C#

UNIT 3 : Advanced Features of C#, Window Based Programming and ADO.NET

SOFTWARE DEVELOPMENT TOOLS(C# and ASP.NET)

II MSc CS

V.B.Buvaneswari

Multithreading

- C# supports the concept of multithreading which enables us to execute two or more "parts" of a program concurrently.
- Each part is known as a thread. A thread is basically a separate sequence of instructions designed for performing a "specific task" in the program.
- A task may represent an operation such as reading data, sending a file over the Internet, printing some results or performing certain calculations.

Multithreading

- Multithreading, therefore, means performing multiple tasks at the same time during the execution of a program.
- The execution of a C# program starts with a single thread called the main thread that is automatically run by the Common Language Runtime (CLR) and the operating system.
- From the main thread, other threads can be created for performing desired tasks in the program.

Multithreading

- The process of developing a program for execution with multiple threads is called multithreaded programming and the process of execution is called multithreading.
- The main advantage of multithreading is that it enables us to develop efficient programs that could optimize the use of computer resources such as memory, 1/0 devices and time.

Multithreading

- C# defines a namespace known as System.
- Threading containing classes and interfaces that are required for developing and running multithreaded programs.
- The statement using System.Threading should be included at the start of any multithreaded program.

Creating and Starting Threads

```
using System;
using System.Threading;
public class AOne
{
    public void First()
    {
        Console.WriteLine("First method of AOne class is running on T1 thread.");
        Thread.Sleep(1000);
        Console.WriteLine("The First method called by T1 thread has ended.");
    }
    public static void Second()
    {
        Console.WriteLine("Second method of AOne class is running on T2 thread.");
        Thread.Sleep(2000);
        Console.WriteLine("The Second method called by T2 thread has ended.");
    }
}
```

Creating and Starting Threads (cont..)

```
public class ThreadExp
{
    public static int Main(String[] args)
    {
        Console.WriteLine("Example of Threading");
        AOne a = new AOne();
        Thread T1 = new Thread(new ThreadStart(a.First)); // Creating thread T1
        T1.Start(); // Starting thread T1
        Console.WriteLine("T1 thread started.");
        Thread T2 = new Thread(new ThreadStart(AOne.Second)); Creating T2
        T2.Start(); Starting thread T2
        Console.WriteLine("T2 thread started.");
        return 0;
    }
}
```

Output

```
Example of Threading
First method of AOne class is running on T1 thread.
T1 thread started.
Second method of AOne class is running on T2 thread.
T2 thread started.
The First method called by T1 thread has ended.
The Second method called by T2 thread has ended.
Press any key to continue . . .
```

Reflection

- *Reflection* is the process by which a program can read its own metadata. A program is said to reflect on itself, extracting metadata from its assembly and using that metadata either to inform the user or to modify its own behavior.
- *Attributes* are a mechanism for adding metadata, such as compiler instructions and other data about data, methods, and classes, to the program itself.

WIN FORMS

- Form is the very first entity typically included in a Windows-based application. lt hosts a number of other controls for performing desired functions.
- At runtime, a form continuously waits for an event to occur, such as the clicking of the mouse or pressing of a key.
- As soon as an event occurs, it triggers the corresponding event-handling code.

WIN FORMS

- In C#,a form can be created by inheriting the Form class contained in the System.windows.Forms namespace.
- The Form class already supports a number of properties and methods, which make the job of the programmer a lot easier.

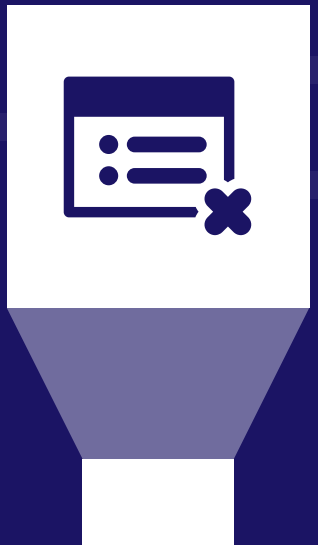
CONTROLS

Textbox

To accept text from the user of an application.

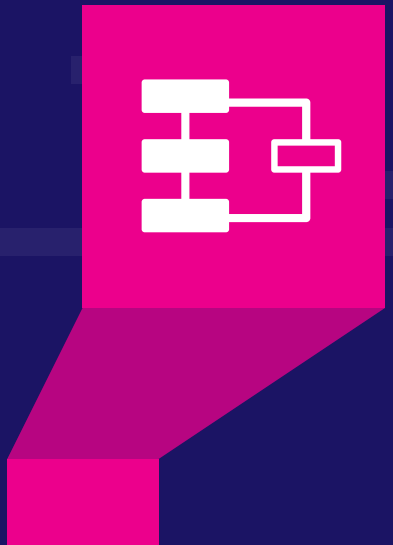
List box

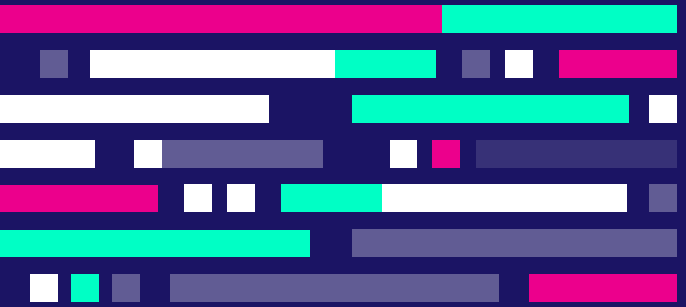
To display a list from which a user can select a specific item.



Buttons

To perform a specific task by causing an event to occur on clicking.



