Understanding Multilayered Architecture in .NET

Akhil Mittal Apr 14, 2016 127.6k 8 11

MultiLayeredArchitecture.zip
Download Free .NET & JAVA Files API

Introduction

This article focuses on understanding a basic multilayered architecture in C#.

The article starts with an introduction to one tier, two tier and n-tier architectures, their pros and cons, and later describes how to do a simple basic multilayered architecture in .Net.

My effort in this article will be to focus on next level programming in .Net for developing an enterprise application.

Layered Architecture

When the various components in a system are organized systematically we call it a system architecture. The architecture is the enterprise-scale division of a system into layers or tiers, each having responsibility for a major part of the system and with as little direct influence on other layers.

One, two, three and n-tier applications

There are plenty of ways for a system to be be split into multiple logical tiers.

One-tier applications

Single-tier applications are basically simple standalone programs.

It's not necessary to consider network communication and the risk of network failure

in these cases since they do not require network access.

Since all the data resides within the same application, these programs do not focus on synchronization of data. When separating the tiers physically the application is slower since the communication over the network will result in a loss of performance, therefore one-tier applications certainly have better performance.

. .

A two-tier application, in comparison to the one-tier application as described, does not combine all functions into a single process but into separate functions. For example a chat application. This kind of application contains two separated tiers, client and a server.

The client has the responsibility of capturing user input and displaying the actual messages.

The server is responsible of the communication between the people that use the chat client.

Three-tier applications

A three-tier application adds another tier to the previous mentioned chat application, this could be in the form of a database. One could think of a three-tier application as a dynamic web application, which has a user interface, business logic, services and a database each placed in different tiers, as illustrated in Figure 1. As mentioned in the previous section a two-tier architecture separates the user interface from the business logic, in the same way the three tier architecture separates the database from the business logic.

MulLayer1.jpg
Figure 1 - Three-tier application

N-tier applications

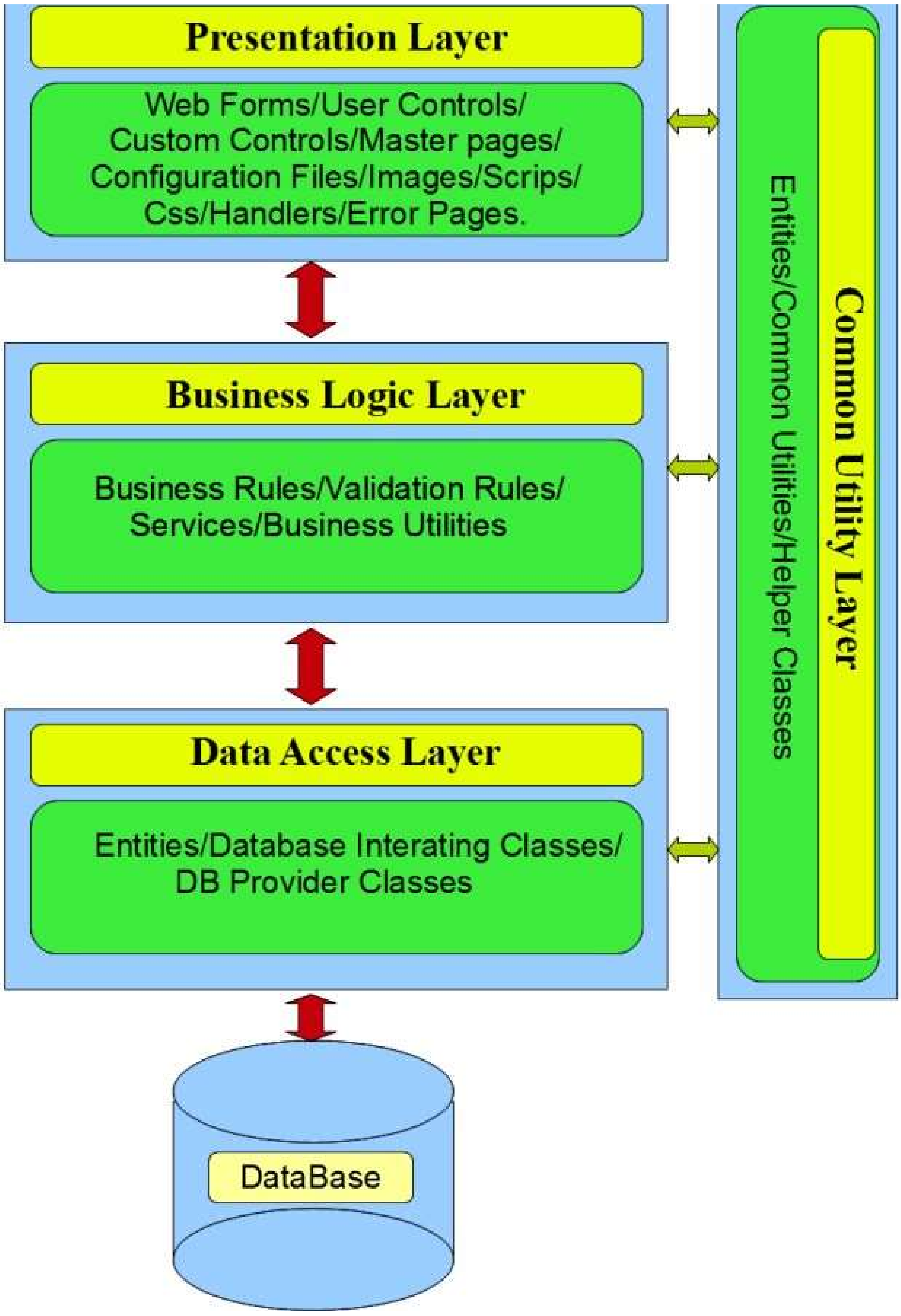
A logical n-tier application is an application where all logical parts are separated into discrete classes. In a typical business application, this generally involves a Presentation Layer, Business Logic Layer and a Data Access Layer. This separation makes the application easier to maintain. The advantages of this architecture are that all business rules are centralized that make them easy to create, use and re-use. The data access is also centralized, that has the same advantage as the centralization of the business rules. Centralizing the data access routines are also good when it comes to maintenance since changes must only be implemented in one location. There are really not that many disadvantages of this kind of architecture, however, it takes a bit longer to get up and running since several separate components need to be developed, that also might make the code a bit more complicated to grasp for less experienced developers.

