

[articles](#) [Q&A](#) [forums](#) [stuff](#) [lounge](#) [?](#)

Search for articles, questions, tips

[Follow](#)

# Three Layer Architecture in C# .NET

**Parikshit Patel**, 25 Feb 2014

4.45 (191 votes)

Rate:

This article describes a typical three layer architecture in C# .NET. It is a very useful approach for coding due to easy code maintenance.

[Download source - 35.28 KB](#)

## Introduction

Here in this article, I would like to cover the typical three layer architecture in C# .NET. It is a very useful approach for coding due to easy code maintenance.

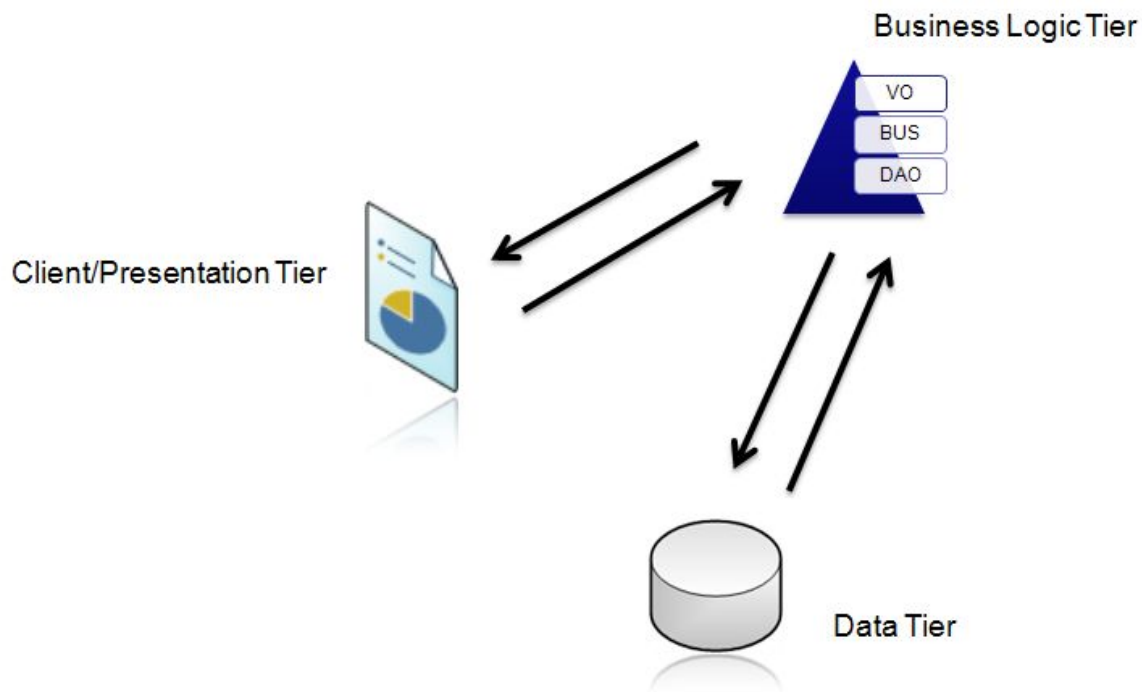
## Overview

First let me give you a small overview about the topic I would like to cover in this article.

1. Tier vs. Layer
2. Three Tier/Layer Architecture Design Components
3. Demo: Three Layer Windows Application in C#.NET

### 1. Tier vs. Layer

1.1 Tier: Tier indicates a physical separation of components, which may mean different assemblies such as DLL, EXE, etc. on the same server or multiple servers.



As you can see in the above figure, Data Tier have no direction with Presentation Tier, but there is an intermediate Tier called Business Tier which is mainly responsible to pass the data from Data Tier to Presentation Tier and to add defined business logic to Data.

So, if we separate each Tier by its functionality, then we come to know the below conclusion:



### Presentation Tier

- The presentation tier is the tier in which users interact with an application.



