

# Data Tier.Net Quick Start

## Setting up Data Tier.Net

Update 8.6.2023: **Major Milestone**

DataTier.Net 4.3.2 has been released and now works with .NET 5, .NET 6 and .NET 7.

The project templates are different for .Net Framework and .Net Core.

.NET 5, NET 6 and .NET 7 project templates are installed via NuGet.

### **.Net Framework vs NET 5, .NET 6 and .NET 7**

The only usage difference is with the .NET Core versions (NET 5, .NET 6 and .NET 7 version) are you must create a system environment variable to hold the connection string.

## DataGateway

Create a new instance of a Gateway object, pass in the environment variable name.

### **.NET Core Versions (.NET5, NET 6 and .NET 7)**

// create a new gateway object and pass in the connectionName

```
Gateway gateway = new Gateway(connectionName);
```

// .NET Framework, the connection string is read from the app or web.config

```
Gateway gateway = new Gateway();
```

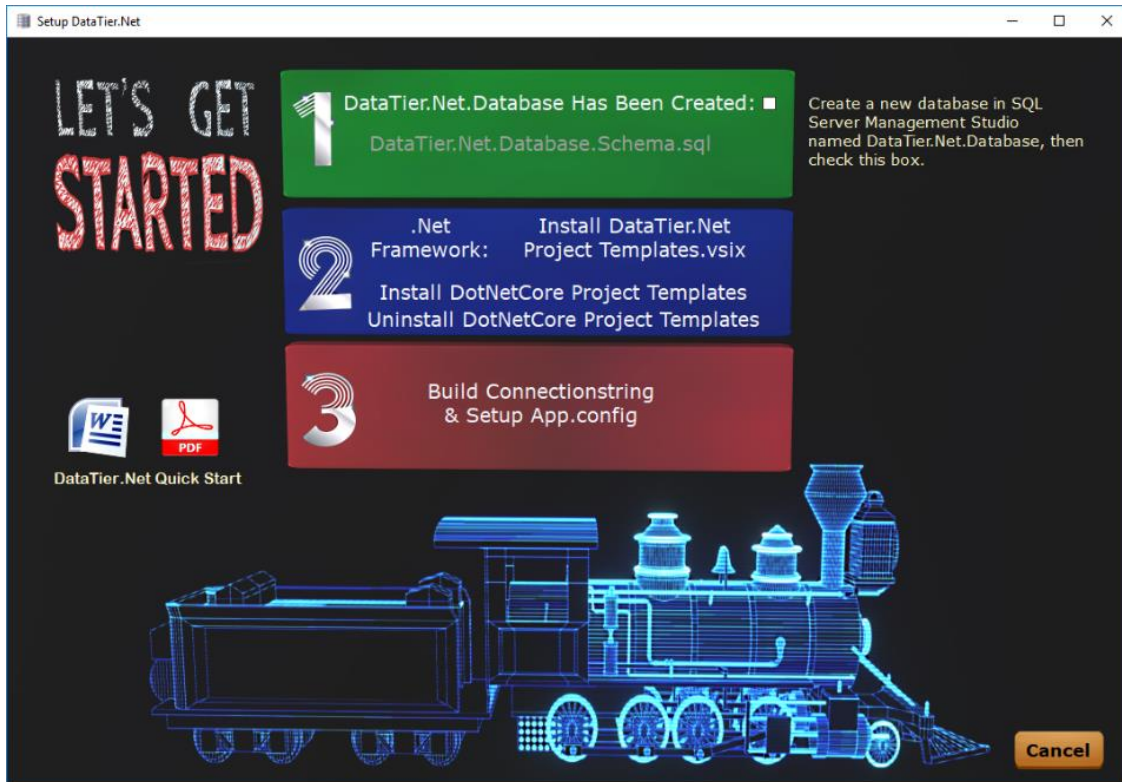
## Installing Project Templates

The Setup Control (shown on the next page), has options for .NET Framework and .NET 5, .NET 6 and .NET 7.

The .NET Framework project templates are for version 4.8.

