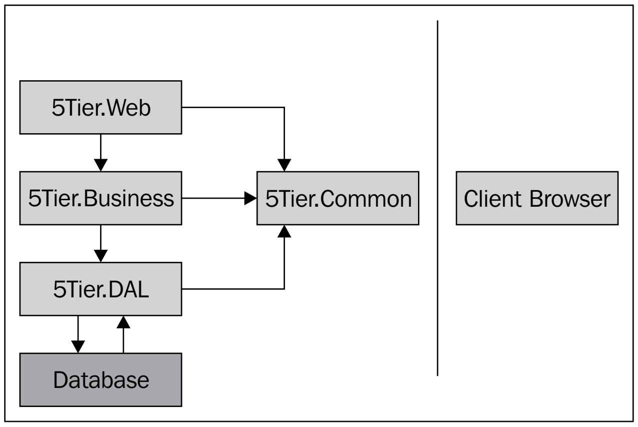
- `5Tier.Common`: This class library project will have code that is common to all layers (such as enumerated types) and will also hold Data Transfer Objects. This project will be referenced by all layers, since it is common to all of them.
- `5Tier.Business`: This class library project will handle all business objects and will call the DAL project and pass data to and from it using DTOs. So this project will reference the `5Tier.Common` project as well as the `NTier.DAL` project.
- `5Tier.Web`: This web project will have the UI layer and will also use DTOs to display and pass data. It will reference the `5Tier.Common` and `5Tier.Business` projects only.

The following diagram shows how these tiers would interact with each other:



Some of you might be thinking why I did not call this system a 6-tier system. Since we have the `5Tier.Common` project compiling to a separate physical assembly, we can include it as another tier and refer the system as one belonging to 6-tier architecture instead of a 5-tier one. But as I said earlier, there are no strict rules and it is up to us to define and configure the application.

I do not see the Common project as a separate tier as I treat the DTOs as *transient structures*, which pass through the different tiers and can be accessed by any layer. So I do not treat them as a separate tier. But if you do want to name it separately, there is nothing wrong with that convention either.

Let us start with the code in the business layer. Here, I will highlight the important portions of the code to understand the architectural aspects of this 5-tier configuration.

Take the example of what the `Customer` class will look like in this business tier. This class will be similar to the 4-tier business classes we saw earlier, and will have all of the same attributes as well as methods such as `Add()`, `Update()`, `Delete()`, `Find()`. Now, instead of putting all fields as private variables and then making properties inside the `Customer` class, we first create a DTO named `CustomerDTO` in the Common project encapsulating all the customer-related attributes.

Create two new class library projects as follows:

- `5Tier.Business`
- `5Tier.Common`

In the 5Tier.Common project, create a new folder named DTO. Add a new class file named `CustomerDTO`, which has the following code:

```
using System;
using System.Collections.Generic;

namespace 5Tier.Common
{
   /// <summary>
   ///DTO for the 'Customer' object.
   /// </summary>
   [Serializable]
   public class CustomerDTO
   {
       #region Constructors
        ///<summary>
      /// Default constructor
          ///</summary>
       public CustomerDTO()
      {
        this.loadStatus = LoadStatus.Initialized;
      }

      ///<summary>
      /// Copy constructor
      ///</summary>
       public CustomerDTO(CustomerDTO sourceDTO)
      {
          loadStatus = dto.LoadStatus;
          ID = sourceDTO.ID;
          Name = sourceDTO.Name;
          Address = sourceDTO.Address;
          PhoneNo = sourceDTO.PhoneNo;
```