### Middle Tier

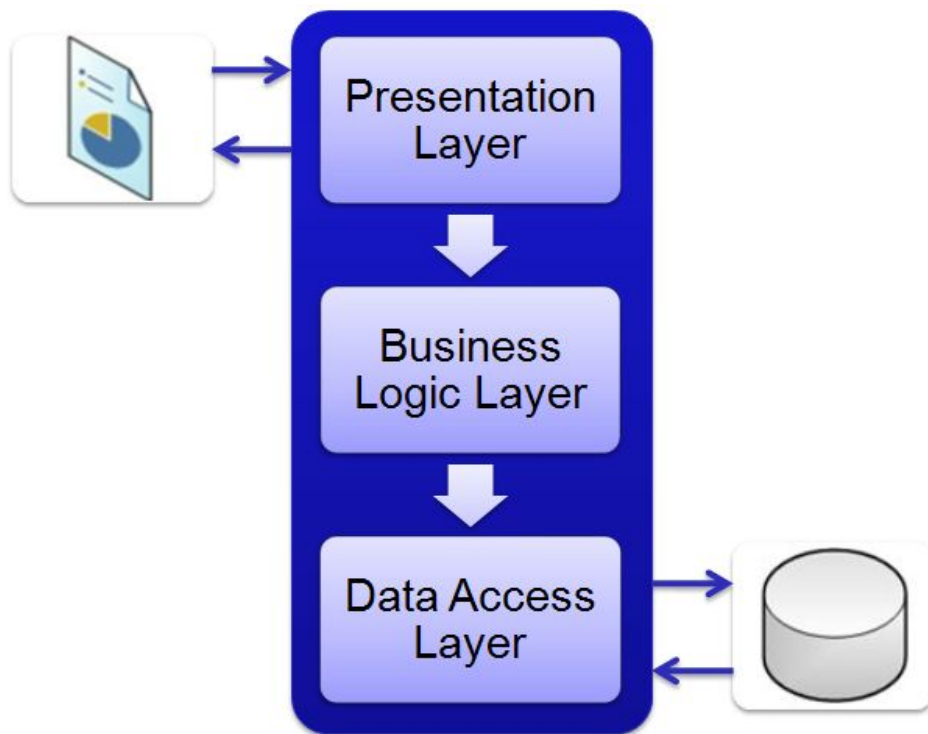
- The middle tier is the layer that the presentation tier and the data tier use to communicate with each other.



### Data Tier

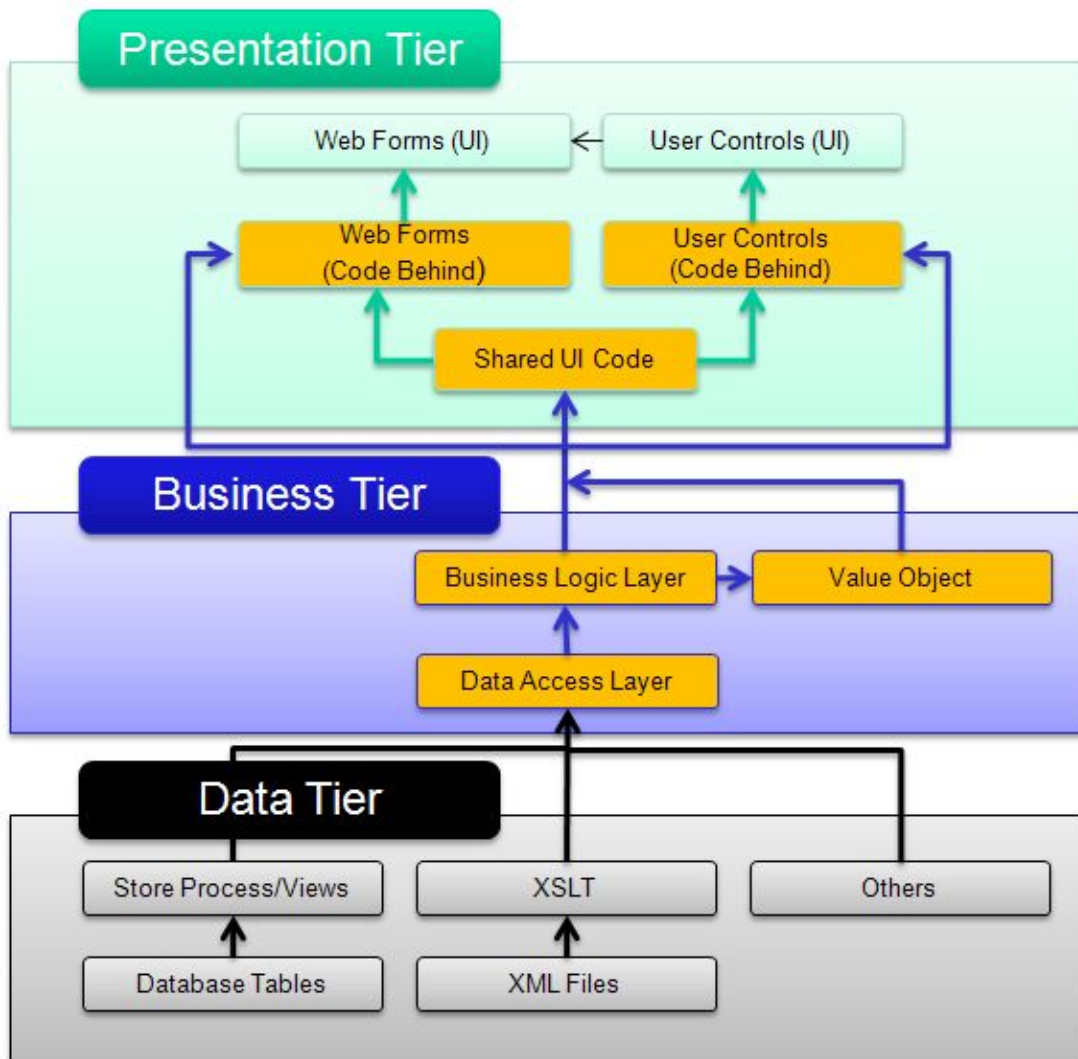
- The data tier is basically the server that stores an application's data.

1.2 Layer: Layer indicates logical separation of components, such as having distinct namespaces and classes for the Database Access Layer, Business Logic Layer and User Interface Layer.