Update 7.21.2019: I created a new setup control:

There are three steps to setting up DataTier.Net:



## Step 1. Create DataTier.Net SQL Server Database

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

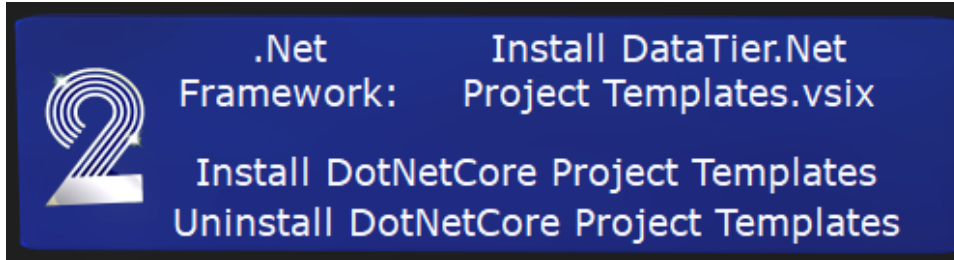
Check the box 'DataTier.Net.Database has been created'.  
This will enable the link to the file:



Execute this SQL script to create the tables and procedures:

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

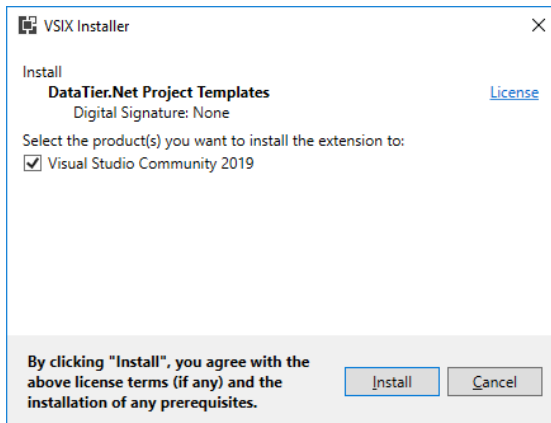
## Step 2: Install DataTier.Net Project Templates:



## .NET Framework

Click the link 'Install DataTier.Net Project Templates.vsix'.

This will launch the Visual Studio Extension Installer:



**Note: .NET Framework Project Templates Have Been Depreciated.**

## **.NET 5, .NET 6 and .NET 7**

You can now skip step 2 on the Setup Control, and the project templates can be installed via the Project Editor and dotnet CLI.

### **Uninstall**

For .NET Framework, manage extensions in Visual Studio.

For .NET5 or .NET6, click the Uninstall Project Templates buttons.

### **Command Line**

```
dotnet new -u DataJuggler.DataTier.Net5.ProjectTemplates
```

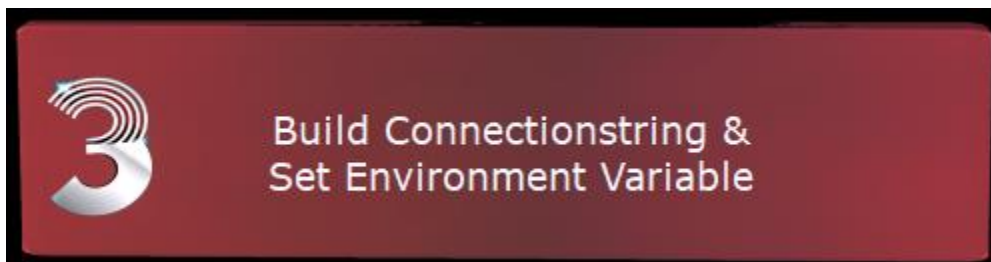
or

```
dotnet new -u DataJuggler.DataTier.Net6.ProjectTemplates
```

or

```
dotnet new -u DataJuggler.DataTier.Net7.ProjectTemplates
```

### **Step 3: Build Connectionstring & Setup App.config**



Click the link 'Build Connectionstring & Setup Environment Variable'.

As the screenshot above indicates, this will launch the `ConnectionStringBuilderForm`.

## 8.6.2023: New Connection String Builder



The screenshot shows the 'Connection String Builder' application window. It features a light beige background with a yellow key icon in the top-left corner of the title bar. The window contains several input fields and controls:

- Database Server:** An empty text input field.
- Database Name:** A text input field containing 'DataTier.Net.Database'.
- Authentication:** Two radio buttons; 'SQL Authentication' is selected (indicated by a blue dot), and 'Windows Authentication' is unselected.
- Database User:** An empty text input field.
- Database Password:** An empty text input field.
- Encryption Note:** A small text block stating: 'Microsoft.Data.SqlClient requires you to set Encrypt=False if your database is not encrypted. Leave this checked if you are targeting .NET Core.'
- Include Encrypt:** A checkbox that is checked with a blue checkmark.
- Encrypt Value:** A dropdown menu currently showing 'False'.
- Connection String:** A large, empty text area for the generated connection string.
- Buttons:** At the bottom, there are four buttons: 'Test' (a small grey button), 'Install Connection String' (a wide grey button), 'Build Conn String' (a grey button), and 'Copy' (a grey button). To the right of 'Copy' is an orange 'Cancel' button.