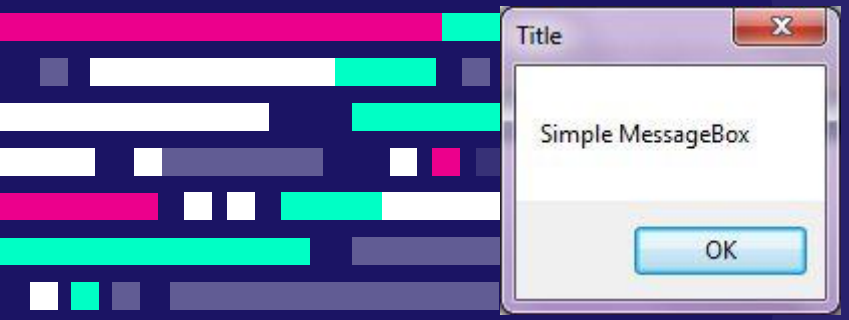


MESSAGE BOX

MessageBox class has an overloaded static Show method that displays a message box with a message and action buttons. The action buttons can be OK and Cancel, Yes and No etc. Here are some of the options that can be used in C# message box.

The simplest form of a MessageBox is a dialog with a text and OK button. When you click OK button, the box disappears.

MESSAGE BOX



The following code snippet creates a simple Message Box.

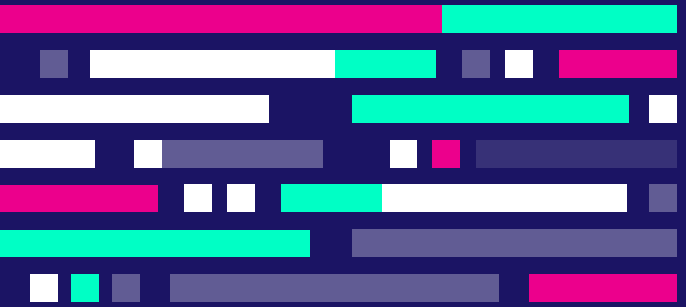
- string message = "Simple MessageBox";
- MessageBox.Show(message);

MessageBox with Title

The following code snippet creates a simple MessageBox with a title.

- string message = "Simple MessageBox";
- string title = "Title";
- MessageBox.Show(message, title);

EVENT HANDLING



- Event handling is performed differently in .NET than it is in Visual C++.
- The Control class, or any derived class, has virtual functions, which can be overridden to raise an event.
- Therefore, a Form class can use any event handler in its hierarchy.
- For example, the Control class has many event methods including GotFocus, ControlRemoved, LostFocus etc.

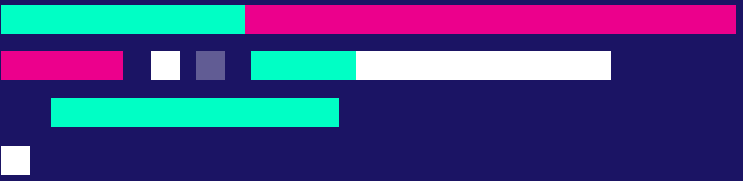
EVENT HANDLING

Event Handler	Description
OnClick, OnDoubleClick	Raise the Click and DoubleClick event for button presses on the mouse.
OnDeactivate	Raises the Deactivate event when a form is deactivated.
OnKeyDown, OnKeyPress, OnKeyUp	Raise Key events from the keyboard.
OnMouseDown, OnMouseEnter, OnMouseHover, OnMouseLeave, OnMouseMove, OnMouseUp	Raise Mouse events.
OnMove	Raises the Move event when a control is moved.
OnPaint	Inheriting classes should override this method to handle this event. Call base class's OnPaint to send this event to any registered event listeners.
OnResize	Raises the Resize event when a control is resized.

ADO.NET OBJECTS

OBJECTS

Used to perform database operations.



- Connection
- Command
- DataREADER
- DataAdapter

SQLConnection

The ADO.NET SqlConnection class belongs to System.Data.SqlClient namespace and is used to establish an open connection to the SQL Server database. The most important point that you need to remember is the connection does not close implicitly even it goes out of scope. Therefore, it is always recommended and always a good programming practice to close the connection object explicitly by calling the Close() method of the connection object

SQLConnection

The ADO.NET SqlConnection class has three constructors which are shown in the below image.

```
public SqlConnection();
public SqlConnection(string connectionString);
public SqlConnection(string connectionString, SqlCredential credential);
```

SQLCommand

The ADO.NET SqlCommand class in C# is used to store and execute the SQL statement against the SQL Server database.

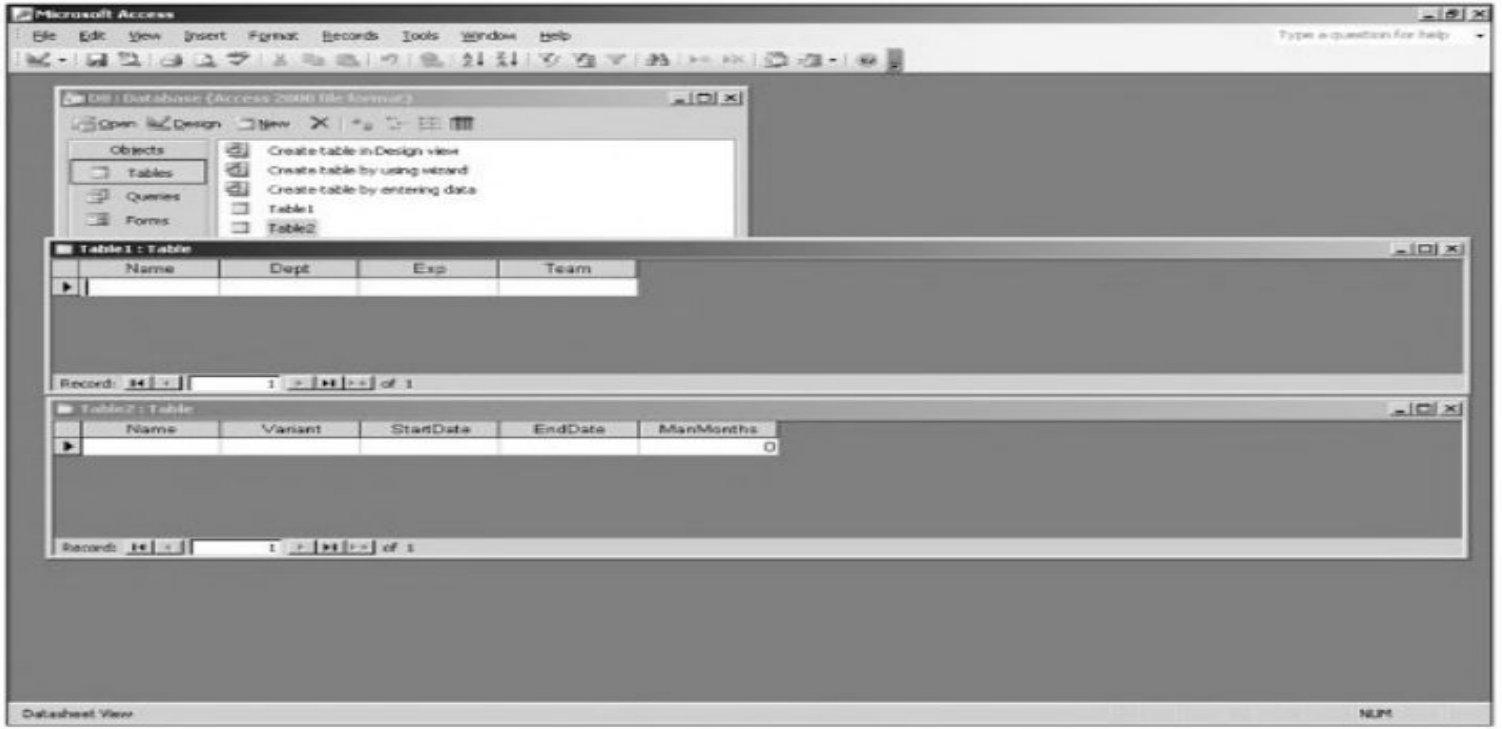
```
public SqlCommand();
public SqlCommand(string cmdText);
public SqlCommand(string cmdText, SqlConnection connection);
public SqlCommand(string cmdText, SqlConnection connection, SqlTransaction transaction);
public SqlCommand(string cmdText, SqlConnection connection, SqlTransaction transaction,
SqlCommandColumnEncryptionSetting columnEncryptionSetting);
```

How to Connect DataGridView

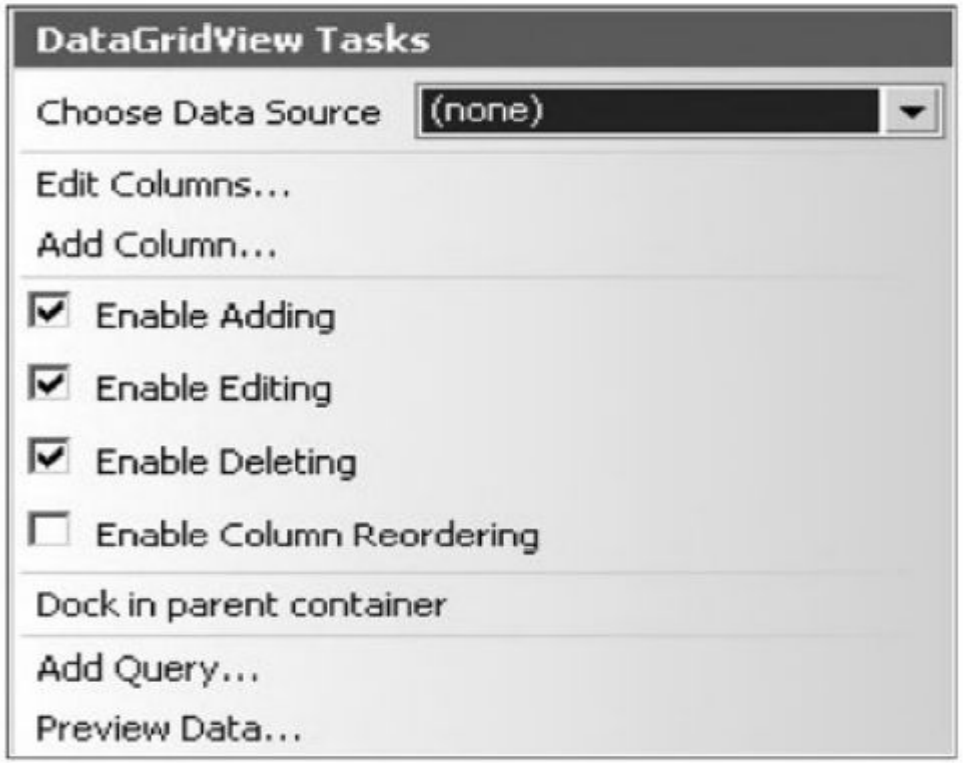
Linking Data Sources

As we can see in the **Design** view, both **Employee** and **Project** tabs use the **DataGridView** control for displaying the records submitted by the user. To populate the **DataGridView** from a backend data source, it must be linked with the corresponding data source. The following are the steps for linking **DataGridView** control with a data source:

1. Select the **DataGridView** control and open the **DataGridView** Tasks pane, as shown below:
2. Select the **Add Project Data Source** option from **Choose Data Source** list to initiate the **Data Source Configuration Wizard**, as shown below:
3. Select the **Database** option and click Next to display the **Choose Your Data Connection** screen, as shown below:
4. Click the **New Connection** button to display the **Add Connection** dialog box, as shown below:



MS access tables



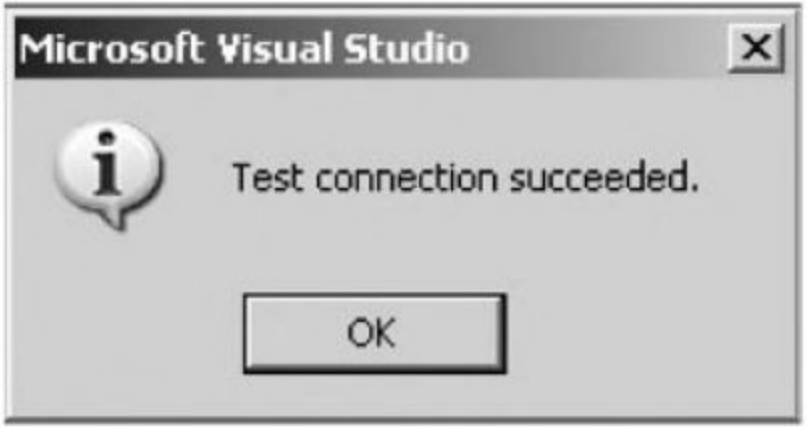
DataGridView tasks pane

The Data source configuration wizard

The choose Your Data connection screen

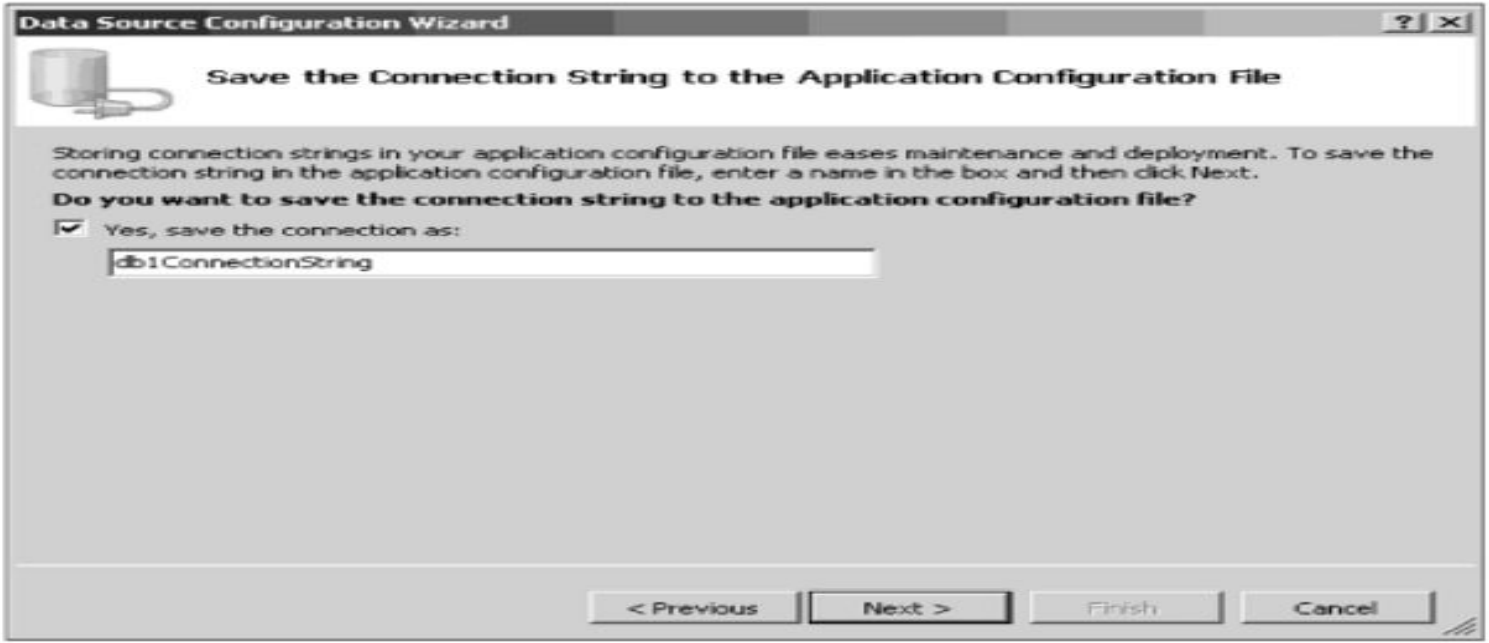
The add Connection Dialog Box

5. Select the **Microsoft Access Database File (OLE DB)** option in the **Data Source** field, enter the location of the backend MS Access file in the **Database** file name text box and click OK. A message box appears confirming the successful data source connection, as shown below:



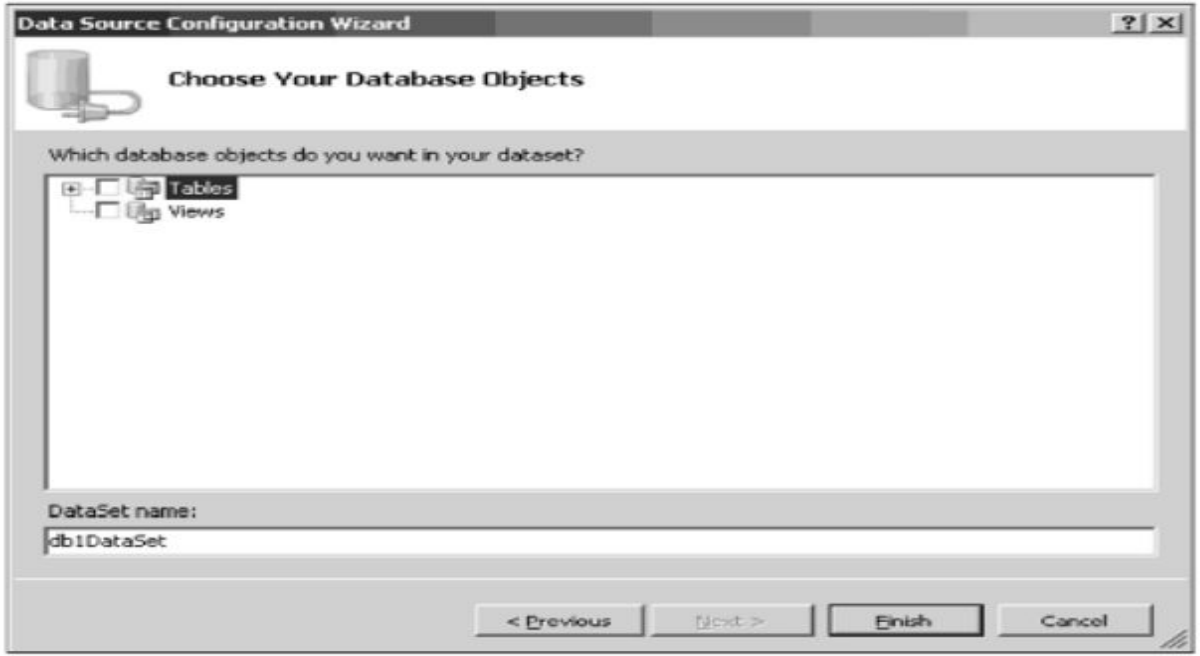
Microsoft Visual Studio Message Box

6. Click **Next** to display the **Save the Connection String to the Application Configuration** file screen, as shown below:



The Save the Connection String to the Application Configuration file screen

7. Click *Next* to display the **Choose Your Database Objects** screen, as shown below:



8. Select the table and the corresponding columns that you want to display in the **DataGridView** and click *Finish* to link the selected data source with the **DataGridView** control.

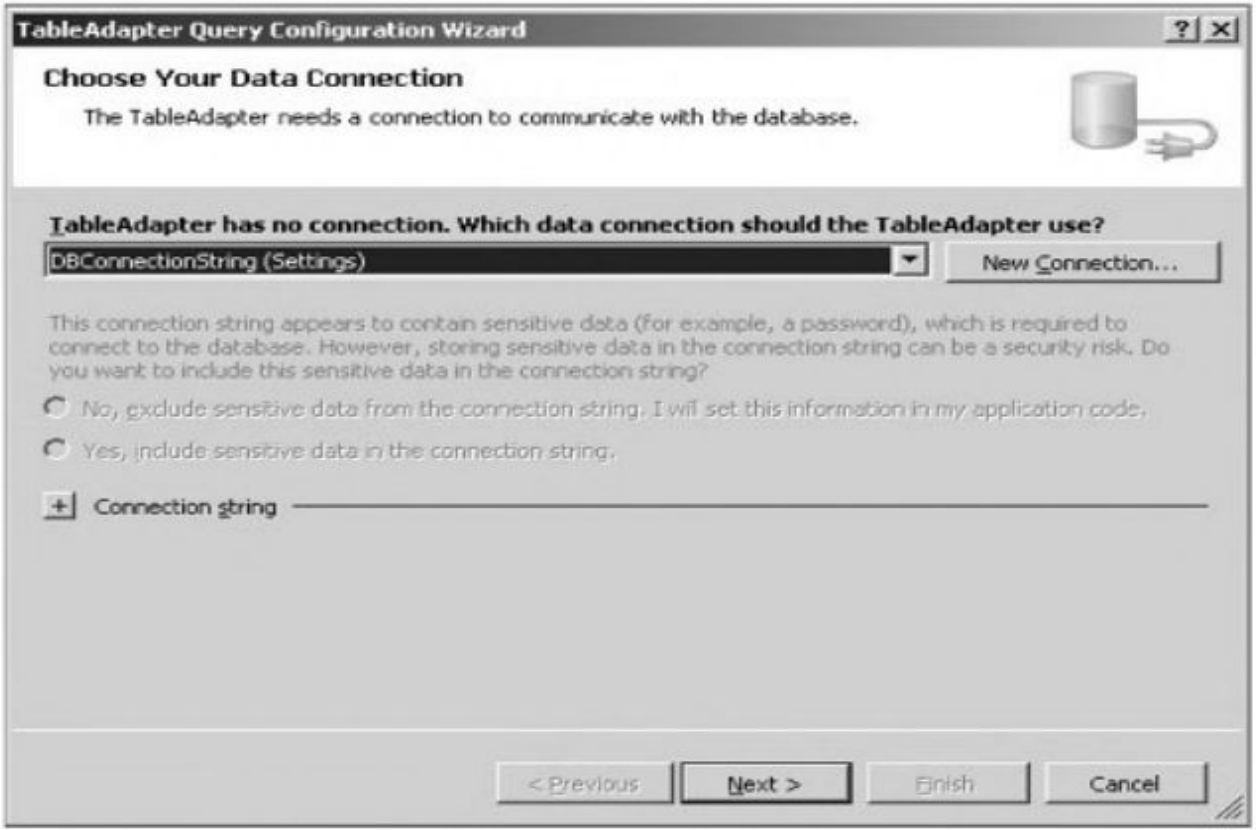
.NET Data Providers

The Database can not directly execute our C# code, it only understands SQL. So, if a .NET application needs to retrieve data or to do some insert, update, and delete operations from or to a database, then the .NET application needs to

1. Connect to the Database
2. Prepare an SQL Command
3. Execute the Command
4. Retrieve the results and display them in the application