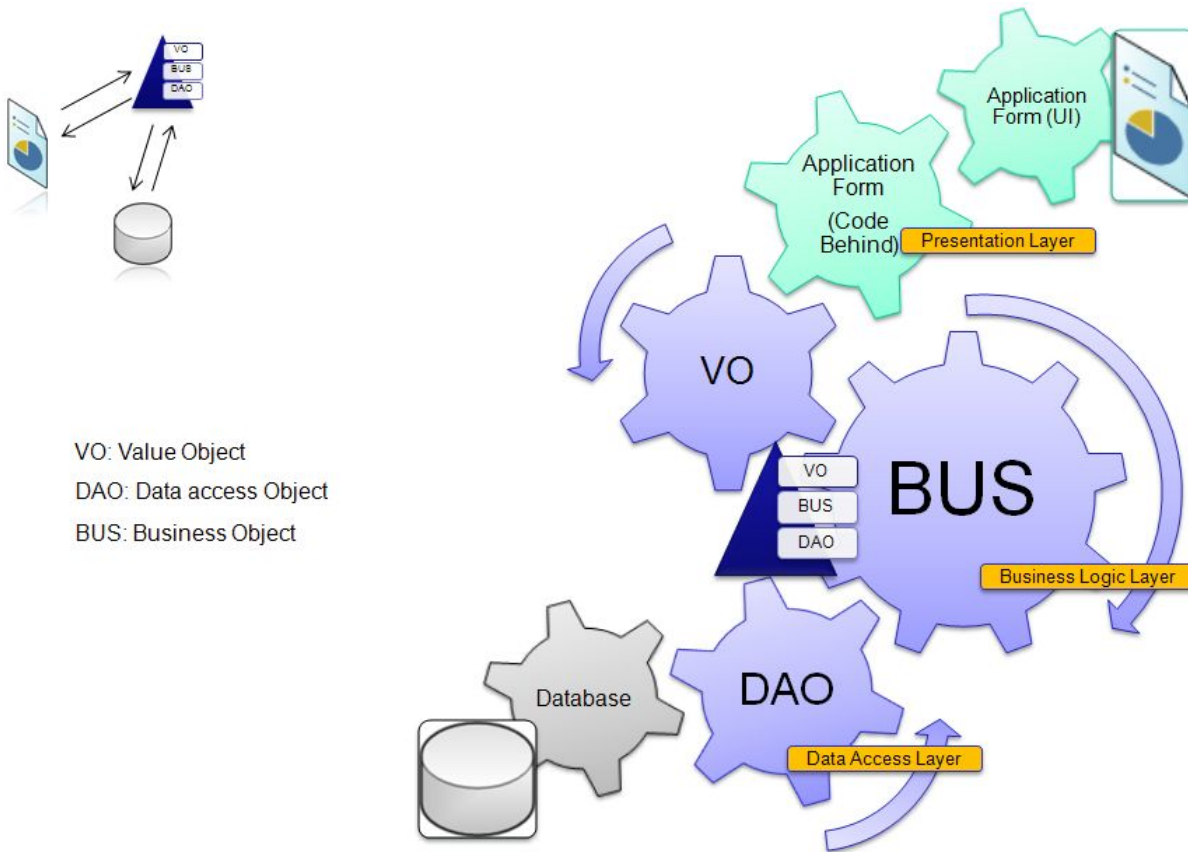


## 2. Three Tier/Layer Architecture Design Components

As we have already seen, tier is the sum of all the physical components. We can separate the three tiers as Data Tier, Business Tier and Presentation Tier.



- Data Tier is basically the server which stores all the application's data. Data tier contents Database Tables, XML Files and other means of storing Application Data.
- Business Tier is mainly working as the bridge between Data Tier and Presentation Tier. All the Data passes through the Business Tier before passing to the presentation Tier. Business Tier is the sum of Business Logic Layer, Data Access Layer and Value Object and other components used to add business logic.
- Presentation Tier is the tier in which the users interact with an application. Presentation Tier contents Shared UI code, Code Behind and Designers used to represent information to user.

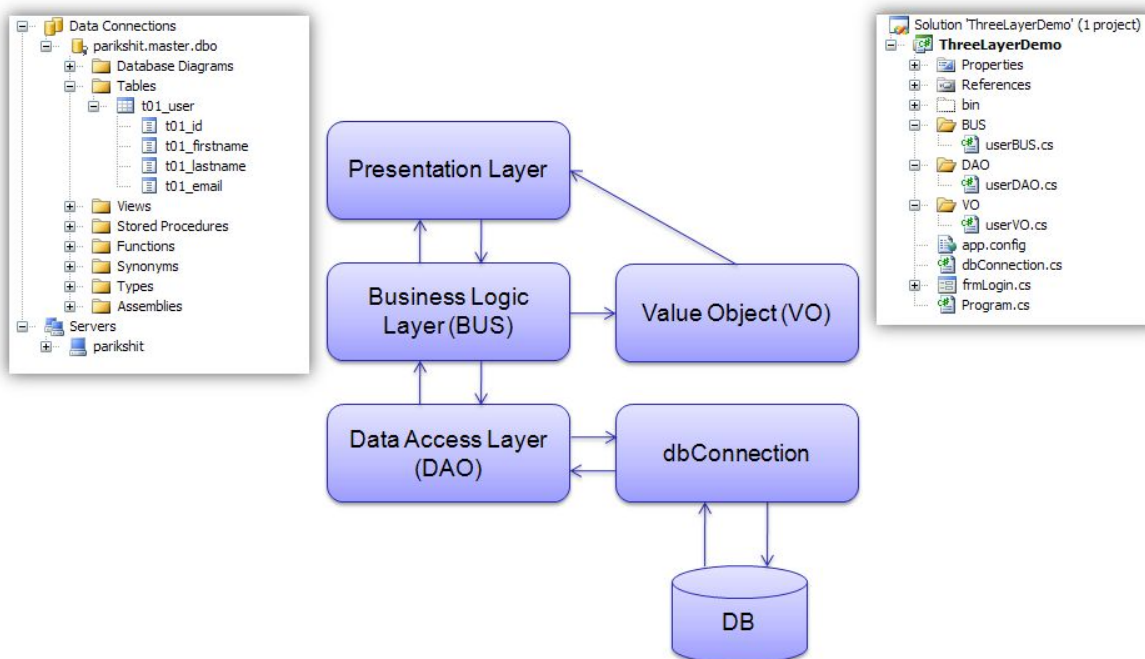


The above figure is a mixture of Three Tier and Three Layer Architecture. Here, we can clearly see a different between Tier and Layer. Since each component is independent of each other, they are easily maintainable without changing the whole code.

This approach is really very important when several developers are working on the same project and some module needs to be re-used in another project. In a way, we can distribute work among developers and also maintain it in the future without much problems.

Testing is also a very important issue for Architecture when we are considering writing a test case for the project. Since it's like a modular architecture, it's very handy testing each module and to trace out bugs without going through the entire code.

### 3. Demo: 3 Layer Windows Application in C#.NET



Let's go though from one module to other to have a better understanding of it.

## dbConnection

This class is mainly used to do the database activity like Select, Update and Delete query to database. It also checks if the database connection is open or not. If database connection is not open, then it opens the connection and performs the database query. The database results are to be received and being passing in Data Table in this class.

This class takes the database setting from the app.config file so it's really flexible to manage the database settings.

Hide Shrink ▲ Copy Code

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;

namespace ThreeLayerDemo.Core
{
    public class dbConnection
    {
        private SqlDataAdapter myAdapter;
        private SqlConnection conn;

        /// <constructor>
        /// Initialise Connection
        /// </constructor>
        public dbConnection()
        {
            myAdapter = new SqlDataAdapter();
            conn = new SqlConnection(ConfigurationManager.ConnectionStrings
                ["dbConnectionString"].ConnectionString);
        }

        /// <method>
        /// Open Database Connection if Closed or Broken
        /// </method>
        private SqlConnection openConnection()
        {
            if (conn.State == ConnectionState.Closed || conn.State ==
                ConnectionState.Broken)
            {
                conn.Open();
            }
            return conn;
        }

        /// <method>
        /// Select Query
        /// </method>
        public DataTable executeSelectQuery(String _query, SqlParameter[] sqlParameter)
        {
            SqlCommand myCommand = new SqlCommand();
            DataTable dataTable = new DataTable();
            dataTable = null;
            DataSet ds = new DataSet();
            try
            {
                myCommand.Connection = openConnection();
                myCommand.CommandText = _query;
                myCommand.Parameters.AddRange(sqlParameter);
                myCommand.ExecuteNonQuery();
                myAdapter.SelectCommand = myCommand;
                myAdapter.Fill(ds);
                dataTable = ds.Tables[0];
            }
        }
    }
}
```

```

        catch (SqlException e)
        {
            Console.WriteLine("Error - Connection.executeSelectQuery - Query: " + _query + " \nException: " + e.StackTrace.ToString());
            return null;
        }
        finally
        {
        }
        return dataTable;
    }