A Practical Approach

Let's start developing a simple multilayered architecture in .Net. I use C# as my programming language, however the programming language is not a barrier at all, one can choose whatever the developer prefers. The architecture that we implement has the following design as mentioned in Figure 2. We will start creating the architecture for a web application, later it could be converted into a Windows application too.

one Business Logic Layer class library, one Data Access Layer class library and one Common Layer class library can be included in the solution.

Let's follow the implementation step-by-step.


Presentation Layer
Web Forms/User Controls/ Custom Controls/Master pages/ Configuration Files/Images/Scrips/ Css/Handlers/Error Pages.
Business Logic Layer
Business Rules/Validation Rules/ Services/Business Utilities
Data Access Layer
Entities/Database Interating Classes/ DB Provider Classes
Common Utility Layer
Entities/Common Utilities/Helper Classes
DataBase

Step 1: Add a Web Project (Presentation Layer).

Open your Visual Studio and add a simple website to the solution, name it Presentation Layer.

MulLayer3.jpg

Your Development Environment may look like:

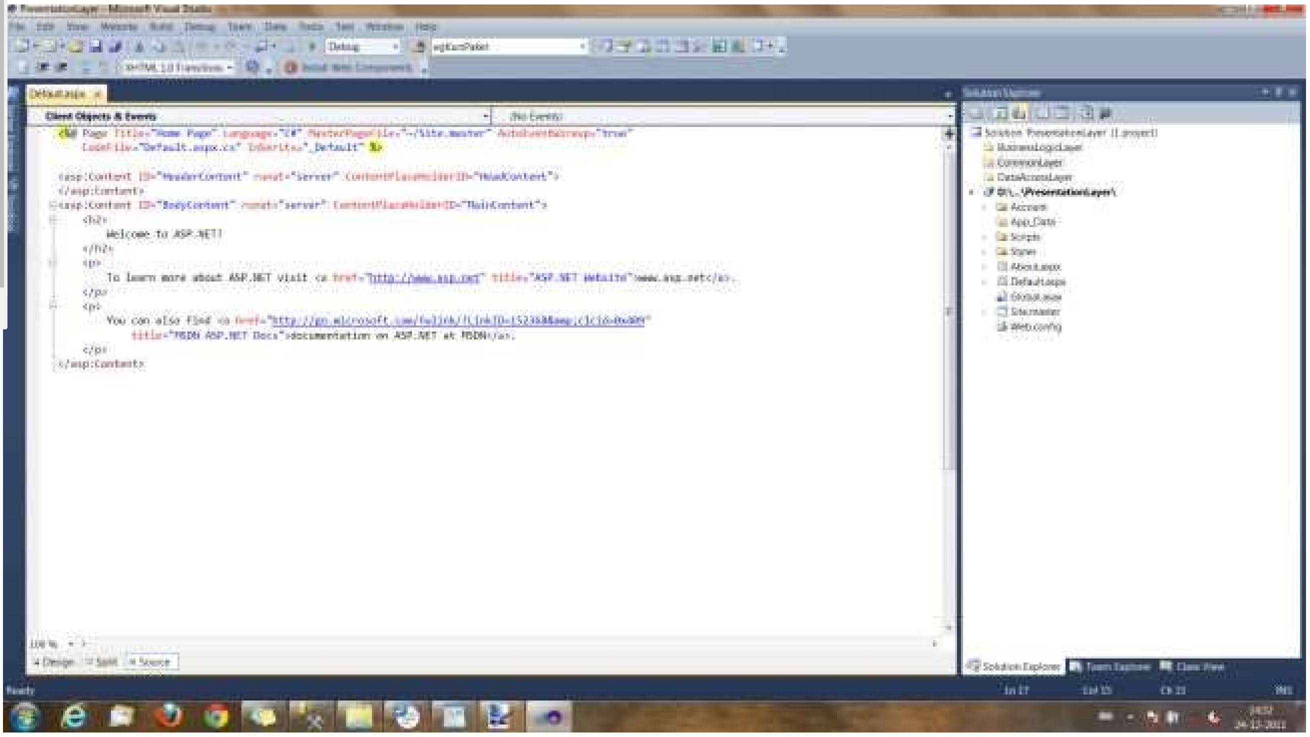
MulLayer4.jpg

Our Presentation Layer is the Web application, that will be exposed to the end user.

The Web application includes Web Forms, in other words aspx pages and User Controls, in other words Ascx pages.

Scripts (client side JavaScripts), Styles (CSS and custom styles for styling the page), Master Page (.master extension, for providing common functionality to a group of desired pages, Configuration Files like web.config and app.config and so on).