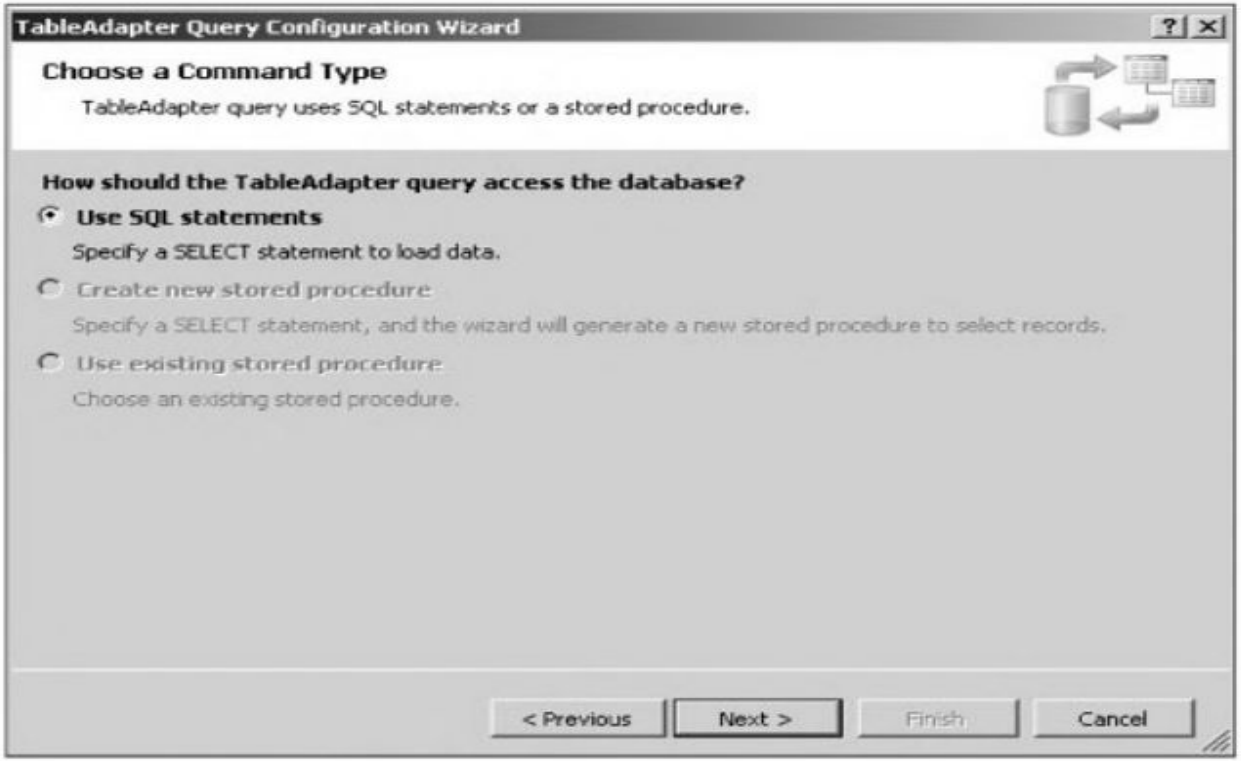
And this is possible with the help of .NET Data Providers.

How to Connect to SQL Data Source



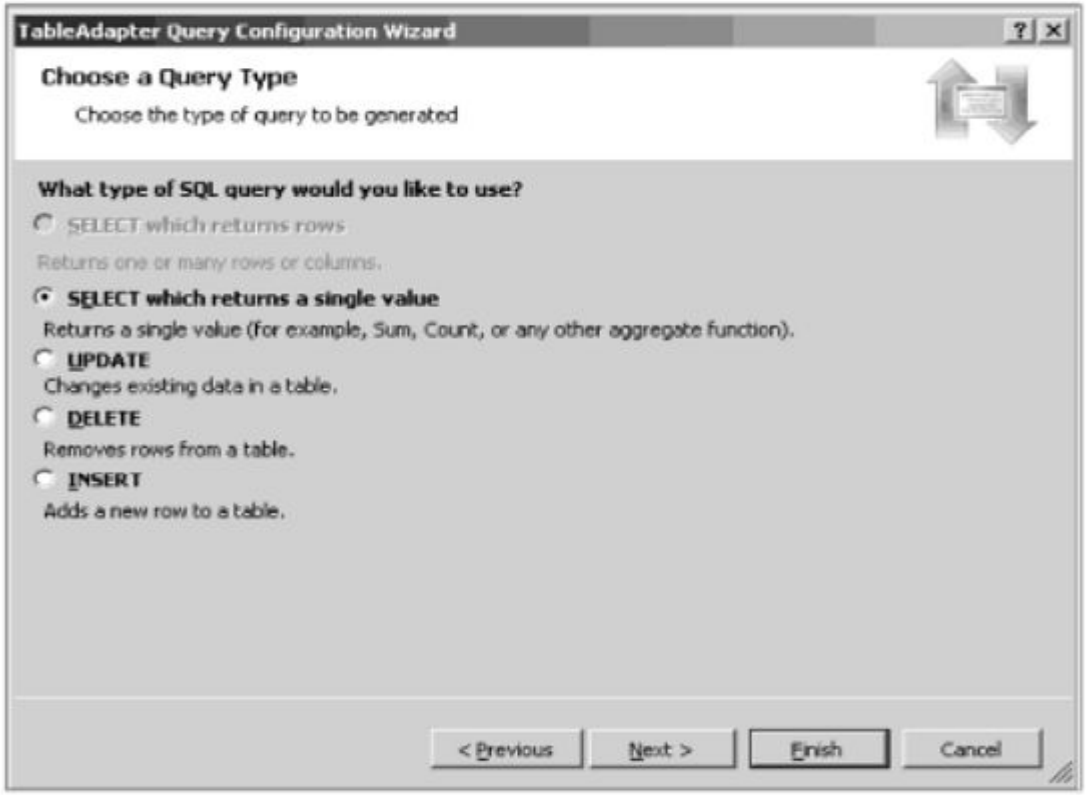
The TableAdapter Query Configuration Wizard

How to Connect to SQL Data Source



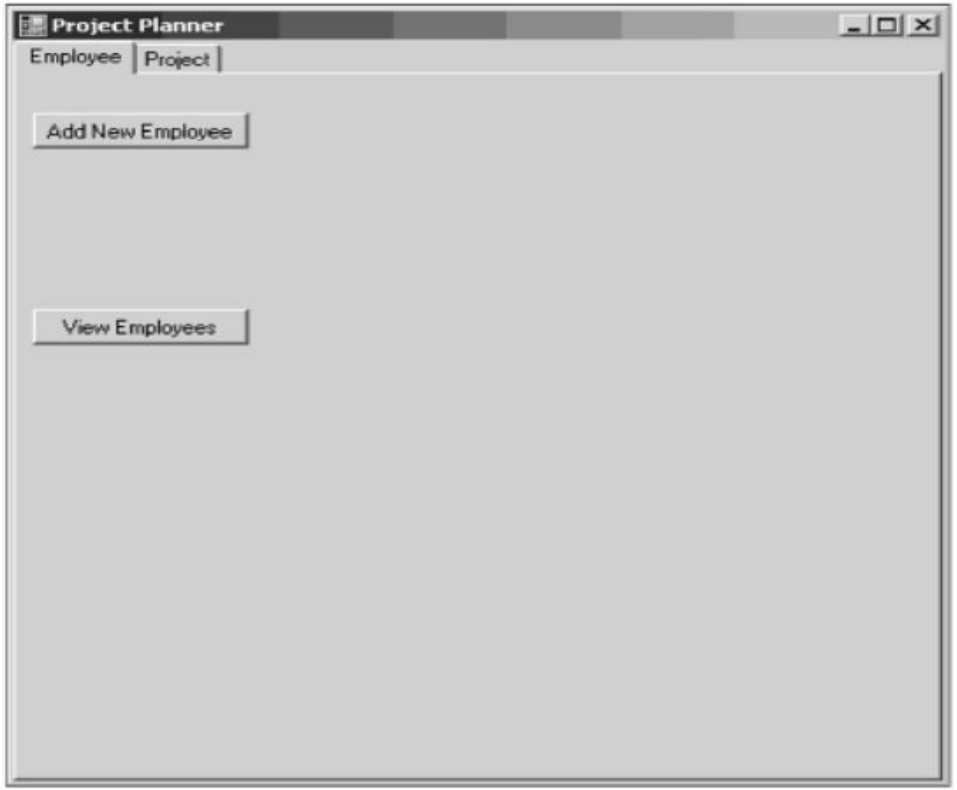
The Choose a Command Type Screen

How to Connect to SQL Data Source



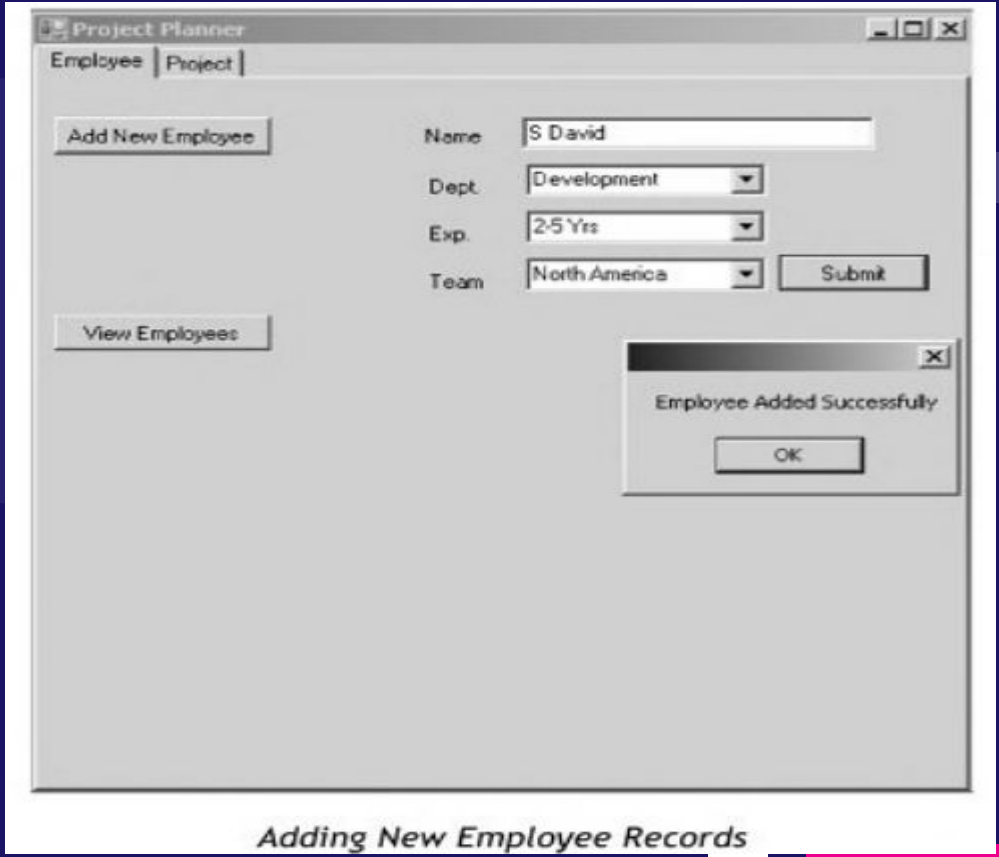
The Choose a Query Type Screen

How to Connect to SQL Data Source



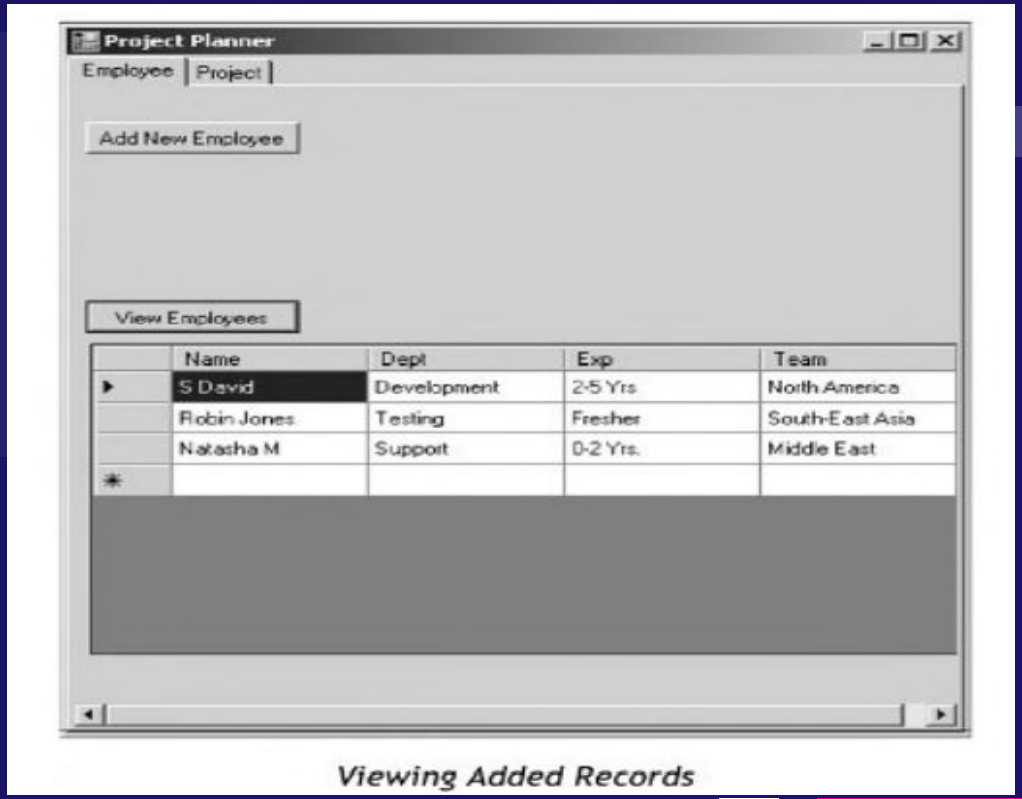
Output of Project Planner Application

How to Connect to SQL Data Source



Adding New Employee Records

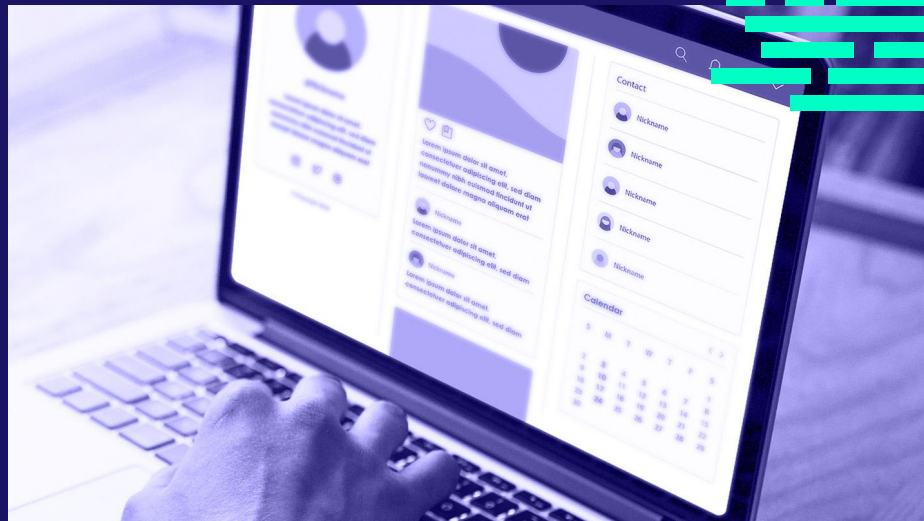
How to Connect to SQL Data Source



Viewing Added Records

How to Connect SQL Database

https://www.c-sharpcorner.com/article/how-to-connect-sql-database-in-asp-net-using-c-sharp-and-insert-and-view-the-data-usi/



Create an employee database and manipulate the records using command objects in ASP.NET (cont..)

WebForm1.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Configuration;
```

Create an employee database and manipulate the records using command objects in ASP.NET (cont..)

```
namespace SQLWebApp
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        SqlConnection con = new
SqlConnection(ConfigurationManager.ConnectionStrings[
"ConnectionString"].ConnectionString);
        protected void Page_Load(object sender,
EventArgs e)
        {
            con.Open();
        }
```

Create an employee database and manipulate the records using command objects in ASP.NET (cont..)

```
protected void Page_Load(object sender, EventArgs e)
        {
            con.Open();
        }