The old Connection String Builder was a stand-alone application and is still in the Tools folder. This works fine, although it added an extra couple of minutes to every tutorial video. I had to explain you must compile Connection String Builder before the 'Launch Connection String Builder' button is clicked because Git Hub does not include executables when you clone.

Note: Microsoft.Data.SqlClient 5.0+ requires you to specify Encrypt=false in your connectionstring, or it will not work.

## Using Connection String Builder

### Windows Authentication (recommended)

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

### SQL Server Authentication

**SA login:** If you are using the sa (system administrator) account on SQL Server then you can skip the next section.

### SQL Server Login:

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

After you fill in the authentication options, the Build Conn String button will become enabled:

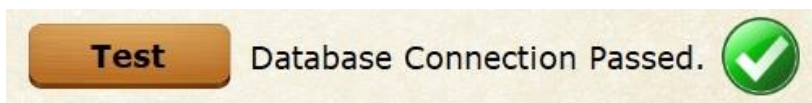


Once you build your connection string, the other buttons will become enabled as shown on the picture at the bottom of the previous page.

### Test Database Connection

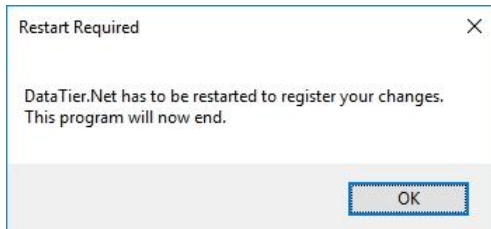
You should test your database connection before you click the 'Install Connectionstring & Update App.config' button.

Provided your Connectionstring works, you should see this message:



Five seconds after you see the installed message, the Connection String Builder Form closes, and you are shown a message that a restart is required:

## Restart Required



The .Net Framework configuration files cannot be reloaded after app start.

Note: You must update the app.config running DataTier.Net in Visual Studio. I frequently run the executable version via a shortcut on my desktop, but this is after setup is complete.

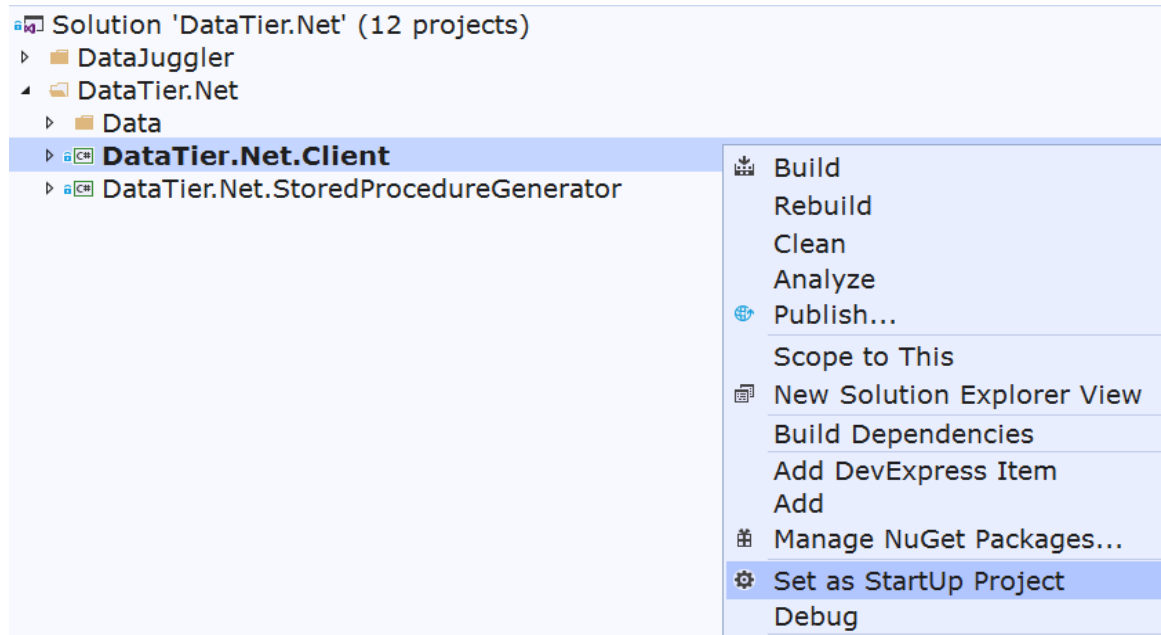
## Run Setup Again

To run the Setup Control again, click the Run Setup Again button:

Note: The Setup Control only updates the app.config. I could update DataTier.Net.exe.config also, but the Setup Control is intended to help new users get up and running with DataTier.Net quickly.

