| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Assignment Number:** 5 | |
| **Aim of Assignment:**  Design a webpage to demonstrate connection to database  through ADO.Net | | | |
| **DOP:** 24.4.23 | | **DOS:** 7.5.23 | |
| **CO Mapped:**  CO2 | **PO Mapped:**  PO3, PO5, PSO1, PSO2 | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 5

**Aim:** Design a webpage to demonstrate connection to database

through ADO.Net

**Theory:**

ADO.Net:

ADO is a rich set of classes, interfaces, structures, and

enumerated types that manage data access from various types of

data stores.

ADO.NET provides mainly the following two types of

architectures:

Connected Architecture

Disconnected Architecture

Connected Architecture:

In the connected architecture, connection with a data source

is kept open constantly for data access as well as data

manipulation operations.

The ADO.NET Connected architecture considers mainly

three types of objects.

SqlConnection con;

SqlCommand cmd;

SqlDataReader dr;

Disconnected Architecture:

Disconnected is the main feature of the .NET framework.

ADO.NET contains various classes that support this

architecture. The .NET application does not always stay

connected with the database. The classes are designed in a

way that they automatically open and close the connection.

The data is stored client-side and is updated in the database

whenever required.

The ADO.NET Disconnected architecture considers

primarily the following types of objects:

* DataSet ds;
* SqlDataAdapter da;
* SqlConnection con;
* SqlCommandBuilder bldr;

1. Connection Object and Connection string

Connection Object

1. One of the first ADO.NET objects is the connection object, that allows you to establish a connection to a data source.
2. The connection objects have the methods for opening and closing connections, for beginning a transaction of data.
3. The .Net Framework provides two types of connection classes: The sqlconnection object, that is designed specially to connect to Microsoft SQL Server and the OleDbConnection object, that is designed to provide connection to a wide range of databases, such as Microsoft Access and Oracle.
4. A connection is required to interact with the database. A Connection object helps to identify the database server name, user name and password to connect to the database. The Connection object is used by commands on the database.
5. A Connection object has the responsibility of establishing a connection with the data store.

2. Command Object

A Command object executes SQL statements on the database.

These SQL statements can be SELECT, INSERT, UPDATE, or

DELETE. It uses a connection object to perform these actions

on the database.

A Connection object specifies the type of interaction to perform

with the database, like SELECT, INSERT, UPDATE, or

DELETE.

A Command object is used to perform various types of

operations, like SELECT, INSERT, UPDATE, or DELETE on

the database.

• SELECT

cmd =new SqlCommand(&quot;select \* from Employee&quot;, con);

• INSERT

cmd = new SqlCommand(&quot;INSERT INTO Employee(Emp\_ID,

Emp\_Name) VALUES (&#39;&quot; + aa + &quot;&#39;,&#39;&quot; + bb + &quot;&#39;)&quot;, con);

• UPDATE

SqlCommand cmd =new SqlCommand(&quot;UPDATE Employee

SET Emp\_ID =&#39;&quot; + aa + &quot;&#39;, Emp\_Name =&#39;&quot; + bb + &quot;&#39; WHERE

Emp\_ID = &#39;&quot; + aa + &quot;&#39;&quot;, con);

• DELETE

cmd =new SqlCommand(&quot;DELETE FROM Employee where

Emp\_ID=&#39;&quot; + aa + &quot;&#39;&quot;, con);

• A Command object exposes several execute methods like:

1. ExecuteScaler()

Executes the query, and returns the first column of the first row

in the result set returned by the query. Extra columns or rows are

ignored.

2. ExecuteReader()

Display all columns and all rows in the client-side environment.

In other words, we can say that they display datatables client-

side.

3. ExecuteNonQuery()

Something is done by the database but nothing is returned by the

database.

3. Data Reader Object

A DataReader object is used to obtain the results of a SELECT

statement from a command object. For performance reasons, the

data returned from a data reader is a forward-only stream of

data. This means that the data can be accessed from the stream

in a sequential manner. This is good for speed, but if data needs

to be manipulated then a dataset is a better object to work with.

4. Data Adapter Object

A Data Adapter represents a set of data commands and a

database connection to fill the dataset and update a SQL Server

database.

A Data Adapter contains a set of data commands and a database

connection to fill the dataset and update a SQL Server database.

Data Adapters form the bridge between a data source and a

dataset.

Data Adapters are designed depending on the specific data

source. The following table shows the

Data Adapter classes with their data source.

Data Source

A Data Adapter object accesses data in a disconnected mode. Its

object contains a reference to a connection object.

It is designed in a way that implicitly opens and closes the

connection whenever required.

It maintains the data in a DataSet object. The user can read the

data if required from the dataset and write back the changes in a

single batch to the database. Additionally, the Data Adapter

contains a command object reference for SELECT, INSERT,

UPDATE, and DELETE operations on the data objects and a

data source.

A Data Adapter supports mainly the following two methods:

• Fill ():

The Fill method populates a dataset or a data table object with

data from the database. It retrieves rows from the data source

using the SELECT statement specified by an associated select

command property.

