

Data Tier.Net Quick Start

Setting up Data Tier.Net

Before you can create a Data Tier.Net project you must install the Data Tier.Net SQL Server database.

Attach Database - SQL Server 2012 or higher

For SQL Server 2012 or higher, you can attach the DataTier.Net.Database database. Copy the database from the source code directory to the data directory of your choice; I typically use C:\Data, or C:\Database.

This database needs to be outside of the repo is why you need to move it.

In SQL Server Management Studio, select your server and attach the database.

Manual Install

Create a new database named 'DataTier.Net.Database' in SQL Server Management Studio.

Ensure you have the new DataTier.Net.Database selected, and execute the following two scripts located here

[DataJuggler\DataTier.Net\DataTier.Net\Database\SQL Scripts](#)

1. DataTier.Net.Tables.sql
2. DataTier.Net.StoredProcedures.sql

Windows Authentication

If you are using Integrated Security, ensure your Windows Login has permission to execute stored procedures and modify data.

SQL Server Authentication

Create a new SQL Server login named 'DataTierNetUser'. Include the user in the following two roles: DataReader, DataWriter.

Solving Security Problems

It is outside of the scope of this document to solve SQL permission problems, but one useful stored procedure included in the DataTier.Net.Database could help. The script is located in the DataTier.Net\Database\SQLScripts folder.

In SQL Server Management Studio, create a new query window and ensure the correct database is selected before running the following query

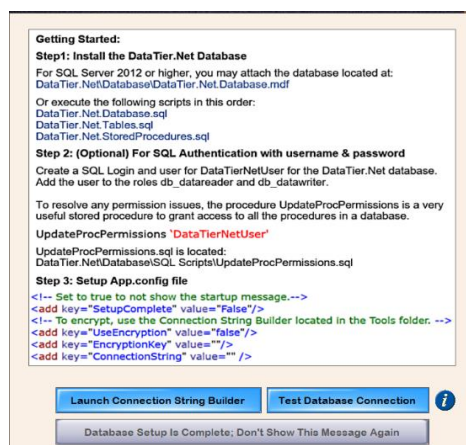
UpdateProcPermissions 'DataTierNetUser'

I regret not saving the author of this procedure to give them credit, but this has saved me on numerous occasions.

Create Your Connection String

(optional – you may paste in your own connection string if you prefer of course).

Launch DataTier.Net and click the 'Launch Connection String Builder' button.



Connection Builder location:
DataTier.Net\DataTier.Net\Tools\ConnectionStringBuilder

Connection String Builder (optional)



This program is self-explanatory.

I keep a shortcut to Connection String Builder on my desktop.
This is another useful tool included in the repo.

Use Encryption (Optional)

Both of the previous two versions of DataTier.Net, DataClassBuilder.Net and RAD Studio Code Generation Toolkit required the connection string to be encrypted. I guess I got a little older and less paranoid and stopped caring if my dog could read my connection string or not.

By default, connection strings are now stored in plain text in the app.config file.

If this is a security problem, you may encrypt your connection string. If you choose to encrypt your connection string, you must set 'Use Encryption' to true in your app.config file, and if wanted an encryption key, else your connection will fail.

```
<!-- To encrypt, use the Connection String Builder located in the Tools folder. -->  
<add key="UseEncryption" value="false"/>  
<add key="EncryptionKey" value="" />
```

The default encryption key is unpublished for security reasons, but if you step through the code you can of course figure it out.

I would rather you not publish it, but if I was too worried about this, I would not be going the open source route.

After you build your connection string, paste it into the app.config in DataTier.Net.

```
<!-- Set the ConnectionString -->  
<add key="ConnectionString" value="" />
```

And the final step is to not show the setup guide, set Setup Complete = true.

```
<!-- Set to true to not show the setup guide.-->  
<add key="SetupComplete" value="true" />
```

I started creating an app.config updater once to do this, but it was just easier to let a user do this, as we are all developers using this tool and it isn't difficult.

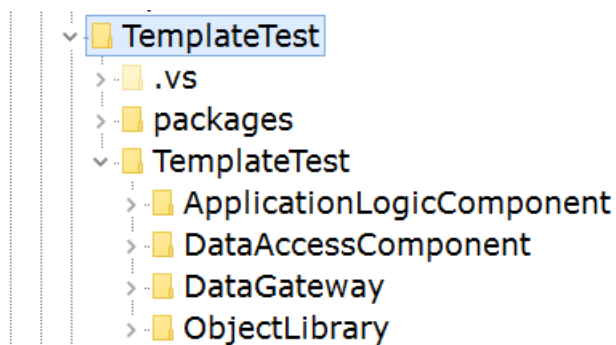
How to Create Your Own Project in DataTier.Net?

Create or Select a Microsoft SQL Server database.

The first step is to choose the database(s) that you are going to use for your project. Read the Data Tier.Net Users Guide for more information.

Install the DataTier.Net Project Templates

The project templates create a Class Library which includes all the required references and projects required to build a DataTier.Net solution. The image below shows the directory structure of a Data Tier.Net Project Template:



Note you can change the name of the project after you create it of course.

To install the Data Tier.Net Project Templates, copy the templates from the install directory of Data Tier.Net\Templates located in the Data Juggler Shared Repo at:

DataJuggler\DataTier.Net\DataTier.Net\Project Templates

The project templates were updated for DataTier.Net

The project templates are currently only available for version 4.61 of the .Net Framework. You can always change it after you create your project.

Paste the file DataTier.Net.ClassLibrary.zip into the Documents folder for Visual Studio 2017 or Visual Studio 2019.

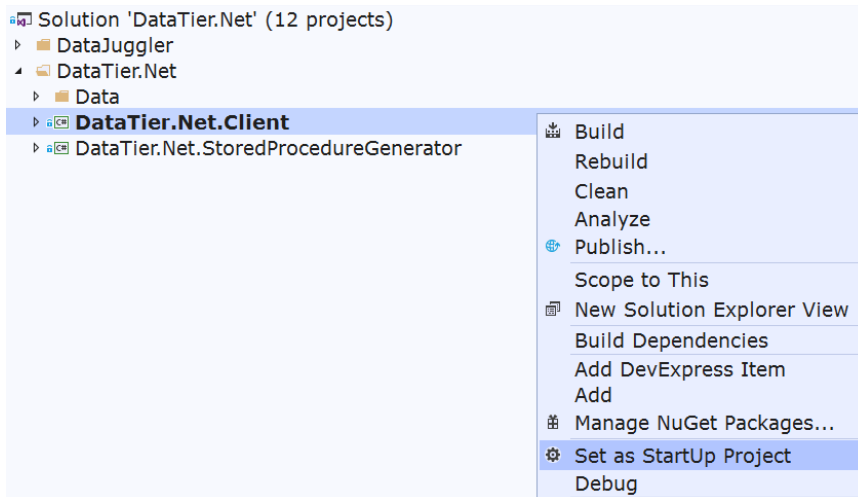
Typical location for Windows 10 and Visual Studio 2019 is shown below:

2017 C:\Users\[UserName]\Documents\Visual Studio
2017\Templates\ProjectTemplates\Visual C#

2019 C:\Users\[UserName]\Documents\Visual Studio
2019\Templates\ProjectTemplates\Visual C#

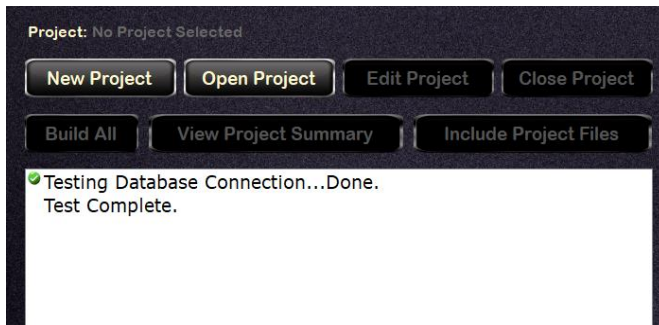
Create a new project in Data Tier.Net

Tip: You will need to set the DataTier.Net.Client project as the start-up project in Solution Explorer in Visual Studio.



Create Your First Project

Start Data Tier.Net and click the New Project Button:



Troubleshooting Connection Issues

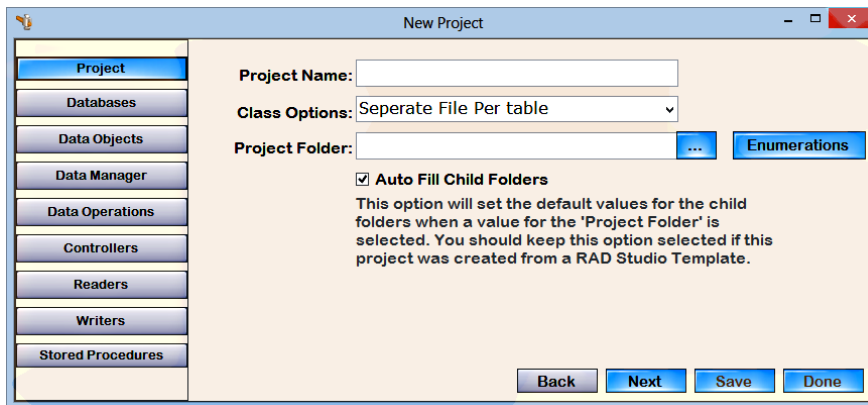
If you cannot connect to the Data Tier.Net database, the most likely cause is either the Data Tier.Net SQL Server database is not installed or the connection string is not set or incorrect in the DataTier.Net.Client.app.config file.

If you are using SQL Server authentication with a user name and password, you may need to install and execute the stored procedure 'UpdateProcPermissions', which is installed in the DataTier.Net database, and can be found in the SQL Scripts folder of the Database folder in DataTier.Net project:

DataTier.Net\DataTier.Net\Database\SQL Scripts\UpdateProcPermissions.sql

Click the New Project button. This will launch the Project Wizard Control.

Project Wizard



The screenshot shows the 'New Project' dialog box. On the left is a sidebar with buttons for 'Project', 'Databases', 'Data Objects', 'Data Manager', 'Data Operations', 'Controllers', 'Readers', 'Writers', and 'Stored Procedures'. The 'Project' button is highlighted. The main area contains the following fields and options:

- Project Name:** A text input field.
- Class Options:** A dropdown menu currently showing 'Seperate File Per table'.
- Project Folder:** A text input field with a browse button (...).
- Enumerations:** A button next to the Project Folder field.
- Auto Fill Child Folders:** A checked checkbox with a descriptive text: 'This option will set the default values for the child folders when a value for the 'Project Folder' is selected. You should keep this option selected if this project was created from a RAD Studio Template.'
- Navigation buttons:** 'Back', 'Next', 'Save', and 'Done' at the bottom right.

Give your project a name and leave default Class Option.

Note: The Single File per Table is the default value and the most tested. The single file per database option did work at one time and should still work, but I have not tested this in over a decade. I strongly recommend clicking Separate File per Table or use at your own risk. I just never got around to removing the option for Single File per Database.

Bug Reporting

If you discover any bugs (undocumented features as I call them) report them on the repo at github.com/DataJuggler. Note I do have a day job and will have to prioritize all bugs based upon severity and my time availability.

Note: bugs that also list the fix will be prioritized ahead of bugs that require debugging.

Required Folders

You are not required to structure your project in the same manner as the templates create, but using the templates is a huge time saver and the structure of the templates is the only tested and supported directory structure.

Auto Fill Child Folders

☒ Auto Fill Child Folders

This option will set the default values for child folders when the value for the 'Project Folder' is selected.

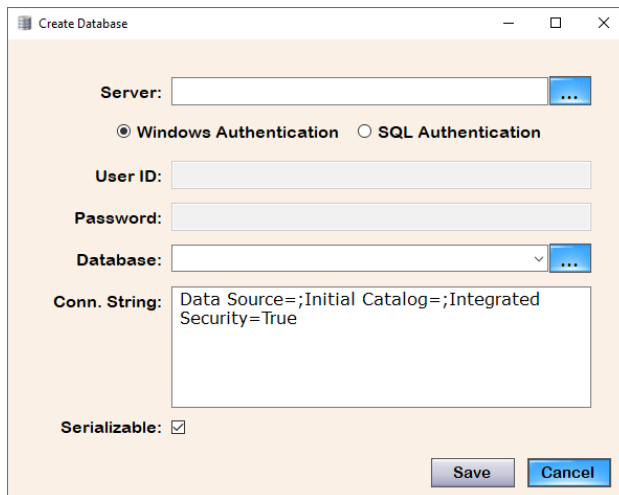
You should keep this option selected if this project was created from a DataTier.Net Template.

By default, this value is selected and will populate the other tabs of the project wizard for you. **Leave this option selected, else use at your own risk.**

Add a database to your project.

Click the database tab and click the Add button:

This brings up the Database Editor



Note: this connection is only used by Data Tier.Net to retrieve the database structure. The connection string for your own DataTier.Net projects is configured in the app.config or web.config files in your solutions.

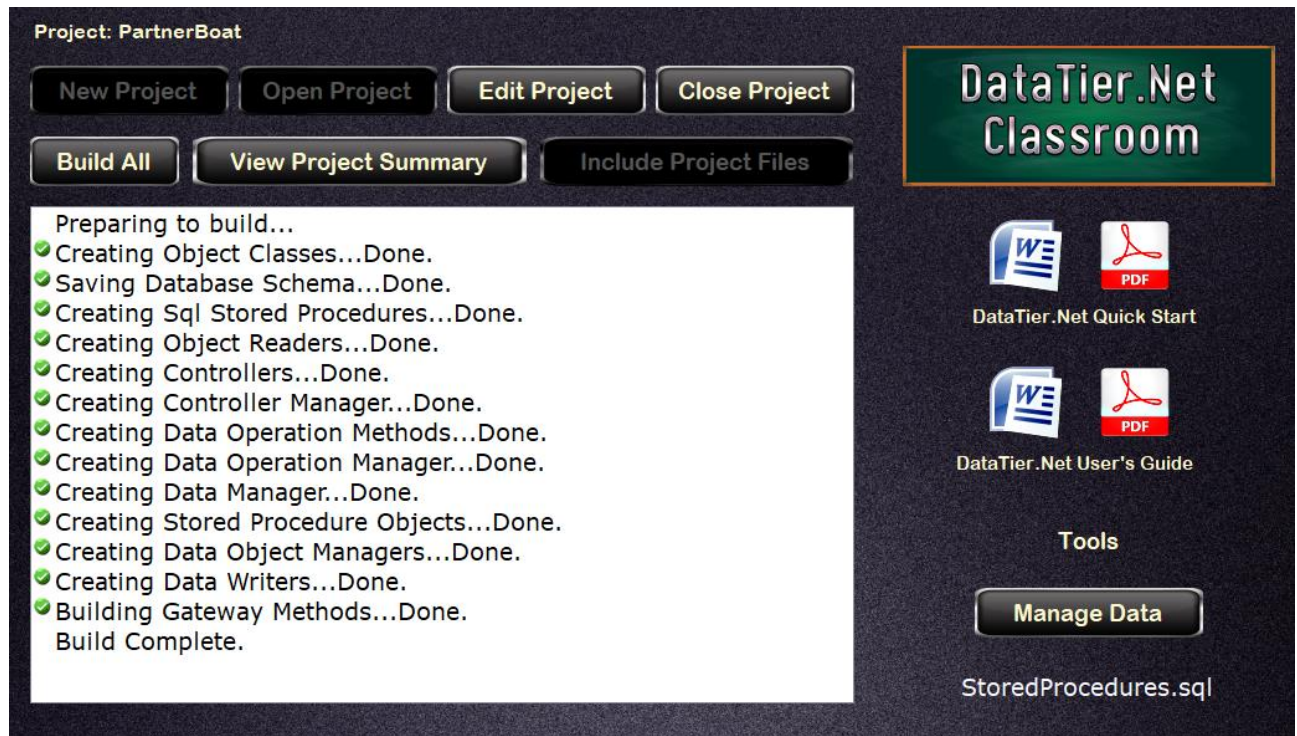
Tip: If you click the Server button to browse for Servers, you may need to add \SQLExpress to the server name detected. This is a known issue.

Provided the connection to the database can be established in the previous step, when you click the Browse Database Button, the list of available database will be populated for you.

Save your database and now you are ready to build your first project .

Build Your Project

Clicking the Build All Button will code generate all the required objects and stored procedures to assemble your data tier.



[Update: 4.5.2019: Screenshot updated for DataTier.Net.](#)

Include the generated files in your project

Files that are code generated must be included into your Visual Studio solution.

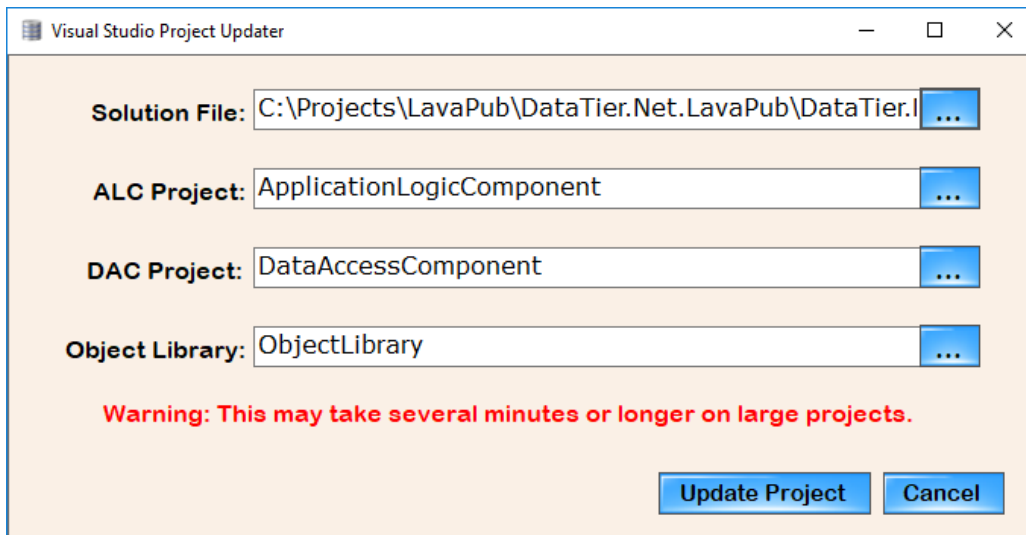
Update: 1.24.2010

For years I have been manually including the generated files in Visual Studio projects. This update allows you to Include project files after a build by clicking the 'Include Project Files' button which becomes enabled when a build is complete.

Using the Visual Studio Project Updater Control

Step 1: The first step is to select your Visual Studio solution file (.sln).


Data Tier.Net will attempt to determine your project names after your solution file is selected:



If you created your project from a Data Tier.Net project template the names will be correct. You can rename your projects after creating from a Data Tier.Net project template.

If the project names are not detected, click the browse button for each project and a browse dialog for each project file will launch.

After your projects are selected, click the Include Project Files button. This will include any files in your project that were generated during the last build.

 You are not required to click the Include 'Project Files' button after each build. It is only necessary to include new files the first time you build, or if new tables or views have been added since the last build.

5.1.2019: The Include Project Files Button Is now clicked on your behalf.

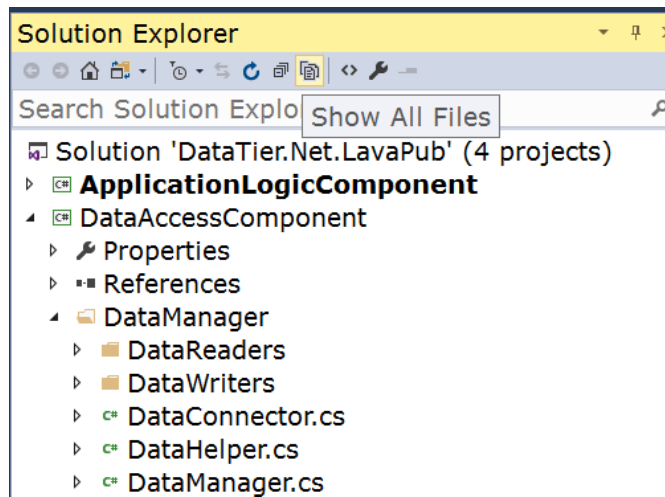
Often, I would forget to click the Include Project Files button, and I would have to manually include all the required project files, and it is quite a tedious job.

You should not have to manually include project files now, but if your power were to go out or if you shut down the Visual Studio project updater, I will leave this in.

Manually include files in your project

After building your project you must include your project files into your Visual Studio solution.

Step 1: Open your Visual Studio project, and in Solution Explorer, click the Show All Files button as shown here.



Toggle the Show All Files Button; this will refresh Solution Explorer to show any files that are on the file system but are not included in your project.

Select any file(s) you wish to include in your project and then right click and select the option "Include in Project".

There are a total of about 10 directories that will need to include files that have been code generated into.

The following is a list of folders to include after you build a Data Tier.Net project.

Project: Application Logic Component

Folders: Controllers
Data Operations

Project: Data Access Component

Folders: DataManager
DataReader
DataWriter *

StoredProcedureManager\Delete Procedures
StoredProcedureManager\Fetch Procedures *
StoredProcedureManager\Insert Procedures
StoredProcedureManager\Update Procedures

Project: Object Library

Folders: BusinessObjects *

* All objects in most tables will have a single file per table.

The following objects have two files per table included in the folder.

The Fetch Procedures will create a find (single instance returned) and a FetchAll method returns a List<T>, where T = the class object created for your table.

The Data Writers create a base class and a derived class for each table.

The Business objects use partial classes so for each table there will be two files created:

<Table Name>.business.cs, <Table Name>.data.cs

Place any customizations in the .business class, as the .data class will be overwritten each time you build with DataTier.Net.

Executing the StoredProcedures.sql

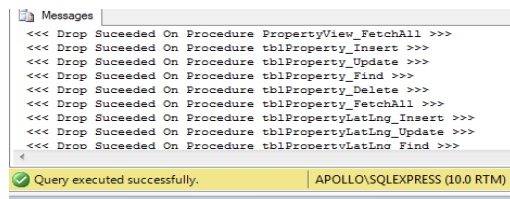
Update 4.2.2019: Click the link 'Stored Procedures.sql' in DataTier.Net

The file is kind of buried, so I added this link to make it exponentially easier to find. Provided you have SQL Server Management Studio installed, Data Tier.Net will launch SSMS and open the stored procedures.sql file that was just created.

Location of stored procedures.sql in the Data Access Component project:

[StoredProcedureManager\StoredProceduresSQL\storedprocedures.sql](#)

After executing the stored procedures, you should see a message like:



Update 12.19.2012: Gateway Generator

The gateway is now code generated when you build.

Load Method

For each table or view in your database, the gateway will create a Load method.

Update 4.5.2019: The above documentation does not state that now each 'Active' table will create a Load method. Tables can now be excluded, but for the remainder of this section, just keep this factor in mind.

Delete, Find and Save Methods

For tables that have an Identity Insert (auto-number) primary key, the Delete, Find and Save methods will be code generated.

Update 4.5.2019: The main engine of DataTier.Net, DataJuggler.Net.SQLDatabaseConnector, is now more robust than it was when this 'must be identity insert' rule was implemented. I just haven't had time to implement this.

Once I publish, volunteers are invited to help make improvements.

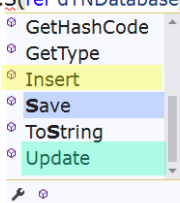
Save Method

The Save method analyzes the IsNew property of the object being saved, and calls Insert or Update accordingly.

Bypassing the Gateway

You may bypass the Gateway if you choose, and create an instance of the AppController as shown here:

```
// perform the save
saved = this.AppController.ControllerManager.DTNDatabaseController.S(ref dTNDatabase);
```



You may choose to directly call Insert or Update in the controller for the table and bypass the Gateway.

Calling Custom Stored Procedures

Update 4.2.2019: When the documentation for calling custom stored procedures was written, the Custom Method Editor was still a decade away.

If you use the Custom Method Editor to create your stored procedure, the files are included for you. Documentation for using the Custom Method editor is included in the DataTier.Net User's guide.

Please check out my YouTube channel, as I will update videos and sample projects as often as I am able. My pesky day job gets in the way of creativity but allows me to eat to continue being creative; thus, a paradox.

<https://www.youtube.com/channel/UCaw0joqvisKr3IYJ9Pd2vHA>

Manually Calling a Custom Stored Procedure

There are three places to make modifications to call a custom stored procedure:

1. The Gateway Class (optional)
2. The Data Writer for the table or view involved
3. The business object that will be passed to the Data Writer.

There are numerous examples in DataTier.Net on how to do this.

One interesting point, to a geek like myself, is that DataTier.Net is used to build Data.Net.

Chicken or the Egg?

As I just mentioned, DataTier.Net was used to build DataTier.Net. Many of the features in DataTier.Net were preexisting conditions, and thus not covered by the update as I didn't rewrite every line of code. What I did do was create a new DataTier.Net project template, rebuilt the DataTier.Net.DataTier.

Procedure Finder

For anyone that might ever need to 'Change' the data tier for a database, there is a useful project located in the Tools folder called Procedure Finder. Procedure Finder will read all the Data Writers for a folder and list all the custom stored procedures that your application calls.

To use Procedure Finder, select the Data Writers Folder when you run the app.

I happen to know the philosophical answer, as I created Data Tier.Net, which was then called Data Class Builder.Net, and later rebranded as RAD Studio Code Generation Toolkit.

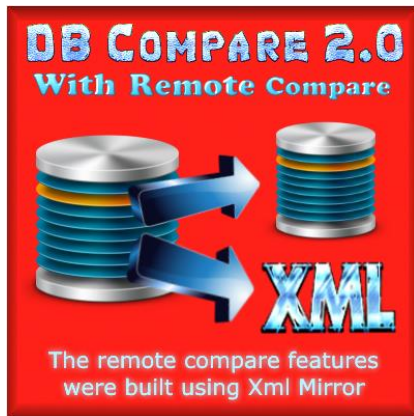
My personal guess is the Chicken in case you are wondering.

Other Project Types

Data Tier.Net was originally designed to be a code generator. This is still the primary function, but there are many uses for Data Tier.Net in addition to being a super awesome, 100% stored procedure powered data tier creator for C# developers (in my completely unbiased opinion).

Sample Projects built using DataTier.Net.

DB Compare



DB Compare compares the schema from two SQL Server databases and reports any schema differences. The core component of DB Compare is the file `DataJuggler.Net.SQLDatabaseConnector`; the same object used by Data Tier.Net to read database schema.

XML Mirror



I created XML Mirror during one of my previous employments, which involved parsing large amounts of XML data.

XML Mirror uses the class `CSharpClassWriter` to create the following files:

[Parser.base.cs](#)

[Parser.custom.cs](#)

`DataJuggler.Net.CSharpClassWriter` is the same file used in `Datatier.Net` to code generate files.

After Xml Mirror was created, I updated DB Compare to perform remote comparisons of a SQL Server database located on a virtual machine, against a database you can connect to via a connection string, or vice versa.

Prior to this feature in DB Compare, I would have to remote into the VM, and bring with me a schema only copy of my database to perform a compare.

I hope after reading the above explanation, you understand one of the reasons all my projects are shared in one large repo.

The other is simply a time saving device. If I tried to manage all these projects, I wouldn't do any of them justice, but since I do use all this code it will get updated.

Data Tier.Net FAQ

Question:

Do I have to use a Data Tier.Net template to create a Data Tier.Net project?

Answer:

No. The Data Tier.Net templates are a time saving tool that you can use if you choose to. If your project would be better suited with a different structure then the templates provide, then feel free to modify your project to suit your needs. Just remember only the template structure has been tested and is supported.

Question:

Data Tier.Net uses multiple projects; can all the required folders be placed in a single project?

Answer:

I once built a single project and it worked, but I prefer the data tier be separated so it is more portable.

Question:

Can a project have multiple databases?

Answer:

The client allows you to create multiple databases and the Data Manager has a databases collection, but I have never built a project using multiple databases in one data tier.

Whenever I use Data Tier.Net for data migrations I create two projects and add them both to one solution. Both projects have their own Data Connector and Gateway. You must manage the multiple connection strings yourself.

Question: I created a project from a Data Tier.Net Template and noticed that there are some files with names like TemporaryDeleteProcedure.cs. Is this class needed for anything?

Answer:

No temporary classes are only used as a placeholder until you have built a Data Tier.Net project and so the references will compile. Feel free to delete these temp files after you have “real” classes in your project.

Final Thoughts

If you like Data Tier.Net and have any positive feedback, questions or comments please create an issue on the Data Juggler Shared Repo on GitHub.

I want to hear what the community wants, as everything up to this point has been what I have wanted.

Data Tier.Net vs Entity Framework

I do not make any claims that DataTier.Net is better than Entity Framework. Entity Framework is Microsoft’s recommended data access component, and I use it at my day job in addition to DataTier.Net. Entity Framework has been tested by thousands of engineers and testers inside of Microsoft, and hundreds of thousands if not millions of developers outside of Microsoft.

My reasons for why I prefer DataTier.Net over Entity Framework are discussed in this video here:

<https://www.youtube.com/watch?v=wFP6sm3pjoY>

Finding Errors with DataTier.Net

If a save fails, finding the last error is very simple in DataTier.Net:

```
// Save the DTNField
tempSaved = gateway.SaveDTNField(ref clone);

// if not saved
if (!tempSaved)
{
    // set to false
    saved = false;

    // get the error, for debugging only
    error = gateway.GetLastException();
}
```

I hope you enjoy DataTier.Net.