## 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.



## 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.

## 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.

## More Info About DataTier.Net Project Templates

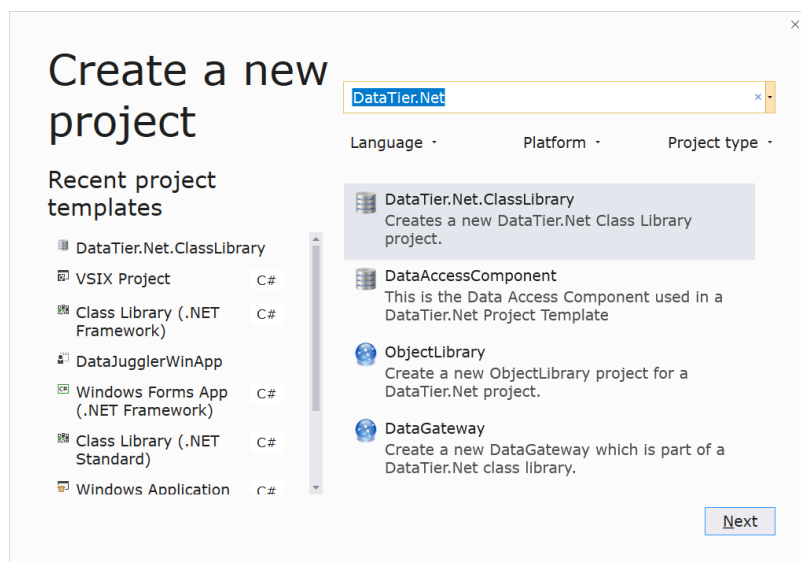
The project templates are used to help you create your own DataTier.Net projects by creating a Class Library which includes all the required references and projects required to build a DataTier.Net solution. The image below shows the directory structure of a Data Tier.Net Project Template:

- .vs
- ApplicationLogicComponent
- DataAccessComponent
- DataGateway
- ObjectLibrary
- packages
- DataTier.Net.Test.sln

## .Net Framework Creating a DataTier.Net DataTier (Deprecated)

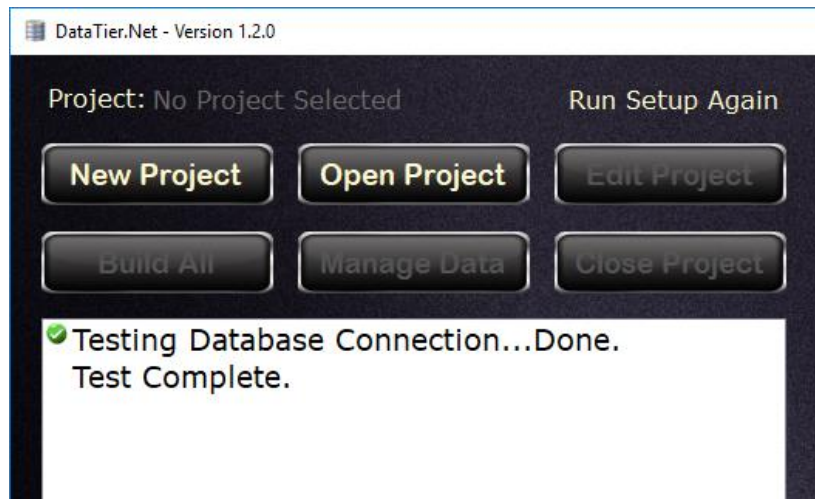
### Create a DataTier.Net.ClassLibrary Project in Visual Studio