    /// <method>
    /// Insert Query
    /// </method>
    public bool executeInsertQuery(String _query, SqlParameter[] sqlParameter)
    {
        SqlCommand myCommand = new SqlCommand();
        try
        {
            myCommand.Connection = openConnection();
            myCommand.CommandText = _query;
            myCommand.Parameters.AddRange(sqlParameter);
            myAdapter.InsertCommand = myCommand;
            myCommand.ExecuteNonQuery();
        }
        catch (SqlException e)
        {
            Console.WriteLine("Error - Connection.executeInsertQuery - Query: " + _query + " \nException: \n" + e.StackTrace.ToString());
            return false;
        }
        finally
        {
        }
        return true;
    }

    /// <method>
    /// Update Query
    /// </method>
    public bool executeUpdateQuery(String _query, SqlParameter[] sqlParameter)
    {
        SqlCommand myCommand = new SqlCommand();
        try
        {
            myCommand.Connection = openConnection();
            myCommand.CommandText = _query;
            myCommand.Parameters.AddRange(sqlParameter);
            myAdapter.UpdateCommand = myCommand;
            myCommand.ExecuteNonQuery();
        }
        catch (SqlException e)
        {
            Console.WriteLine("Error - Connection.executeUpdateQuery - Query: " + _query + " \nException: " + e.StackTrace.ToString());
            return false;
        }
        finally
        {
        }
        return true;
    }
}
}
}

```

## Database Access Layer

Database Access Layer (DAO) builds the query based on received parameters from the Business Logic Layer and passes it the **dbConnection** class for execution. And simple return results from the **dbConnection** class to Business Logic Layer.

[Hide](#) [Shrink](#) ▲ [Copy Code](#)

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace ThreeLayerDemo.Core
{
    public class UserDAO
    {
        private dbConnection conn;

        /// <constructor>
        /// Constructor UserDAO
        /// </constructor>
        public UserDAO()
        {
            conn = new dbConnection();
        }

        /// <method>
        /// Get User Email By Firstname or Lastname and return DataTable
        /// </method>
        public DataTable searchByName(string _username)
        {
            string query = string.Format("select * from [t01_user]
            where t01_firstname like @t01_firstname or t01_lastname
            like @t01_lastname ");
            SqlParameter[] sqlParameters = new SqlParameter[2];
            sqlParameters[0] = new SqlParameter("@t01_firstname", SqlDbType.VarChar);
            sqlParameters[0].Value = Convert.ToString(_username);
            sqlParameters[1] = new SqlParameter("@t01_lastname", SqlDbType.VarChar);
            sqlParameters[1].Value = Convert.ToString(_username);
            return conn.executeSelectQuery(query, sqlParameters);
        }

        /// <method>
        /// Get User Email By Id and return DataTable
        /// </method>
        public DataTable searchById(string _id)
        {
            string query = "select * from [t01_id] where t01_id = @t01_id";
            SqlParameter[] sqlParameters = new SqlParameter[1];
            sqlParameters[0] = new SqlParameter("@t01_id", SqlDbType.VarChar);
            sqlParameters[0].Value = Convert.ToString(_id);
            return conn.executeSelectQuery(query, sqlParameters);
        }
    }
}
```

## Value Object

Value Object is nothing more but a class with the contents **GET** and **SET** methods. It's mainly used to pass Data from one class to another. It's directly connected with Business Logic Layer and Presentation Layer. As you can see in the diagram object values are being SET in Business Logic Layer and GET from Presentation Layer.