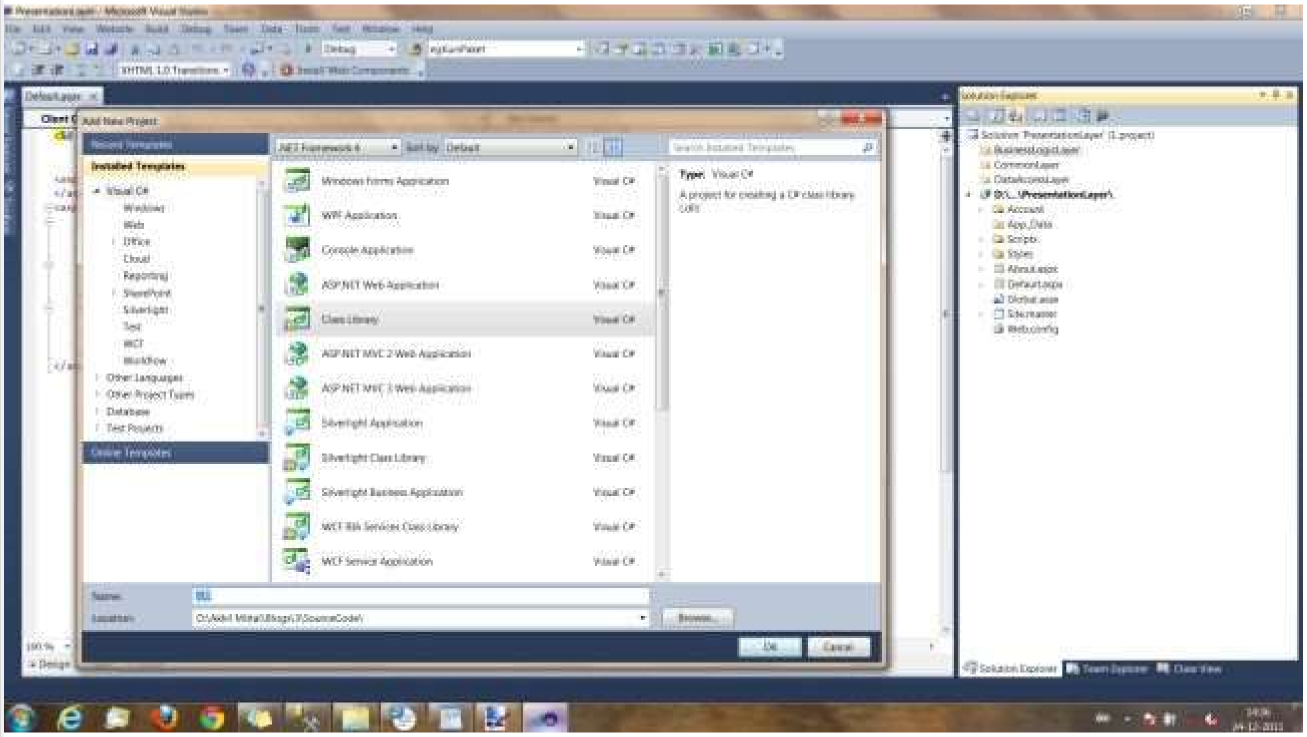
Let's set up our next projects and define them in separate layers. Add three folders to your solution named BusinessLogicLayer, DataAccessLayer and CommonLayer. Your solution will look as in the following:



```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
    CodeFile="Default.aspx.cs" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET Website">www.asp.net</a>.
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?LinkID=152368&amp;clcid=0x409"
            title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
    </p>
</asp:Content>
```

Right-click the Business Logic Layer folder and add a C# class library project to it, call it BLL.csproj.

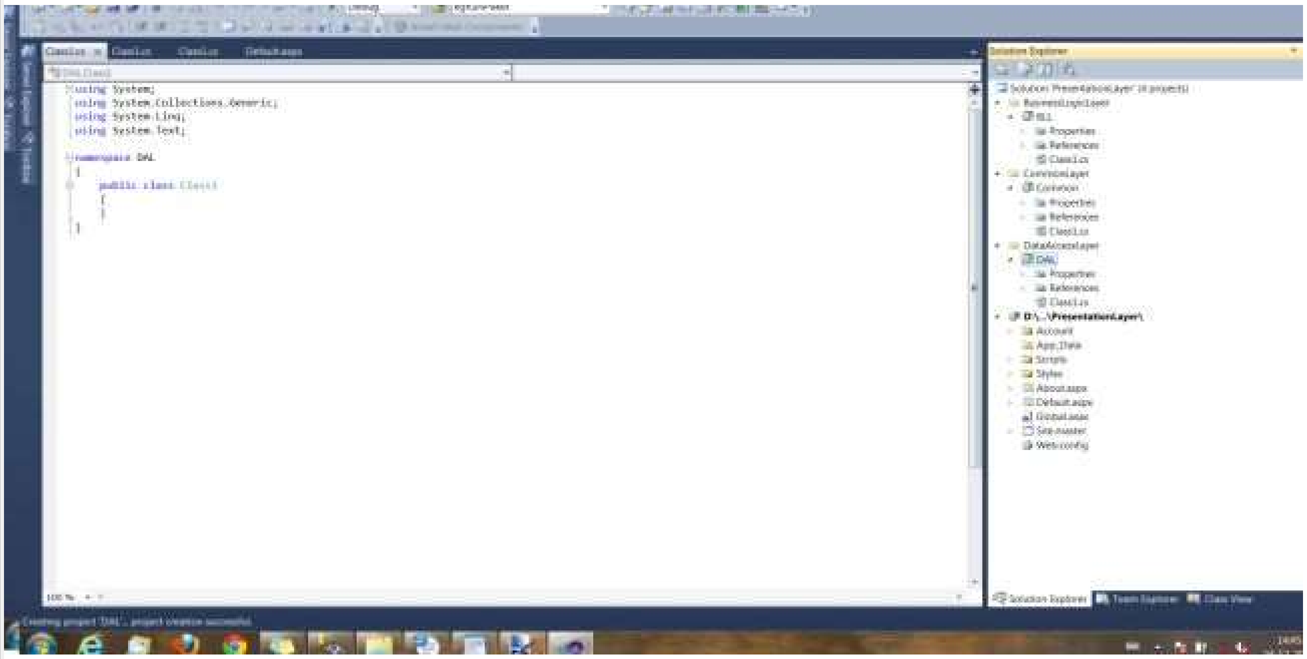


Doing this will add a C# class library project to our solution in the Business Logic Layer Folder.

Likewise add two more projects to the Common Layer and DataAccessLayer folder respectively and call them Common.csproj and DAL.csproj projects.

The Data Access Layer Project Contains the entities classes and objects to communicate with the database, whereas our common project contains properties and helper classes to communicate with all the three layers commonly. It contains the objects to be ed to and from the Presentation Layer to the Data Access Layer commonly, and also acts as a mode of communication among the layers.

Now our solution looks as in the following:



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

namespace DAL
{
    public class Class1
    {
    }
}
```

Step 3: Create a database with a sample table and put some default values in it, for example I have created a database named "EkuBase" and created a simple Student table with the following fields, StudentId, Name, Email, Address, Age, Country. The Create script of the table is as follows:

```
    USE [EkuBase]
GO

/****** Object:  Table [dbo].[Student]    Script Date: 12/24/2011 14:53:14 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TABLE [dbo].[Student](
    [StudentID] [int] NOT NULL,
    [Name] [nvarchar](50) NULL,
    [Email] [nvarchar](50) NULL,
    [Address] [nvarchar](50) NULL,
    [Age] [int] NULL,
    [Country] [nvarchar](50) NULL,
 CONSTRAINT [PK_Student] PRIMARY KEY CLUSTERED
(
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Therefore making studentid as the primary key.

Step 4: Let's add classes to our projetcs, Add StudentBLL.cs, StudentDAL.cs, StudentEntity.cs to BLL, DAL and Common Layer respectively. Ensure you qualify them with logical namespaces, so that it's easy to recognize and use them.

BLL:

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

namespace BLL
{
    public class StudentBLL
    {
    }
}
```

DAL:

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

namespace DAL
{
    public class StudentDAL
    {
    }
}
```

Common:

```
using System.Linq;
using System.Text;

namespace Common
{
    public class StudentEntity
    {
    }
}
```

Step 5: Add a Connection String to web.config and a Helper class to the DAL to interact with the database. In my case I am using SQL helper for ease.

Web.Config:

```
"
            <connectionStrings>

   <add name="Connections" connectionString="Data Source=69B5ZR1\SQLEXPRESS;Initial
   Catalog=EkuBase;Persist Security Info=True;User ID=akhil;word=akhil"
   providerName="System.Data.SqlClient"/>
               </connectionStrings>
```

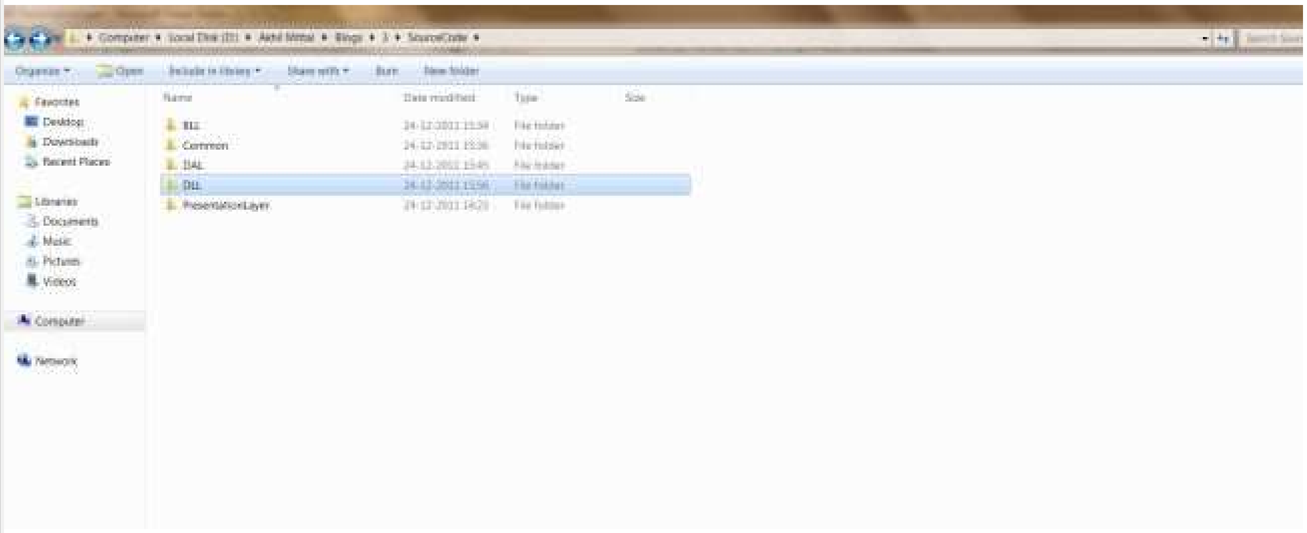
And define the Connection String in your SQL Helper so that you don't need to again and again create a connection and it to your method when you interact with the database.

```
public static readonly string CONN_STRING =
ConfigurationManager.ConnectionStrings["Connections"].ConnectionString;
```

Add a reference to System.Configuration to the DAL project before reading the preceding connection string.

