind As New OleDbConnection()  
cnNorthwind.ConnectionString = _  
    "Provider=SQLOLEDB;" & _  
    "Data Source=London;" & _  
    "Integrated Security=SSPI;" & _  
    "OLE DB Services=-4;" & _  
    "Initial Catalog=Northwind;"
```

# Controlling SQL Server Connection Pooling

- Connection string parameters for connection pooling
  - Connection Lifetime
  - Connection Reset
  - Enlist
  - Max Pool Size
  - Min Pool Size
  - Pooling

# Demonstration: Monitoring SQL Server Activity

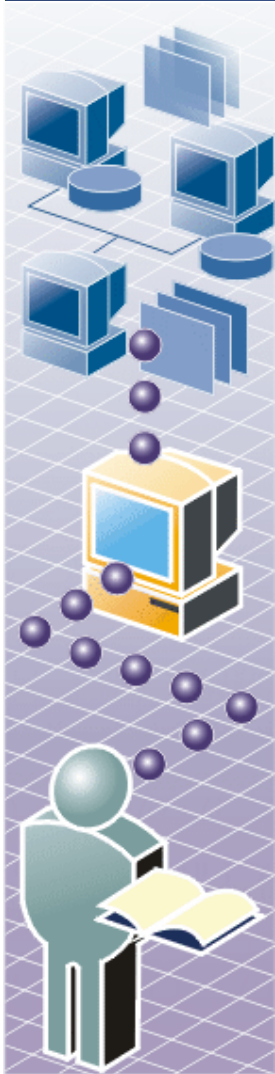


- Creating a SQL Server Profiler trace
- Monitoring SQL Server activity

# Review

- Choosing a .NET Data Provider
- Defining a Connection
- Managing a Connection
- Handling Connection Exceptions
- Connection Pooling

# Lab 2.1: Connecting to Data Sources



- Exercise 1: Creating a Connection to Microsoft SQL Server
- Exercise 2: Handling Common Connection Exceptions
- Exercise 3: Monitoring and Managing Connection Pooling with SQL Server
- Exercise 4 (Optional): Creating a Connection to an OLE DB Data Source