A 4-Tier Approach

In the last chapter, we saw how layers nicely separated the application code into logical partitions. Now Breaking them further into separate physical assemblies will introduce some slight complexity, and to handle that we need to follow some design patterns.

At the simplest level, this is how we can segregate the application to have the following four tiers:

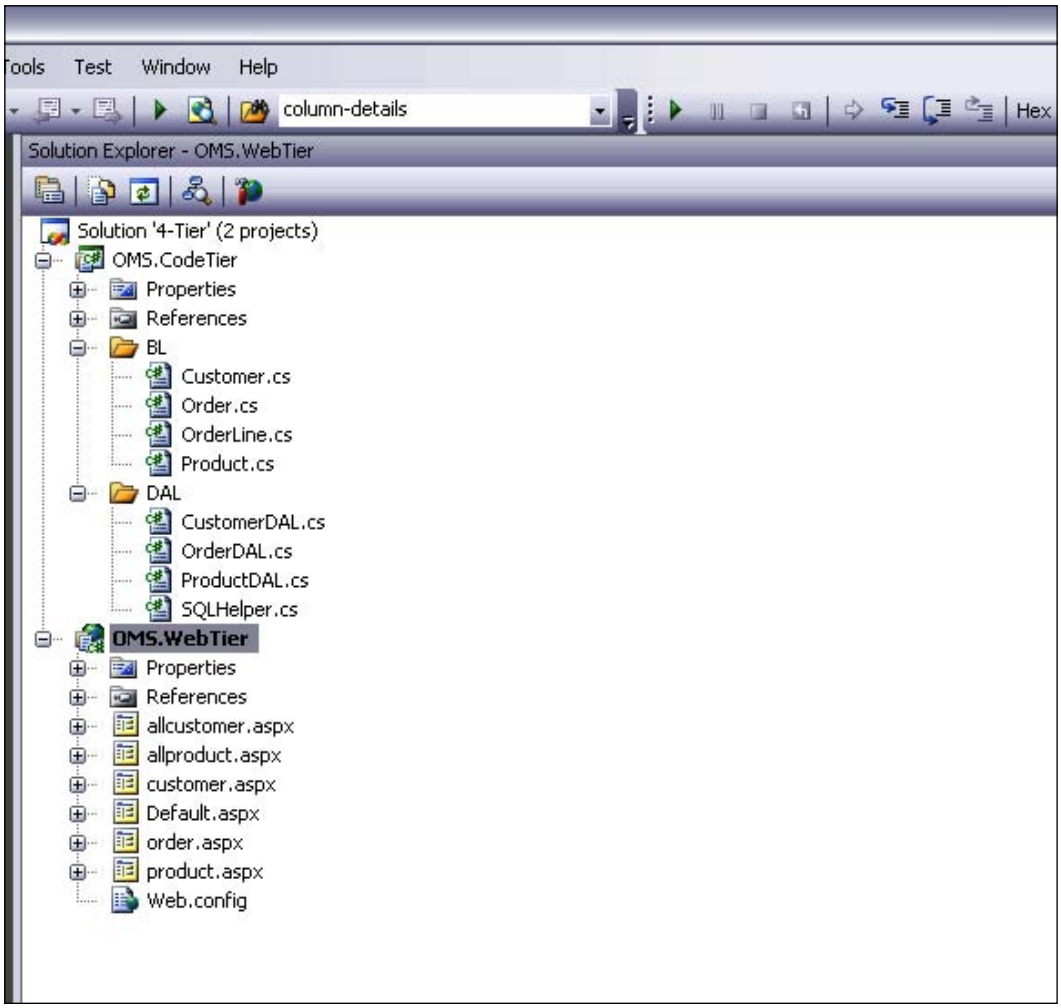
- Presentation tier: client-side browser
- UI tier: Web project having ASPX/ASCX, code-behind files
- Data access and business logic tier: in a separate class library project
- Data tier: the physical database

Note that in our first step to moving from a standard 3-tier web based application to 4-tier architecture, we are separating the DAL and BL code from the main web application into its own assembly. In a later section of this chapter, we will see how we can separate the DAL and BL into different assemblies.

To separate the BL and DAL code from logical layers (in the last chapter) to physical assemblies, we first need to follow these simple steps:

1. Create a new ASP.NET Web project in Visual Studio 2008 and name it **OMS.WebTier**.
2. Create a new class library project named **OMS.CodeTier** that will hold the business logic as well as data access files.
3. Move all of the files from the BL and DAL folders we used in the old GUI project (from Chapter 3) into this new class library project. We just need to modify the namespaces used (we will see the code to do this soon).
4. Add a reference to this class library project in the main web project.

Here is the solution structure image (in Visual Studio):



In the above solution structure, we have a GUI tier (**OMS.WebTier**), which will have all of the web pages, user controls and code-behind classes. This GUI tier will have a reference to a class library project named **OMS.CodeTier**, which will have the complete business logic and data access code separated into different folders under different namespaces. So from the 1-tier approach in the previous chapters, we now have a 2-tier solution, and along with having presentation and the database as separate tiers, we have implemented a classic 4-tier application architecture for our Order Management System. Now let us study how the coding looks for this 4-tier sample. Most of the code is similar to the code in Chapter 3.