

TierDeveloper Migration Guide

(From TierDeveloper 5.2.1 or earlier to TierDeveloper 5.2.2 or later)

In the latest release of TierDeveloper, significant architectural enhancements have been made. The changes have been brought about, keeping in view the demands of our customers and to help them adapt to the evolving Enterprise Application development. Following are the architectural improvements made in TierDeveloper.

Separation of Domain Objects from Persistence Layer

A clear separation has been introduced between the Domain Objects and the Persistence layer. This has been accomplished by adding a Business layer (Domain objects, Interfaces and Collections assembly) and Data Layer (Factory, Hooks, Interface assemblies) and a new Integration layer that works as a Service Provider to your application. The availability of the Interfaces has also hidden the Factory implementation from the user.

This separation allows you authority over the access to the Factory assembly. You can easily provide the Object assembly while keeping the Factory hidden.

Integration Layer

As mentioned above, the Integration layer works as a Service Provider to the client. It contains a Service Provider class, which is used to instantiate the factory interface objects.

This provides better code readability and helps adapt any future changes because the objects are initialized at central location.

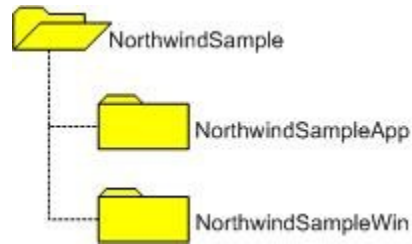
Separate Independent .NET assembly for Custom Hooks

Another enhancement made is the generation of an independent .NET assembly for Custom Hooks. A separate project solution is generated for the Hooks, where you can incorporate your own complex business logic and automatically reference it to the data access layer.

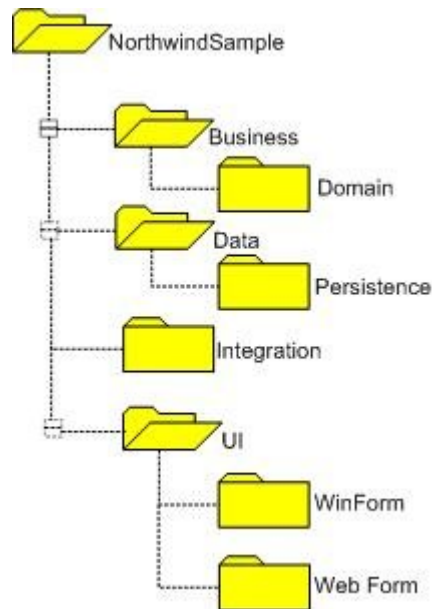
This way you can keep your custom logic separate and any changes or further addition is made without any hassle.

Changes in the Folder Structure

The separation of Namespaces has brought about a clear change in the folder structure of the project generated by TierDeveloper. Suppose the name of your project is NorthwindSample. Before this release, the folder structure generated for a Winform application would have been as shown in the figure below.



The latest release of TierDeveloper generates a rather different folder structure. Refer to the figure below to view the difference



You can clearly visualize the Namespace separation from the new folder structure as well.

Migration Process:

The components generated through latest release of TierDeveloper can easily be used in applications generated with previous version of TierDeveloper. You can readily migrate your application to the latest version of TierDeveloper. Assume that the name of your project is NorthwindSample. Along with the .tier file created by the name NorthwindSample, a folder containing the components is also created when components are generated. Follow the steps below to use your application with the components generated using the latest version of TierDeveloper. Note that instead of NorthwindSample, you have to make changes to whatever your project name is.

- Copy all contents of NorthwindSampleWin to NorthwindSample \UI\WinForm
- Copy following files from NorthwindSample \Business\Domain\Bin

EnterpriseNonServicedLib.dll
NorthwindSample.Business.Domain.dll

- Then paste them into NorthwindSample\UI\WinForm\Bin
- Copy following files from NorthwindSample\Data\Persistence

NorthwindSample.Data.Persistence.dll
NorthwindSample.Data.Persistence.dll.config

- Then paste them into NorthwindSample\UI\WinForm\Bin
- Copy following files from NorthwindSample\Integration

NorthwindSample.Integration.dll

- Then paste these into NorthwindSample\UI\WinForm\Bin
- Open WinForm solution file in VS.Net
- Add following References into the project (Browse to NorthwindSample\UI\WinForm\Bin)

EnterpriseNonServicedLibEval.dll
EnterpriseNonServicedLibEval.dll.config
NorthwindSample.Business.Domain.dll
NorthwindSample.Data.Persistence.dll
NorthwindSample.Data.Persistence.dll.config
NorthwindSample.Integration.dll

- Add following import statements in all forms

```
Imports NorthwindSample.Business.Domain  
Imports NorthwindSample.Business.Interfaces  
Imports NorthwindSample.Integration
```

- Change the way Factory objects are declared and initialized. Earlier, factory objects were instantiated as

```
Dim objFactory As CustomersFactory = New CustomersFactory()  
Dim objFactory As EmployeesFactory = New EmployeesFactory()  
Dim objFactory As OrdersFactory = New OrdersFactory()
```

From TierDeveloper 5.2.2 and on, the above statements will look like

```
Dim objFactory As ICustomersFactory =  
ServiceProvider.getCustomersFactory()  
Dim objFactory As IEmployeesFactory =  
ServiceProvider.getEmployeesFactory()  
Dim objFactory As IOrdersFactory =  
ServiceProvider.getOrdersFactory()
```

After these changes, the application should run without any compilation error. Similar steps should be followed in order to use existing web applications with new components generated using TierDeveloper 5.2.2.

Utilizing Integration Layer Features:

In the previous versions of TierDeveloper, Factory Class was directly accessed to utilize its methods. The user would directly instantiate the Factory methods, so there was no control over restricting the access to Factory Class. Refer to the code fragment given below.

```
Employees objInfo = new Employees();
EmployeesFactory objFactory = new
EmployeesFactory();
```

You can see that the Factory for Employees is being directly instantiated. However, in the latest release of TierDeveloper, the Integration layer has been provided which instantiates the Factory interfaces objects. View the code snippet below for understanding how the interfaces relate to the Factory.

```
public static IEmployeesFactory getEmployeesFactory()
{
    return new EmployeesFactory();
}
```

Now instead of directly accessing the Factory, the factory is accessed through the interface provided by the Service Provider Class in the Integration Layer.

```
Employees objInfo = new Employees();
IEmployeesFactory objFactory =ServiceProvider.getEmployeesFactory();
```

Now you can keep the Factory Class hidden and keep a control over the allowance of letting other access it.

The latest release of TierDeveloper has been enhanced keeping in view the demands and requirements of the customers. The significant architectural changes have been made to allow you to create and modify code with maximum convenience.

If you still have any queries or problems, please contact us - support@alachisoft.com