[Hide](#) [Shrink](#) ▲ [Copy Code](#)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```



```
namespace ThreeLayerDemo.Core
{
    public class UserVO
    {
        private int _idUser;
        private string _firstname;
        private string _lastname;
        private string _email;

        /// <constructor>
        /// Constructor UserVO
        /// </constructor>
        public UserVO()
        {
            //
            // TODO: Add constructor logic here
            //
        }

        public int idUser
        {
            get
            {
                return _idUser;
            }

            set
            {
                _idUser = value;
            }
        }

        public string firstname
        {
            get
            {
                return _firstname;
            }

            set
            {
                _firstname = value;
            }
        }

        public string lastname
        {
            get
            {
                return _lastname;
            }

            set
            {
                _lastname = value;
            }
        }

        public string email
        {
            get
            {
                return _email;
            }

            set
            {
                _email = value;
            }
        }
    }
}
```

```

    }
}

```

## Business Logic Layer

Business Logic Layer (BUS) works as a bridge between Presentation Layer and DAO. All the user values received from the presentation layer are being passed to BUS. The results received from the DAO are in row data in Data Table format but in BUS it's converting into Value Objects (VO). Business Logic Layer (BUS) is the most important class in the whole architecture because it mainly contains all the business logic of the program. Whenever a user wants to update the business logic of the program only need to update this class.

[Hide](#) [Shrink](#) [Copy Code](#)

```

using System;
using System.Collections.Generic;
using System.Text;
using System.Data;

namespace ThreeLayerDemo.Core
{
    /// <summary>
    /// Summary description for UserBUS
    /// </summary>
    public class UserBUS
    {
        private UserDAO _userDAO;

        /// <constructor>
        /// Constructor UserBUS
        /// </constructor>
        public UserBUS()
        {
            _userDAO = new UserDAO();
        }

        /// <method>
        /// Get User Email By Firstname or Lastname and return VO
        /// </method>
        public UserVO getUserEmailByName(string name)
        {
            UserVO userVO = new UserVO();
            DataTable dataTable = new DataTable();

            dataTable = _userDAO.searchByName(name);

            foreach (DataRow dr in dataTable.Rows)
            {
                userVO.idUser = Int32.Parse(dr["t01_id"].ToString());
                userVO.firstname = dr["t01_firstname"].ToString();
                userVO.lastname = dr["t01_lastname"].ToString();
                userVO.email = dr["t01_email"].ToString();
            }
            return userVO;
        }

        /// <method>
        /// Get User Email By Id and return DataTable
        /// </method>
        public UserVO getUserById(string _id)
        {
            UserVO userVO = new UserVO();
            DataTable dataTable = new DataTable();
            dataTable = _userDAO.searchById(_id);

            foreach (DataRow dr in dataTable.Rows)
            {
                userVO.idUser = Int32.Parse(dr["t01_id"].ToString());
                userVO.firstname = dr["t01_firstname"].ToString();
            }
        }
    }
}

```

```

        userV0.lastname = dr["t01_lastname"].ToString();
        userV0.email = dr["t01_email"].ToString();
    }
    return userV0;
}
}
}

```

## Presentation Layer

Presentation Layer is the only layer which is directly connected with the user. So in this matter, it's also a really important layer for marketing purposes. Presentation Layer is mainly used for getting user data and then passing it to Business Logic Layer for further procedure, and when data is received in Value Object then it's responsible to represent value object in the appropriate form which user can understand.

[Hide](#) [Shrink](#) ▲ [Copy Code](#)

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ThreeLayerDemo.Core;

namespace ThreeLayerDemo
{
    public partial class frmLogin : Form
    {
        private UserBUS _userBUS;

        public frmLogin()
        {
            InitializeComponent();
            _userBUS = new UserBUS();
        }

        private void btnSearch_Click(object sender, EventArgs e)
        {
            UserVO _userVO = new UserVO();
            _userVO = _userBUS.getUserEmailByName(txtUsername.Text);
            if (_userVO.email == null)
                MessageBox.Show("No Match Found!", "Not Found",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            else
                MessageBox.Show(_userVO.email, "Result",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}

```

## Summary

Hope this explanation helps the beginner specially looking for a generic approach. There are also some methods which are far better than the architecture described above, mostly with skipping Database Access Layer and Value Object Class, and making it dynamically which is really handy for maintenance in case of frequent database change. I will try to post some in the near future. 😊

## Revision

- 31.05.2009: Initial version
- 02.06.2009: Using parameterized SQL queries
  - **dbConnection** updated
  - **UserDAO** updated

## License

This article, along with any associated source code and files, is licensed under [The Code Project Open License \(CPOl\)](#)


## Share

[TWITTER](#)[FACEBOOK](#)

## About the Author



### Parikshit Patel

Engineer  
Germany 

[Follow  
this Member](#)

No Biography provided

## You may also be interested in...

[Public, Private, and Hybrid Cloud: What's the difference?](#)

[Machine Learning in Excel](#)

[A Task Management System using Three Layer Architecture](#)

[Implement Observer Pattern in .NET \(3 Techniques\)](#)

[Remoting Architecture in .NET](#)

[Understanding Three Layer Architecture and its Implementation in C# .NET](#)

# Comments and Discussions





[First](#) [Prev](#) [Next](#)

**Where is a local database?**

**Jordan Nash** 10-Dec-18 16:25

**When working with service references**

**turkush\_** 15-Apr-18 4:30

**procedures**

**Elham Kohestani** 20-Oct-17 22:40

Re: procedures

**Роман Саченок** 6-Dec-17 21:25

**Static Class**

**Member 11968598** 7-Apr-17 20:17

**Can we skip/bypass the intermediate layer**

**Ashish Shuklaa** 23-Mar-17 0:26

**Awesome**

**EnzoDuvallle** 25-Oct-16 3:30

**table column name change requires changes in both BLL and DAL**

**Member 10508775** 20-Oct-16 15:38

**My vote of 4**

**Ujjval Shukla** 28-Aug-15 20:28

**The arrow on your VO gear is backwards.....**

**Foosfam** 15-Aug-15 23:43

Re: The arrow on your VO gear is backwards.....

**PIEBALDconsult** 15-Aug-15 23:45

Re: The arrow on your VO gear is backwards.....

**Foosfam** 16-Aug-15 18:08

Re: The arrow on your VO gear is backwards.....

**PIEBALDconsult** 16-Aug-15 22:44

Re: The arrow on your VO gear is backwards.....

**Foosfam** 17-Aug-15 2:40

**Three layer Architecture**

**Member 10355710** 11-Dec-14 5:40

**where does validation logic go?** **tyler.durden404** 16-Oct-14 18:45

Re: where does validation logic go?

**Member 11317809** 12-Feb-15 1:31**Great ....** **rasikasamith** 11-Oct-14 19:51**Why not use OV in DataAccess and pass by OV?** **kanlei** 22-May-14 13:47**Thanks for sharing! However...** **dicktse** 1-Apr-14 22:33**Where is the Database?** **Antonio Barros** 7-Mar-14 21:55**Excellent Article** **Suchi Banerjee, Pune** 3-Mar-14 18:38**My vote of 3** **KarstenK** 3-Mar-14 16:49**My vote of 1** **Member 10501366** 26-Feb-14 14:32

Re: My vote of 1

**Eddy Vluggen** 28-Feb-14 20:13[Refresh](#)[1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [Next »](#)[General](#) [News](#) [Suggestion](#) [Question](#) [Bug](#) [Answer](#) [Joke](#) [Praise](#) [Rant](#) [Admin](#)

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.

[Permalink](#) | [Advertise](#) | [Privacy](#) | [Cookies](#) | [Terms of Use](#) | [Mobile](#)  
Web05 | 2.8.181222.1 | Last Updated 24 Feb 2014Select Language   
Layout: [fixed](#) | [fluid](#)Article Copyright 2009 by Parikshit Patel  
Everything else Copyright © [CodeProject](#), 1999-2018