In Visual Studio select File > New Project



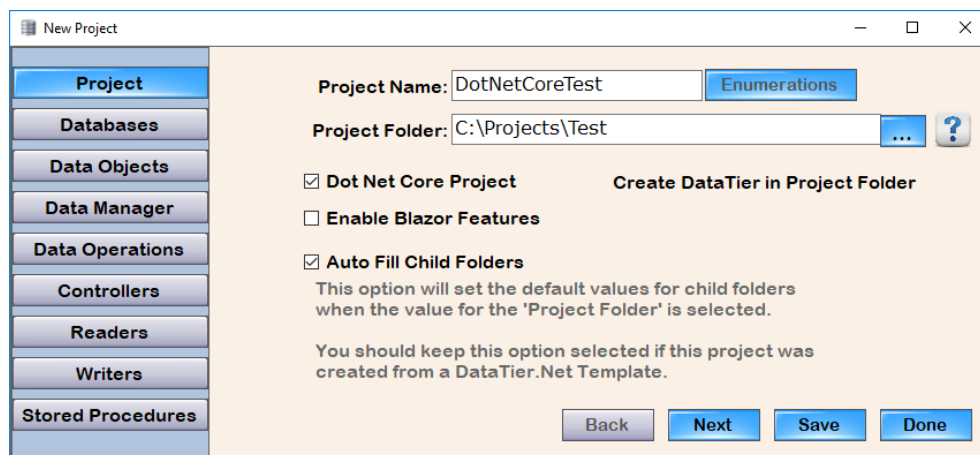
This screenshot is from Visual Studio 2019.

## Create a DataTier.Net Project



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

## New 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'.

## Creating .NET Core Projects (.NET5, .NET 6 and .NET 7)

.Net Core projects are different as you start by creating a new project.

Click the New Project button and set the name and project folder.

Check the box for Dot Net Core Project, and this will show the Create Data Tier

New Project

Project Name: DotNetCoreTest Enumerations

Project Folder: C:\Projects\Test ... ?

☒ Dot Net Core Project **Create DataTier in Project Folder**

☐ Enable Blazor Features

☒ Auto Fill Child Folders

[Click To Create The Four Projects In a DataTier.Net.DataTier.](#)

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.

Back Next Save Done

Ignore the .Net Framework instructions below when they tell you to create a new project. Other than that, almost everything is the same with .Net Framework and .Net Core.

### Enable Blazor Features:

If your project is a Dot Net Core project, you have the option to enable Blazor Features. If enabled, new features are available.

Edit Project

Project Name: Password Vault Enumerations

Project Folder: C:\Projects\GitHub\PasswordVault\DataTier.Net ... ?

☒ Dot Net Core Project **Create DataTier in Project Folder**

☒ Enable Blazor Features Bindings: Allow Binding

☒ 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.

Back Next Save Done

## Bindings

I added a new enumeration to handle the Binding Type.

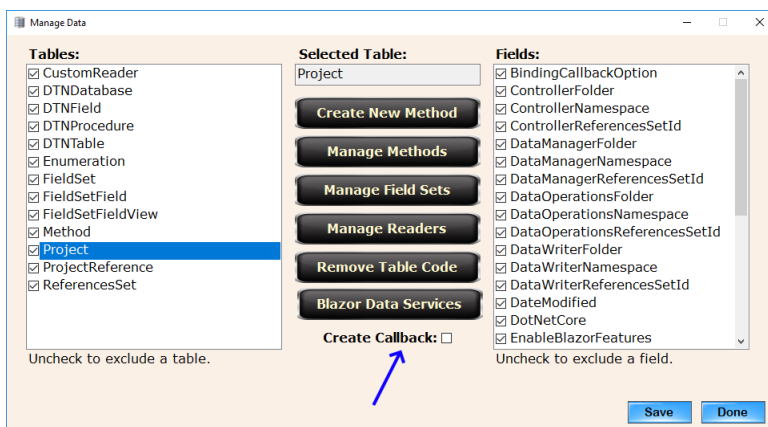
You can select No Bindings, Allow Binding & Create Bindings.

### No Binding

No Binding is the default option and your data objects will not be generated with callbacks enabled if changes occur.

