# **3-Tier Architecture in ASP.Net**

This topic explains how to create and implement a 3-tier architecture for our project in ASP.Net.

This article explains how to create and implement a 3-tier architecture for our project in ASP.Net.



## **What is a Layer?**

A layer is a reusable portion of code that performs a specific function.

In the .NET environment, a layer is usually set up as a project that represents this specific function. This specific layer is in charge of working with other layers to perform some specific goal.

Let’s briefly look at the latter situation first.

**Data Layer**

A DAL contains methods that helps the Business Layer to connect the data and perform required actions, whether to return data or to manipulate data (insert, update, delete and so on).

## **Business Layer**

A BAL contains business logic, validations or calculations related to the data.

Though a web site could talk to the data access layer directly, it usually goes through another layer called the Business Layer. The Business Layer is vital in that it validates the input conditions before calling a method from the data layer. This ensures the data input is correct before proceeding, and can often ensure that the outputs are correct as well. This validation of input is called business rules, meaning the rules that the Business Layer uses to make “judgments” about the data.

## **Presentation Layer**

The Presentation Layer contains pages like .aspx or Windows Forms forms where data is presented to the user or input is taken from the user. The ASP.NET web site or Windows Forms application (the UI for the project) is called the Presentation Layer. The Presentation Layer is the most important layer simply because it’s the one that everyone sees and uses. Even with a well structured business and data layer, if the Presentation Layer is designed poorly, this gives the users a poor view of the system

## **Advantages of a 3-Tier Architecture**

The main characteristic of a Host Architecture is that the application and databases reside on the same host computer and the user interacts with the host using an unfriendly dumb terminal. This architecture does not support distributed computing (the host applications are unable to connect to a database of a strategically allied partner). Some managers find that developing a host application takes too long and it is expensive. These disadvantages consequently led to a Client-Server architecture.

A Client-Server architecture is a 2-Tier architecture because the client does not distinguish between Presentation Layer and Business Layer. The increasing demands on GUI controls caused difficulty in managing the mixture of source code from a GUI and the Business Logic (Spaghetti Code). Further, the Client Server Architecture does not support enough the Change Management. Let us suppose that the government increases the Entertainment tax rate from 4% to 8 %, then in the Client-Server case, we need to send an update to each client and they must update synchronously on a specific time otherwise we may store invalid or incorrect information. The Client-Server Architecture is also a burden to network traffic and resources. Let us assume that about five hundred clients are working on a data server. Then we will have five hundred ODBC connections and several ruffian record sets, that must be transported from the server to the clients (because the Business Layer remains in the client side). The fact that Client-Server does not have any caching facilities like in ASP.NET causes additional traffic in the network. Normally, a server has better hardware than the client, therefore it is able to compute algorithms faster than a client, so this fact is also an additional argument in favor of the 3-Tier Architecture. This categorization of the application makes the function more reusable easily and it becomes too easy to find the functions that have been written previously. If a programmer wants to make further updates in the application then he can easily understand the previous written code and can update it easily.