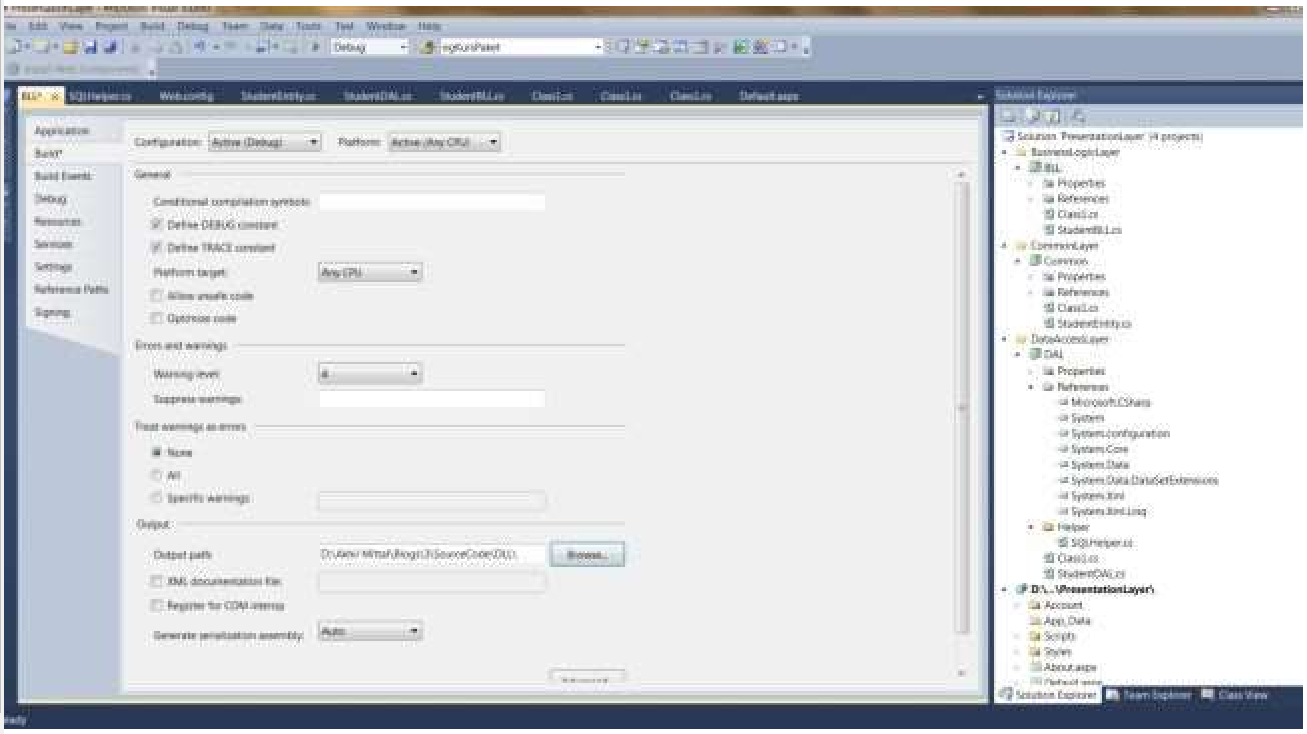
Step 6: Configure the solution and add DLLs to dependent layers

Since we need to separate the business logic, presentation and data access, and we want the Presentation Layer to neither directly interact with the database nor the business logic, we add a reference of the Business Logic Layer to our Presentation Layer and Data Access Layer to the Business Logic Layer and Common Layer to all the three layers. To do that, add a common DLL folder to the physical location of the solution and give the build path of all the three class projects to that DLL folder, doing this we can directly get access to the DLLs of all the layers to the layers needed for that DLL. We'll get the DLLs in that folder once we complete our project.



Computer ▸ Local Disk (D:) ▸ Akhil Mittal ▸ Blogs ▸ 3 ▸ SourceCode ▸

Name	Date modified	Type	Size
BLL	24-12-2011 15:34	File folder	
Common	24-12-2011 15:36	File folder	
DAL	24-12-2011 15:45	File folder	
DLL	24-12-2011 15:56	File folder	
PresentationLayer	24-12-2011 14:20	File folder	



Configuration: Active (Debug) Platform: Active (Any CPU)

General
Conditional compilation symbols:
☑ Define DEBUG constant
☑ Define TRACE constant
Platform target: Any CPU
☐ Allow unsafe code
☐ Optimize code

Errors and warnings
Warning level: 4
Suppress warnings:

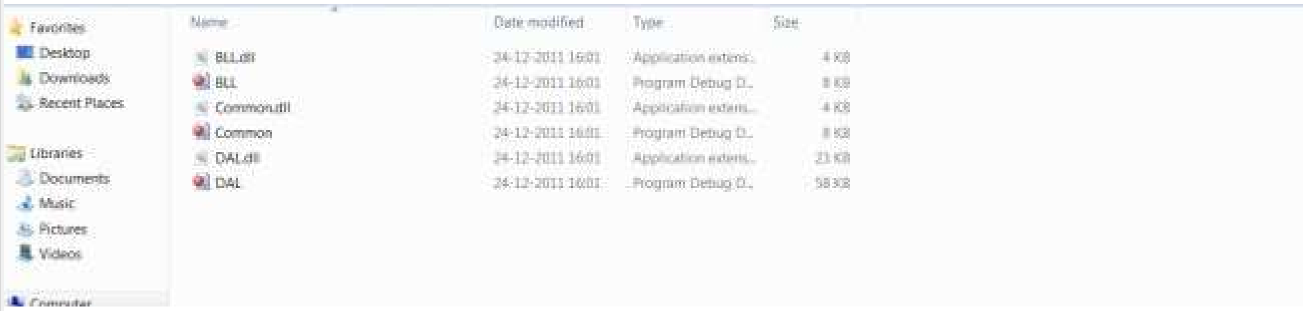
Treat warnings as errors
● None
○ All
○ Specific warnings:

Output
Output path: D:\Akhil Mittal\Blogs\3\SourceCode\DLL\
☐ XML documentation file:
☐ Register for COM interop
Generate serialization assembly: Auto

Solution 'PresentationLayer' (4 projects)
- BusinessLogicLayer
  - BLL
    - Properties
    - References
    - Class1.cs
    - StudentBLL.cs
- CommonLayer
  - Common
    - Properties
    - References
    - Class1.cs
    - StudentEntity.cs
- DataAccessLayer
  - DAL
    - Properties
    - References
      - Microsoft.CSharp
      - System
      - System.configuration
      - System.Core
      - System.Data
      - System.Data.DataSetExtensions
      - System.Xml
      - System.Xml.Linq
    - Helper
      - SQLHelper.cs
    - Class1.cs
    - StudentDAL.cs
- D:\...\PresentationLayer\
  - Account
  - App_Data
  - Scripts
  - Styles
  - About.aspx

Figure 8

Do this for the DAL and the Common Layer as shown in Figure 8. After compiling all the projects we get the DLLs in the DLL folder created.



