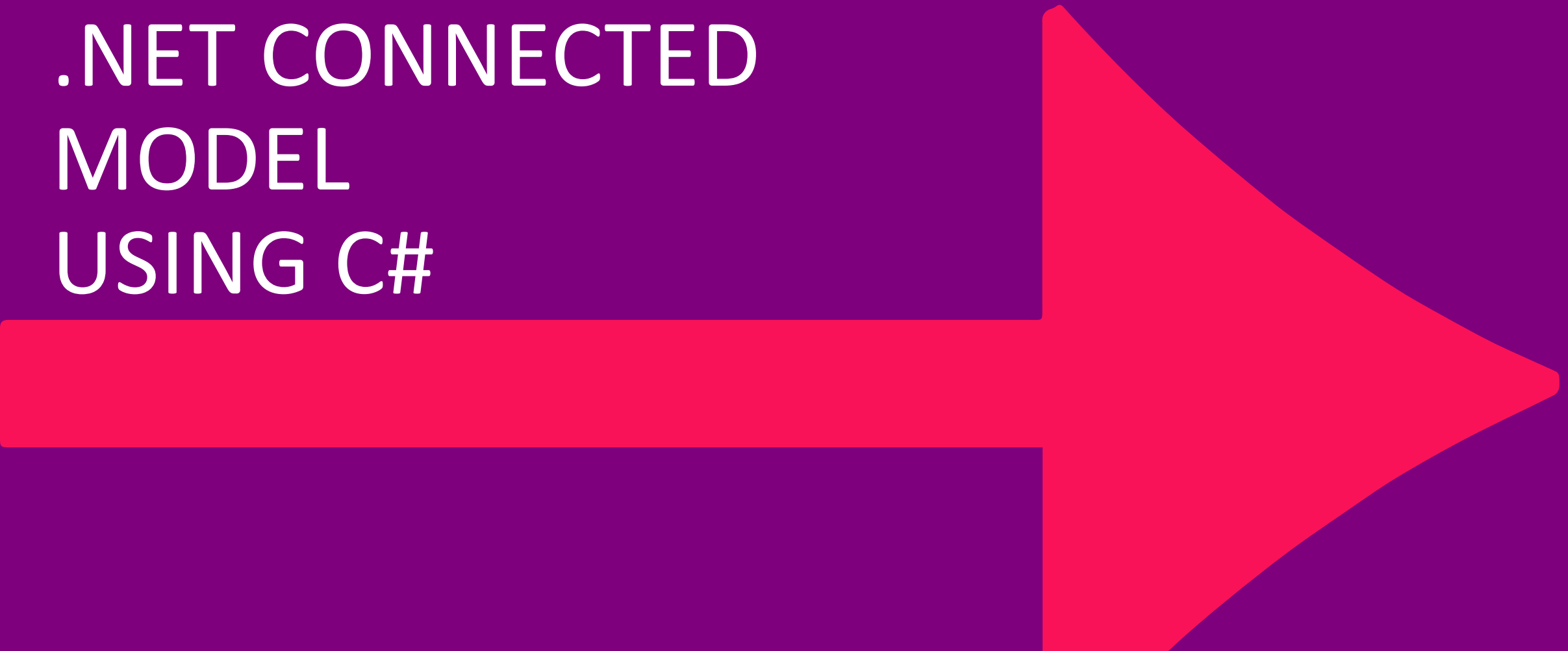
POWERING POTENTIAL

Accessing databases using code

INTRODUCTION TO ACCESSING DATABASES USING .NET

.NET CONNECTED MODEL USING C#



Module objectives

After completing this module you will be able to:

Describe how to use the ADO.NET's Connected model

Demonstrate

- → How to connect to databases
- → How to read data
- → Your ability to update databases by issuing commends

The Connection object

- Used in both the Connected and Disconnected models
- Requires a connection string to define the data source
  - Defines the database's server name, ADO provider, Pooling options, access security credential and more…
- **Open()** method to open the connection

- Always call the **Close()** methods to close an open connection
  - Open as late as possible and close as soon as not needed

- Connections with **the same connection string** are pooled together

An example of creating connections

→ **You can specify connection strings in code**

```
SqlConnection Cn = new SqlConnection(
@"Data Source=.\sqlexpress;Initial Catalog=Northwind;Integrated Security=True" );
```

- **Always use a config file to store connection strings** (see the annex for code example)

  - **Recommended way:**

```
using ( SqlConnection Cn = new SqlConnection(
        @"Data Source=.\sqlexpress;Initial Catalog=Northwind;Integrated Security=True" ))
{
      // Use the connection
      // The system will implicitly Dispose and Close the connection on exit from this block of code.
}
```

Data Source=*ServerName*; **Initial catalog**=*DatabaseName*;**User ID**=*UserName*; **Password**= *Password*

The Command object

- Used to execute any command (Create/Read/Insert/Update/Delete)
  - To retrieve data a DataReader object is needed (seen later)

- ADO.NET distinguishes between different types of execution actions
  - **`ExecuteReader( )`** To read the result of a query
  - **`ExecuteScalar( )`** To read a single scalar value back
  - **`ExecuteNonQuery( )`** To update a database
  - **`ExecuteXmlReader( )`** Read query returns XML

- Let's explore these using code examples...

