2
1-Tier 1-Layer Architecture in ASP.NET

It's time to get our hands dirty with ASP.NET coding! In this chapter, we will understand through the use of examples:

- How every web application is N-tiered by default
- How applications based on classic inline ASP are tightly coupled
- What 1-tier 1-layer architecture is
- Code-behind classes in ASP.NET as another layer in the UI tier
- How Data Source Controls fit into the application architecture of ASP.NET web applications

This chapter is not a guide to how data source controls work, but is rather focused on the architectural aspects of using them and learning about the advantages and disadvantages of data source controls, instead of going into the deep technical details of using them.

Default N-Tier Nature of Web Applications

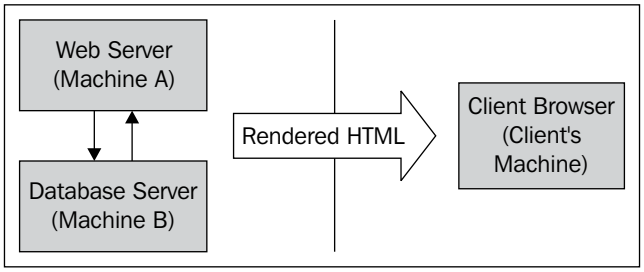
When working with web applications, a very important concept to grasp is that by its very own nature each web application is distributed and is inherently 2-tier by default (or 3-tier if we include the database as a separate tier). Therefore, it is not possible to have a single-tier (or 1-tier) architecture at all, when dealing with web applications. And as we saw in the last chapter, if we include a database and client browser in our system, then we already have a basic 3-tier application structure.

Let's understand this concept in detail with a sample configuration for a simple ASP.NET web application:

- **Web Server**: A machine running a web server such as IIS, handling all HTTP requests and passing them onto the ASP.NET runtime process. The deployed project files (`ASPX, ASCX, DLLs` etc) are published on this server.
- **Database Server**: This will be the physical database such as SQL Server, Oracle and so on. It can be on the same machine as the web server or on a separate machine.
- **Client Browser**: This will be the browser that the client is running to view the web application. The browser runs and uses client machine resources.

The example shows a deployment scenario, where we have the web application deployed on machine A, which is running IIS.

This is how the configuration will look:



Based on the above diagram, we have a distributed 3-tier architecture with the following tiers:

- **Presentation Tier**: This is the client browser displaying rendered HTML from the web server.
- **Application Tier**: This is machine A, which has the web server running along with the application's UI, business logic, and data access code, all deployed together on the same machine.
- **Data Tier**: This is the database running on machine B. We can also call this a Data Layer.

An important point to note is that in ASP.NET web applications, the presentation tier (browser) is different from the GUI (which is actually the ASPX/ASCX application frontend).