 protected void Button1_Click(object sender, EventArgs e)
        {
            SqlCommand cmd = new SqlCommand("insert into
Emp values('" + TextBox1.Text + "','" + TextBox2.Text + "','" +
TextBox3.Text + "','" + TextBox4.Text + "','" + TextBox5.Text +
"')", con);
```

Create an employee database and manipulate the records using command objects in ASP.NET (cont..)

```
cmd.ExecuteNonQuery();
            con.Close();
            GridView1.DataBind();
            Label1.Visible = true;
            Label1.Text = "Updated!!!";
            TextBox1.Text = "";
            TextBox2.Text = "";
            TextBox3.Text = "";
            TextBox4.Text = "";
            TextBox5.Text = "";

        }
    }
}
```

Deleting the record using command objects in ASP.NET (cont..)

```
SqlCommand cmd = new
SqlCommand("delete from
Emp where Id = " +
TextBox1.Text + " ", con);
```

Updating the record using command objects in ASP.NET (cont..)

```
SqlCommand cmd = new
SqlCommand("update Emp set
Name= '"+TextBox2.Text+"' where
Id =    "+TextBox1.Text+" ", con);
```

Searching the record using command objects in ASP.NET (cont..)

```
SqlCommand cmd = new
SqlCommand("select * from
Emp where Id= " +
TextBox1.Text + " ", con);
```

THANKS !

CREDITS

1. Event Handling in windows programming using C# (c-sharpcorner.com)
2. https://www.oreilly.com/library/view/programming-c/0596001177/ch18.html
3. E.Balagurusamy , "Programming in C#" , Third Edition, Tata McGraw Hill Education Private Limited, New Delhi.