Name	Date modified	Type	Size
BLL.dll	24-12-2011 16:01	Application extens...	4 KB
BLL	24-12-2011 16:01	Program Debug D...	8 KB
Common.dll	24-12-2011 16:01	Application extens...	4 KB
Common	24-12-2011 16:01	Program Debug D...	8 KB
DAL.dll	24-12-2011 16:01	Application extens...	23 KB
DAL	24-12-2011 16:01	Program Debug D...	58 KB

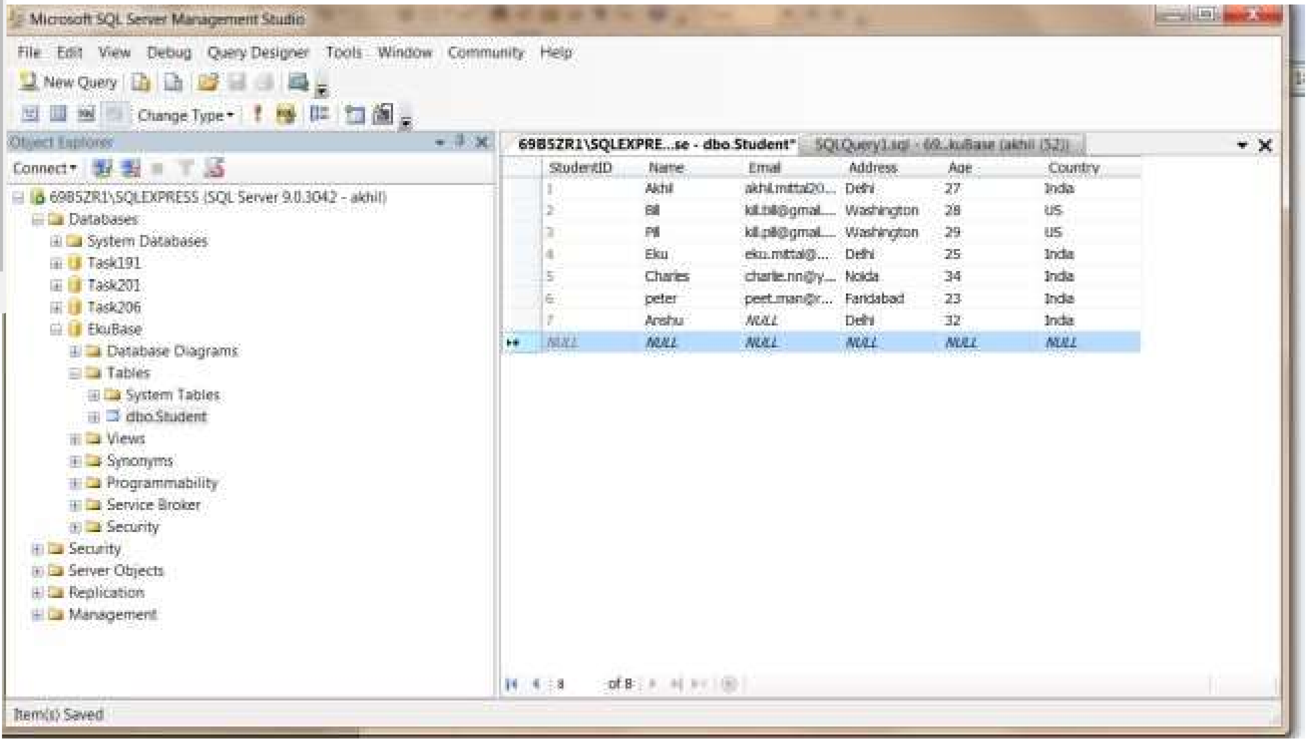
Now add a reference of Common.dll and BLL.dll to the Presentation Layer, Common.dll and DAL.dll to the BLL Layer and

Common.dll to the DAL layer and compile your solution.

Now the code of the BLL is accessible to the Presentation Layer and the DAL is accessible to the BLL and Common is accessible to all three layers.

Step 7: Write methods to get the flow.

Now we need to write some code to our layers to get the feel of the flow among all the layers. Let's create a scenario. Suppose we need to get the details of all the students whose student id is less than 5. For that add some sample data to your Student table, about 7 rows would work (Figure 11) and add a GridView to Default.aspx in the Presentation Layer to show the data.