Reading data using a DataReader

- Use a Command's **ExecuteReader( )** method to read the result
- DataReader gets one row at a time – fast, **read-only**, **forward-only** cursor

```
static void Main(string[] args)
{
    SqlConnection Cn = new SqlConnection(
        @"Data Source=.\sqlexpress;Initial Catalog=Northwind;Integrated Security=True");

    SqlCommand Com = new SqlCommand("SELECT * FROM customers", Cn);
    Cn.Open();

    SqlDataReader Dr = Com.ExecuteReader();
    while (Dr.Read())
    {
        Console.WriteLine(Dr[0] + " - " + Dr["contactName"]);
    }
    Cn.Close();
}
```

Access by index (column no)

Or use the column name

How to update databases?

→ Use a Command object to update a database

```
SqlConnection Cn;     // then instantiate the connection object

SqlCommand Com = new SqlCommand (
        @"UPDATE customers SET Region='DC' WHERE CustomerID='ALFKI'", Cn);

Cn.Open();

int Res = Com.ExecuteNonQuery();

Cn.Close();

Console.WriteLine(Res + " rows updated");
```

Insert a new row

```
string query = @"INSERT INTO Customers (ID, company, name, city)
                 VALUES ('ID888', 'QA', 'Mike B', 'London')";

SqlConnection Cn; // then instantiate the connection object

SqlCommand Com = new SqlCommand(query, Cn);

Cn.Open();

int Res = Com.ExecuteNonQuery();

Cn.Close();

Console.WriteLine(Res + " rows inserted");
```

How to delete rows?

```
SqlConnection Cn;       // then instantiate the connection object

SqlCommand Com =
        new SqlCommand("DELETE customers WHERE CustomerID='ID888'", Cn);

Cn.Open();

int Res = Com.ExecuteNonQuery();

Cn.Close();

Console.WriteLine(Res + " rows deleted");
```

Scalar queries

```
SqlConnection Cn;       // then instantiate the connection object

SqlCommand Com =
        new SqlCommand("SELECT Count(*) FROM customers", Cn);

Cn.Open();

int total = (int)Com.ExecuteScalar();

Cn.Close();

Console.WriteLine(total + " rows detected");
```

Note: **ExecuteScalar()** Returns and Object. Casting is required.

Module objectives

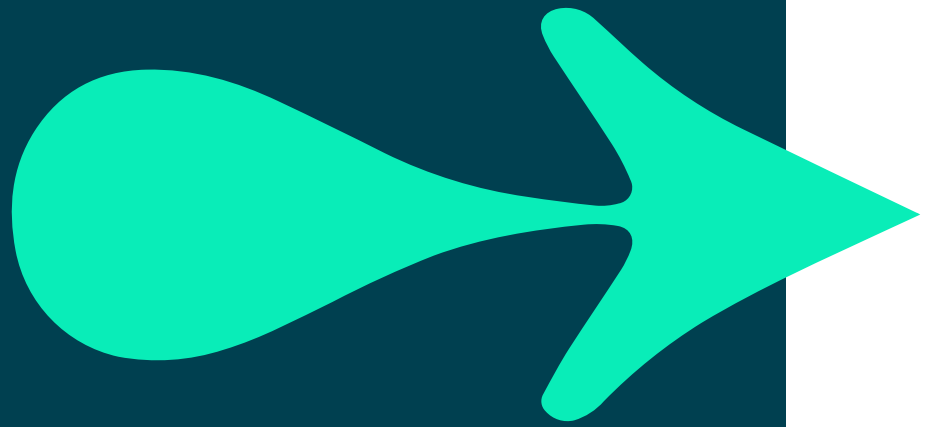
After completing this module you will be able to:

Describe how to use the ADO.NET's Disconnected model

Demonstrate how to:

- → Connect and read data into a client side cursor in memory
- → How to manipulate the disconnected data
- → Read and navigate through rows
- → Execute additional queries on the client side
- → Update databases with changes

EXERCISE



Lab

Accessing Databases using Java

Duration 30 minutes
(Java code is provided)