The Fill method leaves the connection in the same state as it

encountered it before populating the data. If subsequent calls to

the method for refreshing the data are required then the primary

key information should be present.

• Update ()

The Update method commits the changes back to the database.

It also analyzes the Row State of each record in the DataSet and

calls the appropriate INSERT, UPDATE, and DELETE

statements. A Data Adapter object is formed between a

disconnected ADO.NET object and a data source.

5. DataSet Object

In the disconnected scenario, the data retrieved from the

database is stored in a local buffer called DataSet. It is explicitly

designed to access data from any data source. This class is

defined in the System.Data namespace.

A Data Set object is an in-memory representation of the data. It

is specially designed to manage data in memory and to support

disconnected operations on data.

A Data Set is a collection of DataTable and DataRelations. Each

DataTable is a collection of DataColumn, DataRows, and

Constraints.

6. Command Builder Object

Automatically generates insert, update, delete queries using the

SelectCommand property of a DataAdapter.

A Command Builder Object is used to build commands for data

modification from objects based on a single table query.

CommandBuilders are designed depending on the specific data

source.

**Code:**

File: WebForm1.aspx

﻿<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs" Inherits="WebApplication1.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

</head>

<body>

<form id="form1" runat="server">

<article>

<h2>Original Order of table games</h2>

<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" AutoGenerateColumns="False" DataKeyNames="id">

<Columns>

<asp:BoundField DataField="id" HeaderText="id" ReadOnly="True" SortExpression="id"></asp:BoundField>

<asp:BoundField DataField="name" HeaderText="name" SortExpression="name"></asp:BoundField>

<asp:CheckBoxField DataField="is\_outdoor" HeaderText="is\_outdoor" SortExpression="is\_outdoor"></asp:CheckBoxField>

<asp:BoundField DataField="pcount" HeaderText="pcount" SortExpression="pcount"></asp:BoundField>

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString='<%$ ConnectionStrings:collGamesOgConnectionString %>' SelectCommand="SELECT \* FROM [ajayk57\_games]"></asp:SqlDataSource>

</article>

<article>

<h2>Table games ordered by player count(ascending)</h2>

<asp:GridView ID="GridView2" runat="server" DataSourceID="SqlDataSource2" AutoGenerateColumns="False" DataKeyNames="id">

<Columns>

<asp:BoundField DataField="id" HeaderText="id" ReadOnly="True" SortExpression="id"></asp:BoundField>

<asp:BoundField DataField="name" HeaderText="name" SortExpression="name"></asp:BoundField>

<asp:CheckBoxField DataField="is\_outdoor" HeaderText="is\_outdoor" SortExpression="is\_outdoor"></asp:CheckBoxField>

<asp:BoundField DataField="pcount" HeaderText="pcount" SortExpression="pcount"></asp:BoundField>

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource2" runat="server" ConnectionString='<%$ ConnectionStrings:collGamesOgConnectionString %>' SelectCommand="SELECT \* FROM [ajayk57\_games] ORDER BY [pcount]"></asp:SqlDataSource>

</article>

</form>

<footer>

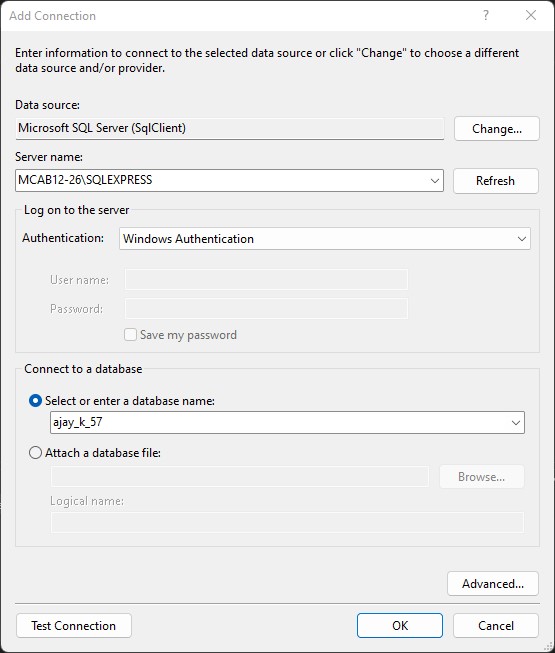
Author: Ajay Karthikesan, Roll No. : 57

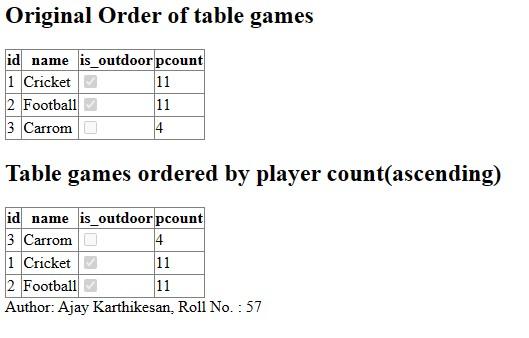
</footer>

</body>

</html>

**Output:**





**Conclusion:**

Successfully designed a webpage to demonstrate connection to

database through ADO.Net