69B5ZR1\SQLEXPRE...se - dbo.Student*

StudentID	Name	Email	Address	Age	Country
1	Akhil	akhil.mittal20...	Delhi	27	India
2	Bill	kill.bill@gmail....	Washington	28	US
3	Pill	kill.pill@gmail....	Washington	29	US
4	Eku	eku.mittal@....	Delhi	25	India
5	Charles	charlie.nn@y....	Noida	34	India
6	peter	peet.man@r....	Faridabad	23	India
7	Anshu	NULL	Delhi	32	India
NULL	NULL	NULL	NULL	NULL	NULL

Default.aspx:

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.master" AutoEventWireup="true"
    CodeFile="Default.aspx.cs" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadContent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
    <h2>
        Welcome to ASP.NET!
    </h2>
    <p>
        To learn more about ASP.NET visit <a href="http://www.asp.net" title="ASP.NET
Website">www.asp.net</a>.
    </p>
    <p>
        You can also find <a href="http://go.microsoft.com/fwlink/?LinkID=152368&clcid=0x409"
            title="MSDN ASP.NET Docs">documentation on ASP.NET at MSDN</a>.
    </p>
     <div>
    <asp:GridView runat ="server" ID="grdStudent"></asp:GridView>
    </div>
    <div>
               <asp:Label runat="server" ID="lblError" Font-Bold=true ForeColor=red ></asp:Label>
              </div>

</asp:Content>
```

Now decorate your StudentEntity Class in the Common Layer with the following code. To make properties for each column we access from the Student table:

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

namespace Common
```

```
{
    int studentID;

    public int StudentID
    {
        get { return studentID; }
        set { studentID = value; }
    }
    string name;

    public string Name
    {
        get { return name; }
        set { name = value; }
    }
    string email;

    public string Email
    {
        get { return email; }
        set { email = value; }
    }
    string address;

    public string Address
    {
        get { return address; }
        set { address = value; }
    }
    int age;

    public int Age
    {
        get { return age; }
        set { age = value; }
    }
    string country;
```

```
        {
            get { return country; }
            set { country = value; }
        }
    }
}
```

And StudentDAL with the following code calls the data from the database. We always write data interaction code in this layer only, making it a protocol.

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

namespace DAL
{
    public class StudentDAL
    {
        public DataSet FetchSelectedStudents()
        {
            string sqlCommand = "select * from Student where Studentid<5";
            DataSet dataSet = SqlHelper.ExecuteDataset(SqlHelper.CONN_STRING, CommandType.Text,
sqlCommand);
            return dataSet;
        }

    }
}
```

Here we simply make a call to the database to get students having an id less than 5.

We use the following code for the BLL class, where we perform a validation check for the id, whether it is less than or greater than 5, and the correspondingly throws an error when it is greater than 5, that we show in our default.aspx page by setting the message to the error label text. The BLL makes a call to the DAL to fetch the students, and es the data to the Presentation Layer, where the data is shown in the GridView.

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DAL;
using System.Data;
using Common;

namespace BLL
{
    public class StudentBLL
    {
        public DataTable GetStudentBelow5(StudentEntity student)
        {
            StudentDAL studentDAL = new StudentDAL();
            if (ValidateID(student.StudentID))
            {
                return studentDAL.FetchSelectedStudents().Tables[0];
            }
            return null;
         }

        private bool ValidateID(int studentID)
        {
            if (studentID > 5)
            {
                throw new ApplicationException("Id Should be less than 5");
            }
            return true;
        }
    }
}
```

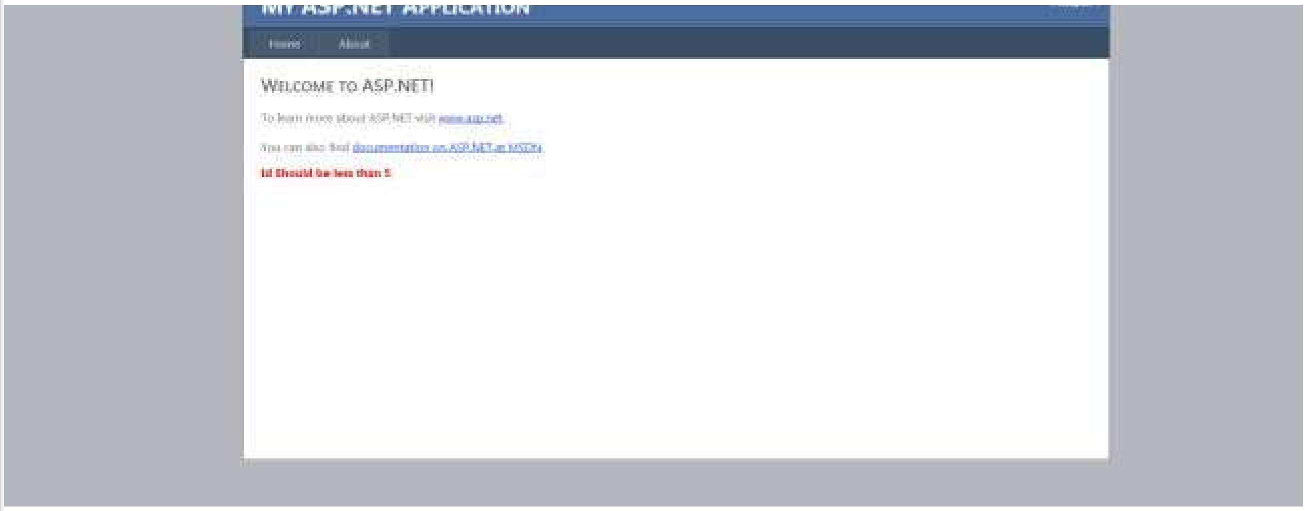
And in the Presentation Layer we write code to either bind our GridView or show the error message when an error is returned from the BLL.

Default.aspx.cs:

```
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using BLL;
using Common;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        StudentBLL studentBLL = new StudentBLL();
        StudentEntity studentEntity=new StudentEntity();
        studentEntity.StudentID=6;
        DataTable dTable = new DataTable();
        try
        {
            dTable = studentBLL.GetStudentBelow5(studentEntity);
            grdStudent.DataSource = dTable;
            grdStudent.DataBind();
        }
        catch (ApplicationException applicationException)
        {
            lblError.Text = applicationException.Message;
        }
    }
}
```

In the code above we specify the student id as 6 in the Student Entity and it to the BLL, when we run the code we get the following page with our error label set with the error message:



WELCOME TO ASP.NET!

