The CustomerDAL class is pretty simple: we are fetching the data from the database using data readers, and performing all data related operations using the Customer business object. This Customer class is defined in the Customer.cs class we created earlier. This BL class is calling DAL methods, so it needs a reference to the DAL namespace (using DomainModel.DAL). Similarly, the DAL class we created earlier used Customer business objects. That's why it also needed the BL namespace.

We are using generics to create a collection of Customer objects. The BL communicates with DAL to get the data and perform updates. Now, we will see how the UI (which is under a different namespace) talks to BL without even knowing about DAL.

Layer 3: The UI Layer

Here is the code in the AllCustomers.aspx.cs page that shows a list of all of the customers from the DB (there is a data list on the web form, which will show a list of the customers):

```
using DomainModel.BL;
namespace DomainModel.UI
    {
    //page load
    private void FillAllCustomers()
        {
            Customer c = new Customer();
            c.GetAll();
            List<Customer> cuList = c.CustomerCollection;
            dtlstAllCustomer.DataSource = cuList;
            dtlstAllCustomer.DataBind();
        }
    }
```

So in the UI class, we neither have any data access code (as we had in the previous chapter), nor are we calling data access class methods from this layer (as was the case with the 1-tier 2-layer style we saw earlier in this chapter). We have a reference to the BL layer (using DomainModel.BL), and we are using the Customer business object to return a generic list of customer objects, which we are binding to the data list control (showing a list of all the customers). So the GUI layer does not know anything about the DAL layer, and is completely independent of it.

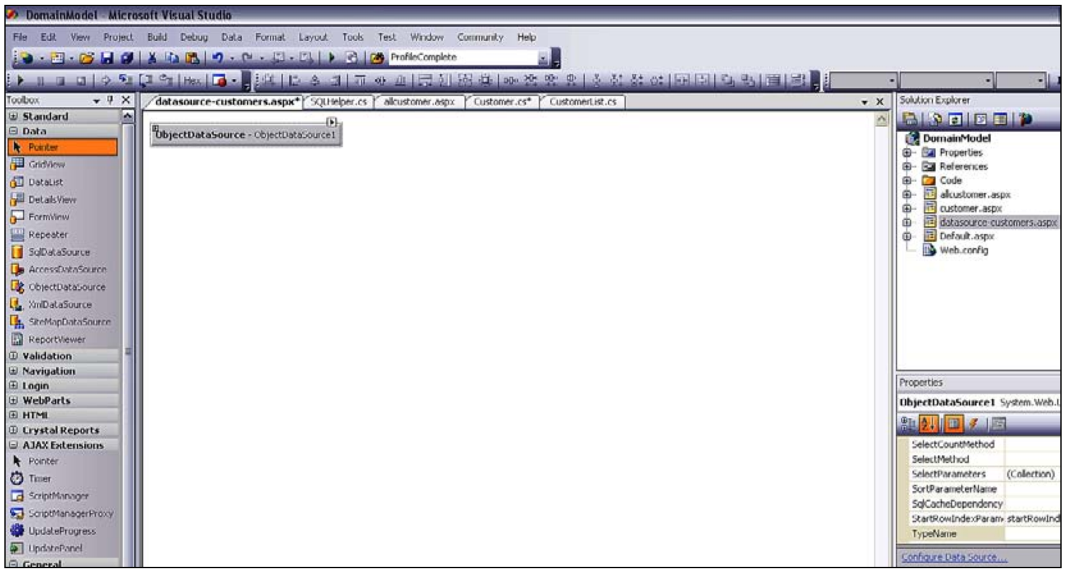
The idea here is to understand how a 3-Layer architecture can provide more flexibility and loose-coupling to your project. In the next section, we will learn how we can use object data source controls to implement a 3-layer architecture without writing much code ourselves.

Object Data Source Controls

We looked at the data source controls in the last chapter and saw how they replaced the data access code, but tightly coupled the GUI to the data methods. To overcome this problem, Microsoft introduced object data source controls, so that we can bind directly to business objects, making it possible to use them in a 3-tier architecture.

Let's see how using object data source controls will shape our application:

1. Create a new web project using VS.
2. Add a new form named `datasource-customer.aspx`.
3. Add an object data source control, as shown here (drag and drop the object data source control from the **Data** tab under **ToolBox** in VS):



4. Now, we need to configure this object data source control. We first need to set the **Business object** , where we select our customer class: