Data Tier.Net Quick Start

Setting up Data Tier.Net

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

Create DataTier.Net SQL Server Database

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

Execute the following SQL scripts to create the database schema.

Folder: DataTier.Net\DataTier.Net\Database\SQL Scripts

File: DataTier.Net.Tables and Procedures.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 installed in the DataTier.Net.Database.

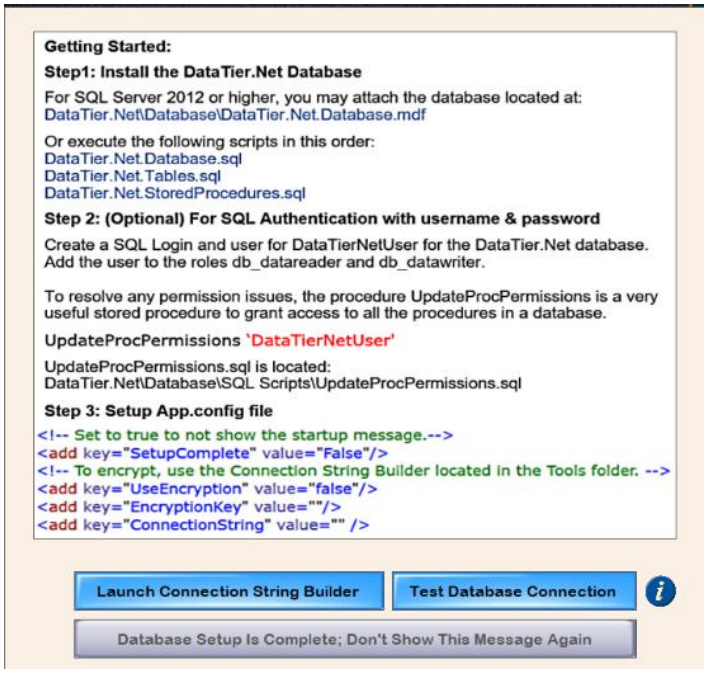
In SQL Server Management Studio, create a new query window and ensure the correct database is selected before running the following query

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).

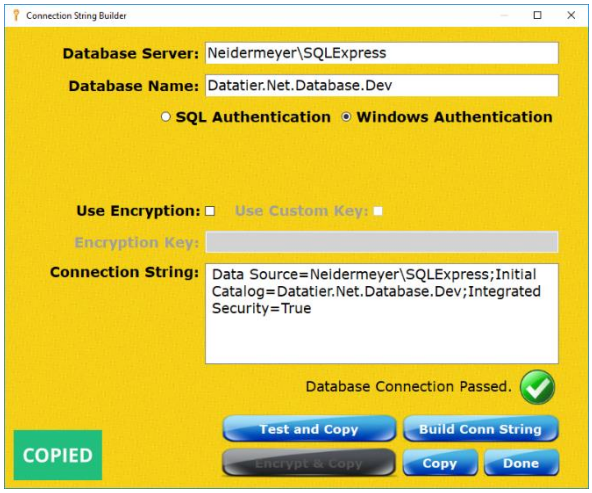
Launch DataTier.Net and click the ‘Launch Connection String Builder’ button.



Tip: After cloning or downloading DataTier.Net from Git Hub, you must build Connection String Builder because the .exes are not stored in the repo.

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

Connection String Builder



This program is self-explanatory.

I keep a shortcut to Connection String Builder on my desktop.
This is another useful tool included with DataTier.Net.

Use Encryption (Optional)

The previous two brandings of DataTier.Net, DataClassBuilder.Net and RAD Studio Code Generation Toolkit required the connection string to be encrypted. As I have gotten older, I must be less paranoid since I have stopped caring if my dog can 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.

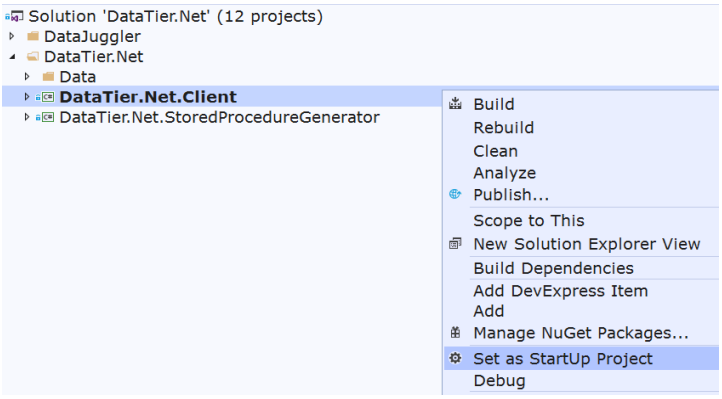
```
<!-- Set the ConnectionString -->
<add key="ConnectionString" value="" />
```

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

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

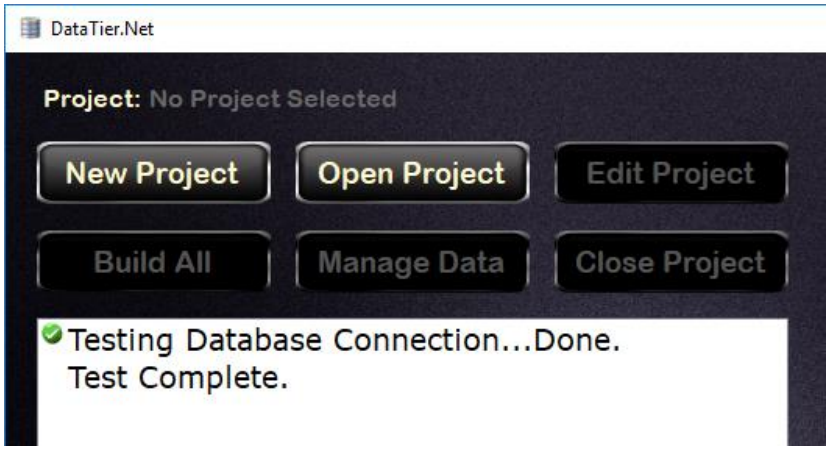
Test Your Setup

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



Database Connection Passed

If your setup is configured correctly, you will see the following message when your run DataTier.Net:

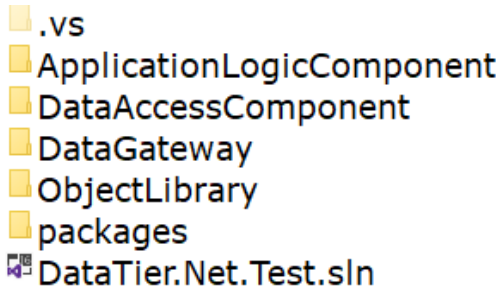


Troubleshooting Connection Issues

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 invalid in the file DataTier.Net.Client.app.config.

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:



Install DataTier.Net Project Templates

Update 7.13.2019: I created a VSIX installer to make installing the project templates much easier. Double click the file:

DataTier.Net\VSIX\DataTier.Net.vsix

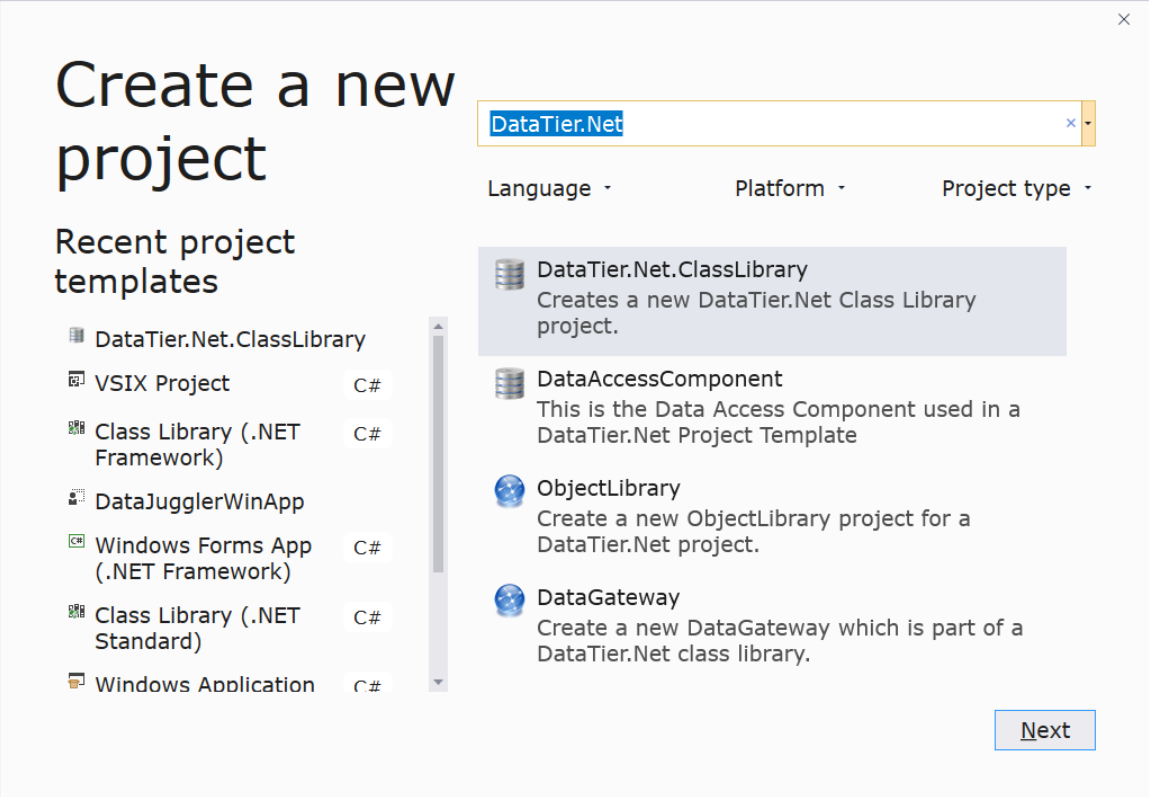
Select Visual Studio 2017 or 2019 or both.

Tip: You will need to restart VS after installation to register the templates.

Once your database connection passes and the templates are installed, you are ready to create your own DataTier.Net projects!

Create a DataTier.Net.ClassLibrary Project in Visual Studio

In Visual Studio select File > New Project

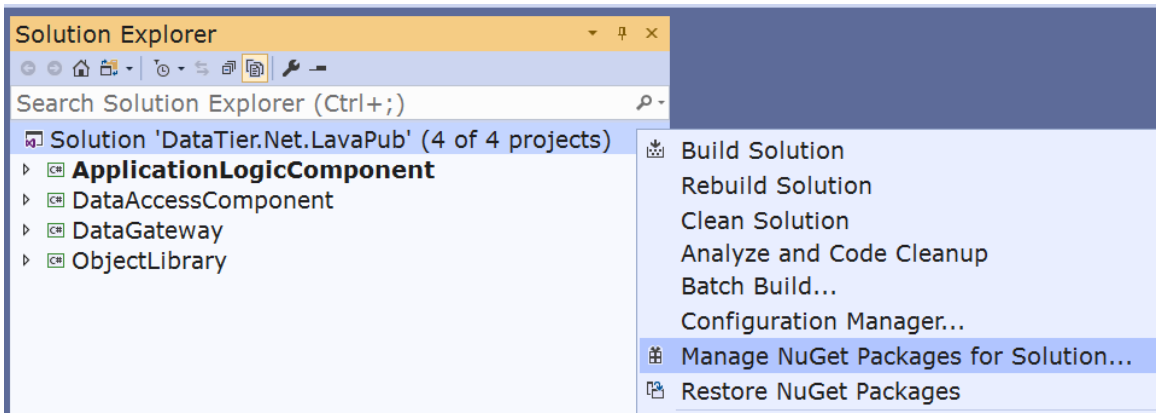


This screenshot is from Visual Studio 2019.

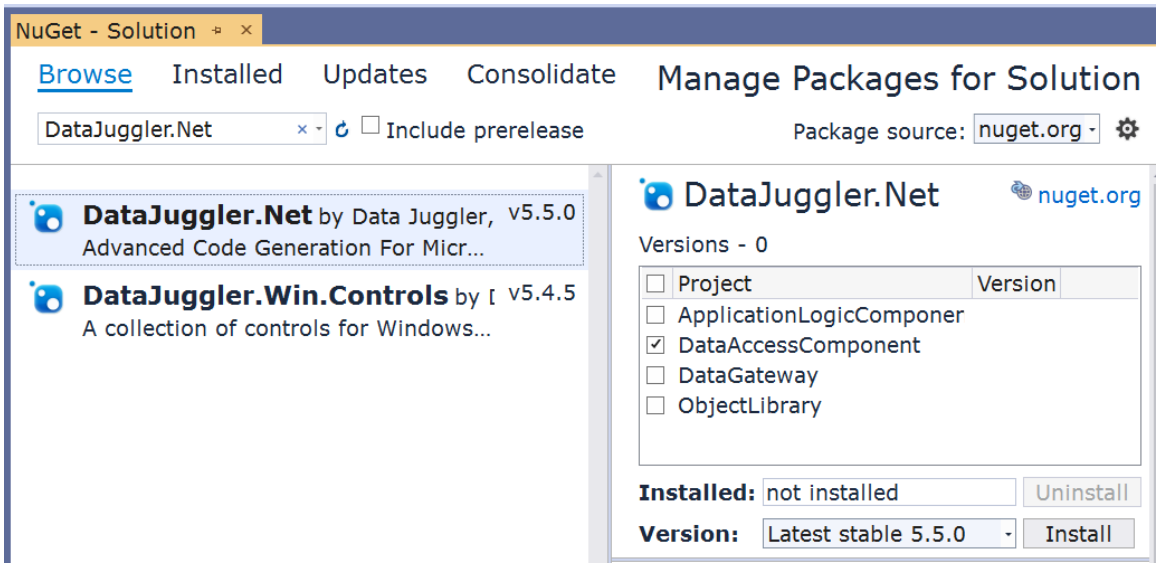
Type in DataTier.Net in the search box and select DataTier.Net.ClassLibrary.

If DataTier.Net.ClassLibrary is not available, install the project templates as described earlier in this section.

Click the Next button.



Once your project is created, right click the solution and select ‘Manage NuGet Packages for Solution’ as shown above.



In the search box, type in DataJuggler.Net.

Select DataJuggler.Net and check the DataAccessComponent project.

Click Install and accept any prompts with ‘OK’.

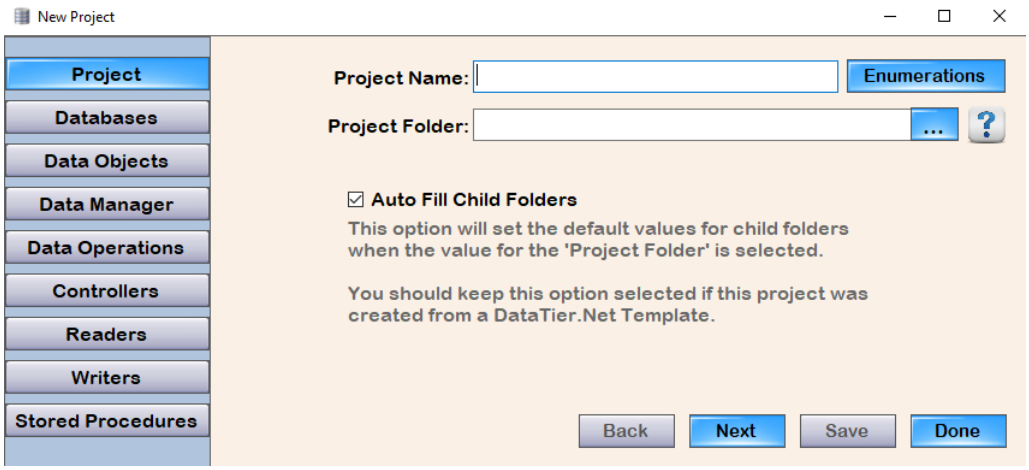
Build your solution in Visual Studio.

You are now ready to build your project in DataTier.Net.

Create a DataTier.Net Project

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

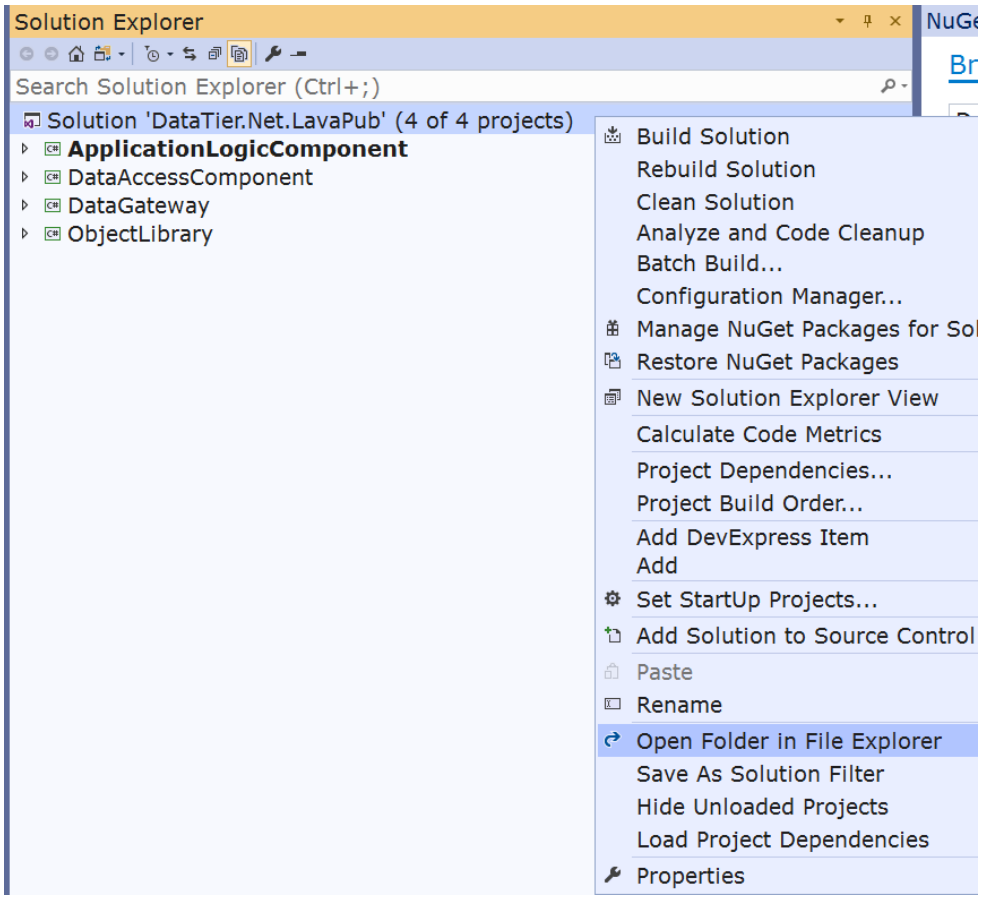
Project Wizard



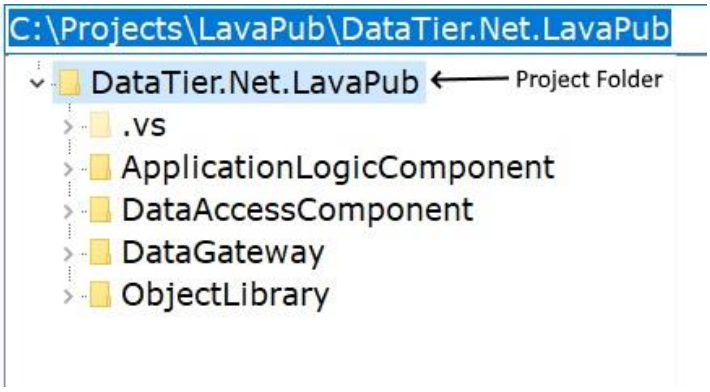
Type in a Name for your project and browse for or enter a Project Folder.

The project folder is the folder above the four projects in your data library.

Tip: An easy way to find your project folder is to right click your solution in Visual Studio and select ‘Open Folder in File Explorer’.

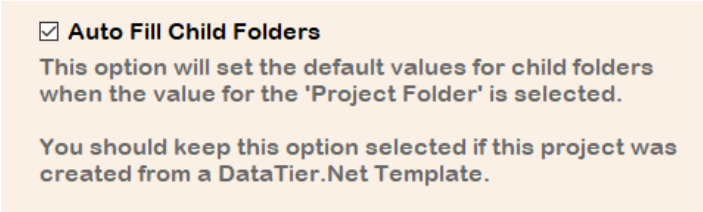


Project Folder for a DataTier.Net Project



The project folder is the folder above the four projects in your solution.

Auto Fill Child Folders

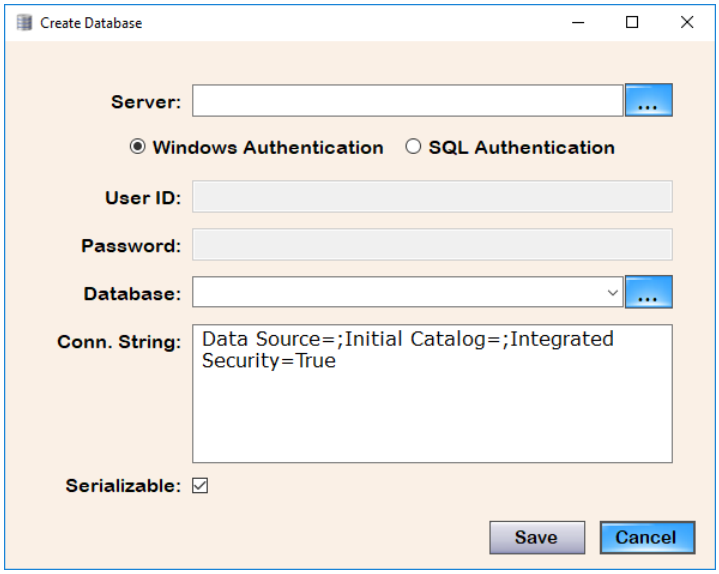


By default, this value is selected and will populate the other tabs of the project wizard for you. Leave this option selected, else use at your own risk.

Add a database to your project.

Click the ‘Next Button’ to select the database tab and click the Add button:

This will launch the Database Editor



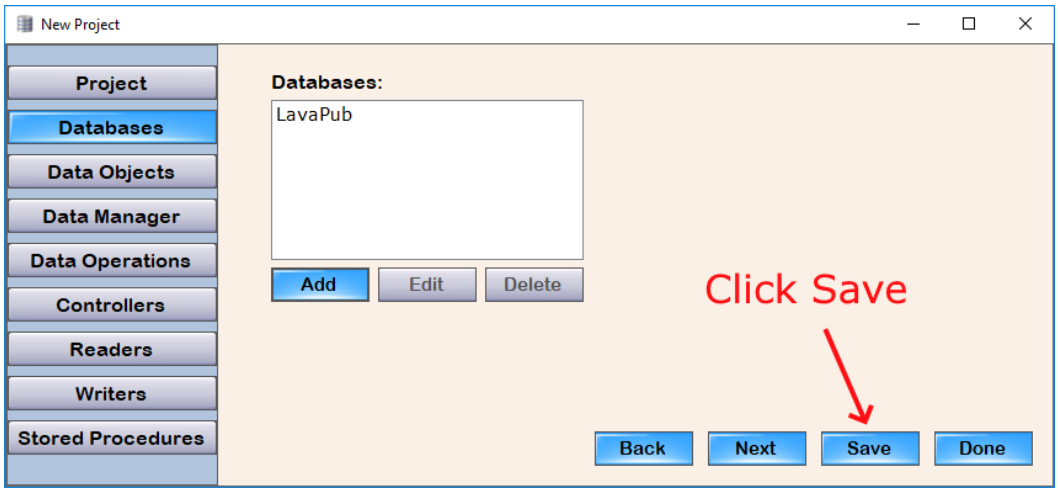
Note: this connection is only used by Data Tier.Net to retrieve the database structure. The connection string for your own DataTier.Net projects is configured in the app.config or web.config files in your solutions.

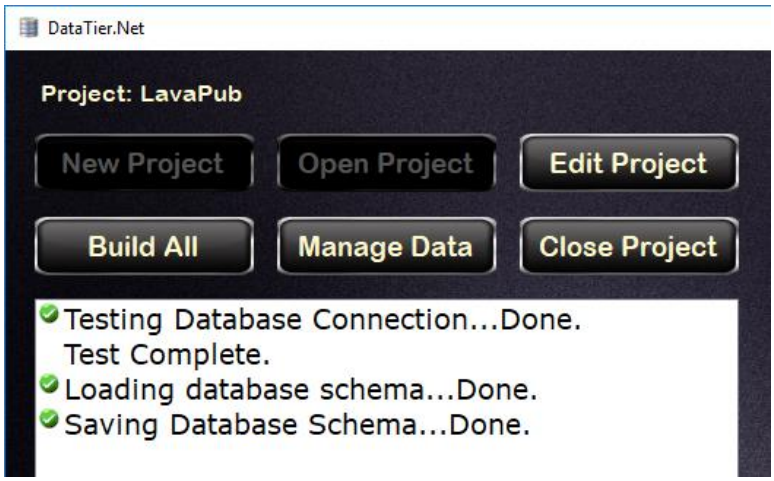
Tip: If you click the Server button to browse for Servers, you may need to add \SQLExpress to the server name detected. This is a known issue.

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

Select the desired database then click the ‘Save’ button to save your database.

Click the ‘Save’ button again on the Project Wizard:



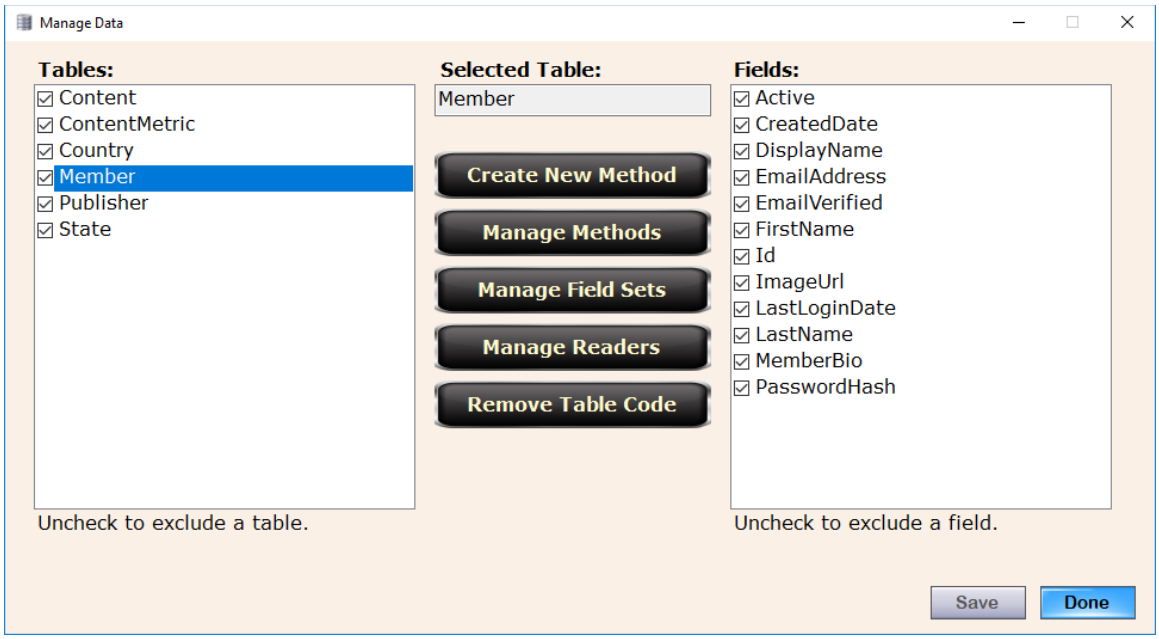


Update: 7.13.2019: Screenshot updated for DataTier.Net.

When you save your project, the database schema is read and then saved.

Manage Data

If you wish to exclude any tables, views or fields from your project, click the Manage Data button before you build.



If you exclude any tables or fields, the Save button will become enabled.

Note: Views are listed as tables, although their behavior is different.

Click ‘Save’ if you make any changes. Click Done to exit this form.

You are now ready to build your project with DataTier.Net!

Build Your Project

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

Include the generated files in your project

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

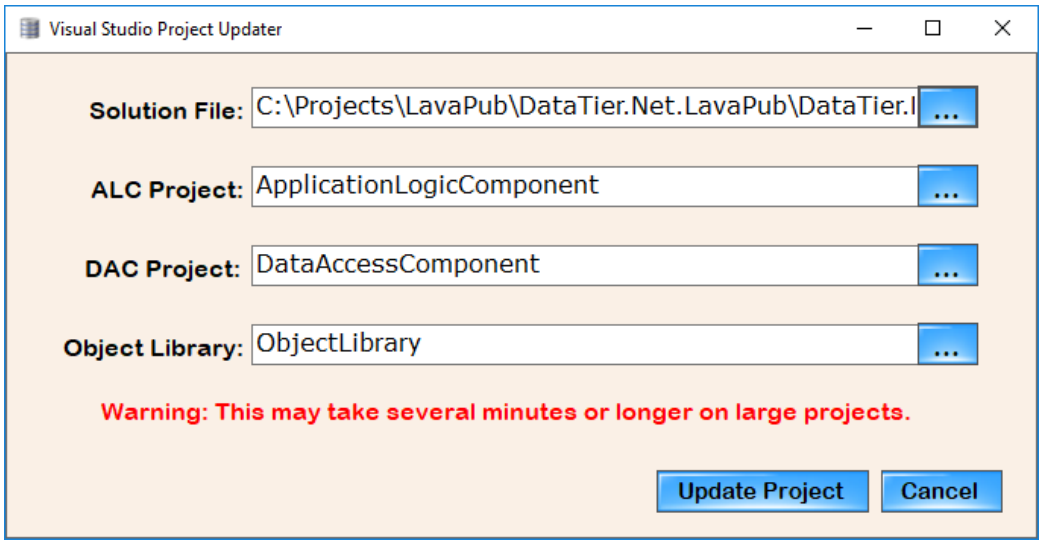
When you rebuild your project with DataTier.Net, you will only have to include new files if you add a new table or view.

If DataTier.Net determines that new files were added, the Visual Studio Project Updater will launch after you build.

Using the Visual Studio Project Updater Control

Step 1: Select the DataTier.Net.ClassLibary VS solution file (.sln).

Data Tier.Net will read the project names from the solution file.



If you created your project from a DataTier.Net.ClassLibrary 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.

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

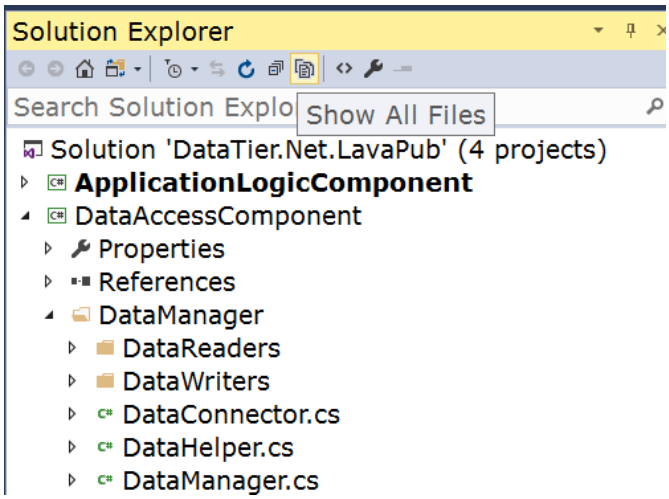
Often, I would forget to click the Include Project Files button, and I would have to manually include all the required project files, and it is quite a tedious job.

You should not have to manually include project files now, but for anyone that ever has this need I will leave the instructions 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.



Toggling 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 is 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

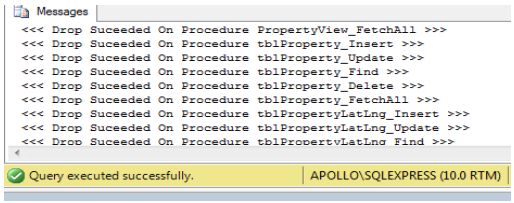
Update 4.2.2019: Click the link 'Stored Procedures.sql' in DataTier.Net

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.

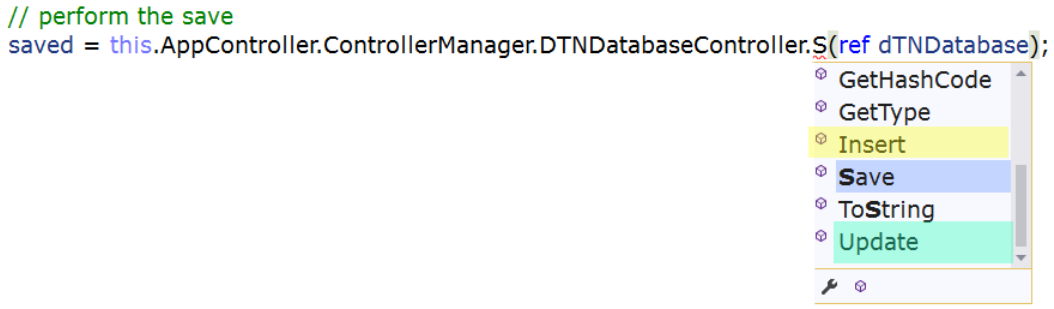
Once I publish, volunteers are invited to help make improvements.

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:



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

Read the DataTier.Net User's Guide for more information.

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/UCaw0joqvisKr3lYJ9Pd2vHA

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.