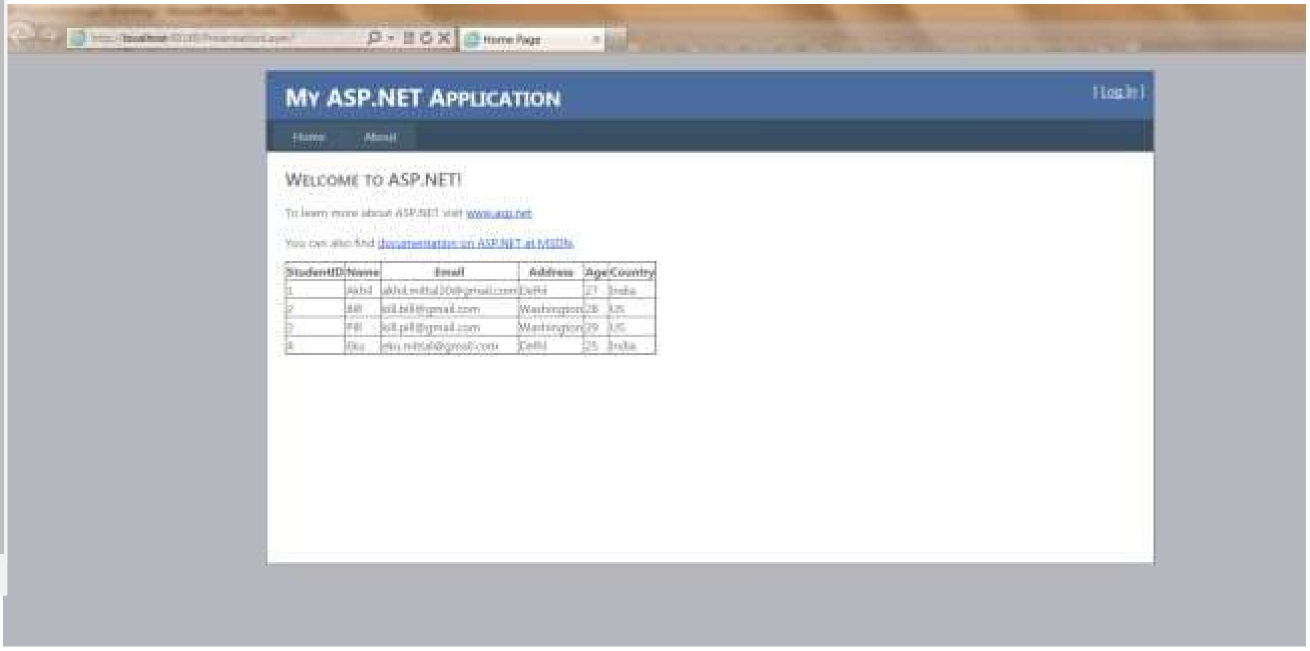
To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

Id Should be less than 5

It clearly states that id should be less than 5. Note that we do not get to the DAL before the data is validated.

Now change the student id to 5 and see the result; we get the following page:



WELCOME TO ASP.NET!

To learn more about ASP.NET visit www.asp.net.

You can also find documentation on ASP.NET at MSDN.

StudentID	Name	Email	Address	Age	Country
1	Akhil	akhil.mittal20@gmail.com	Delhi	27	India
2	Bill	bill.bill@gmail.com	Washington	28	US
3	Pill	bill.pill@gmail.com	Washington	29	US
4	Eku	eku.mittal@gmail.com	Delhi	25	India

Thus we get the clear result.

Here we have seen how we communicated through various layers performing various roles to fetch the data.

General advantages of layered applications

components and low coupling among various components. A system built on these principles will be more robust, easier to adjust, easier to maintain and understand and it allows various software components to be developed in parallel. The key point is that a system should be split into various smaller parts that are as cohesive and independent as possible. Each part has distinct responsibilities and interacts with other parts of the system to accomplish its tasks and responsibilities. This also ensures that the systems can cooperate across various platforms and communication protocols and makes it easier to reuse existing solutions to problems often encountered.

All in all, these advantages are desirable because they work in the direction of low coupling and high cohesion. The hard part is, however, to implement this in practice and to learn when and how it should be implemented. What objects should have responsibility for the various tasks and how they interact.

Summary:

In this article I discussed what layered applications are, various types of layered applications and how to create a multilayered application in .Net. We can handle exceptions more intelligently at the DAL and BLL, however that was not the scope of the article so that part is skipped and I'll surely discuss this in my future articles.

The article focused on development for beginners, that face challenges creating an architecture before starting development.

Read the following article of mine to learn how to create an enterprise-level application using MVC and Entity Framework:

http://www.c-sharpcorner.com/UploadFile/1492b1/creating-an-application-using-entity-framework-4-1-code-firs/

Read more:

- C# and ASP.NET Questions (All in one)
- MVC Interview Questions
- C# and ASP.NET Interview Questions and Answers
- Web Services and Windows Services Interview Questions

Other Series

My other series of articles:

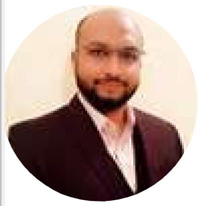
- Learning WebAPIs in .Net
- Learning OOP concepts with C#

Happy Coding :-) .

.net C# Layered Architecture Multilayered Architecture in .NET

RECOMMENDED FREE EBOOK

SIMILAR ARTICLES



Akhil Mittal TOP 100

Akhil Mittal is two times Microsoft MVP (Most Valuable Professional) firstly awarded in 2016 continued in 2017 in Visual Studio and Technologies category, C# Corner MVP since 2013, Code Project MVP since 2014, a blogger,... Read more

https://www.c-sharpcorner.com/authors/1492b1/

78 7.3m 8 2

View All Comments

8

Type your comment here and press Enter Key (Minimum 10 characters)