### Allow Binding

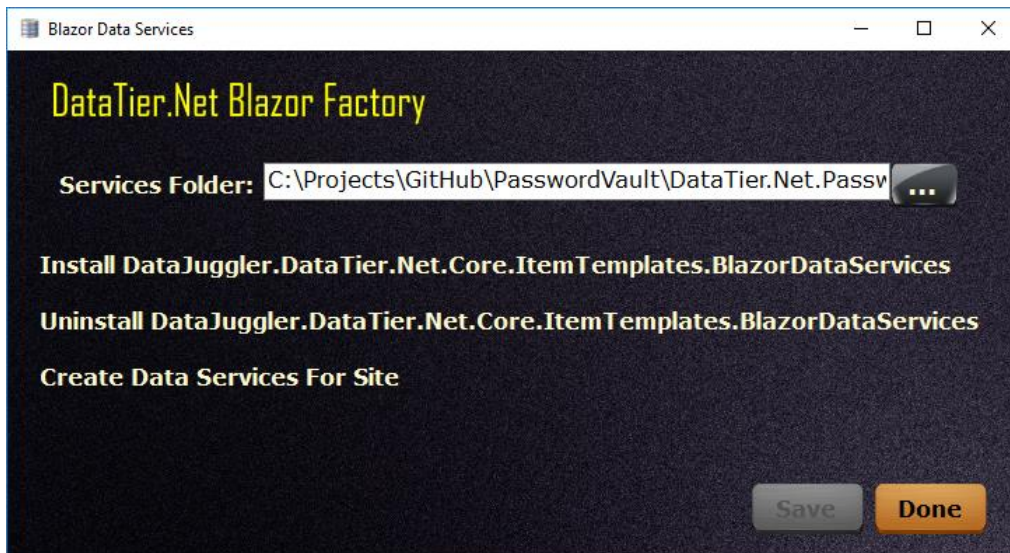
With Allow Binding, you must turn on binding for any tables you want binding.



I published a project on Git Hub with a working example of callbacks:  
<https://github.com/DataJuggler/BlazorToDoList>

## Blazor Data Factory

After I built my first project with Blazor, I realized I needed a way to intercept calls when changes occur so I can save the changes. I suspect as Blazor matures Microsoft is going to rethink their not letting you have an OnChange event with a bind property set. Until then, I think this is cool the way I did it.



### Services Folder

The Services Folder defaults to DataGateway\Services, but you may change it.

#### Create Data Services (for table)

First you must install the Nuget package DataJuggler.DataTier.Net.Core.ItemTemplates.BlazorDataServices.

Then click the Create Data Services For (table) button.

This will create a DataWatcher and a DataService class tailored for the Selected Table.

#### Example Data Watcher & Data Service:

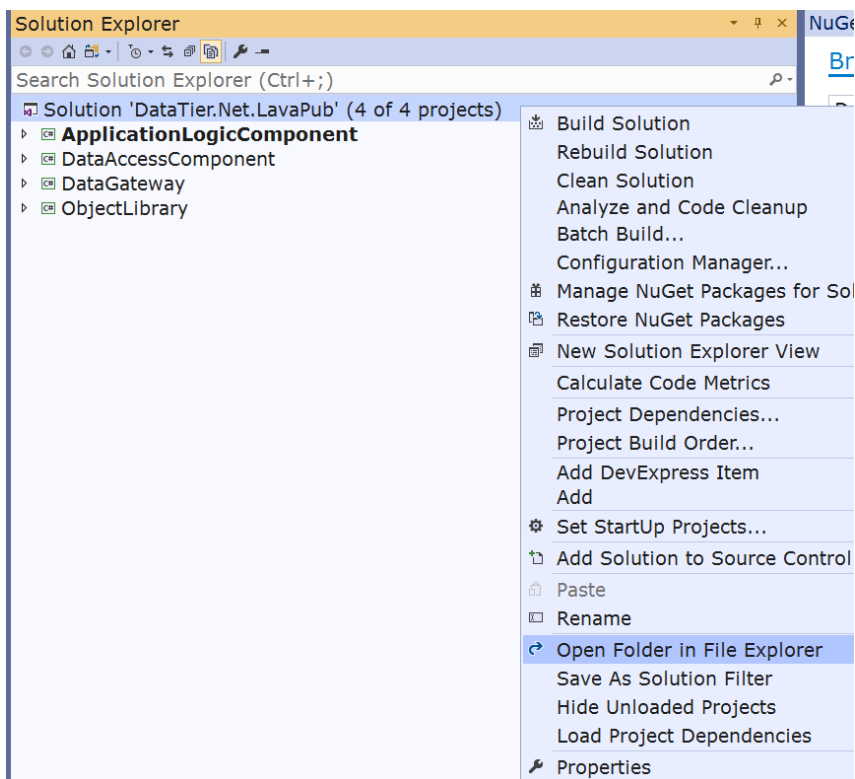
<https://github.com/DataJuggler/BlazorToDoList/tree/master/Data/DataGateway/Services>

## Data Watcher

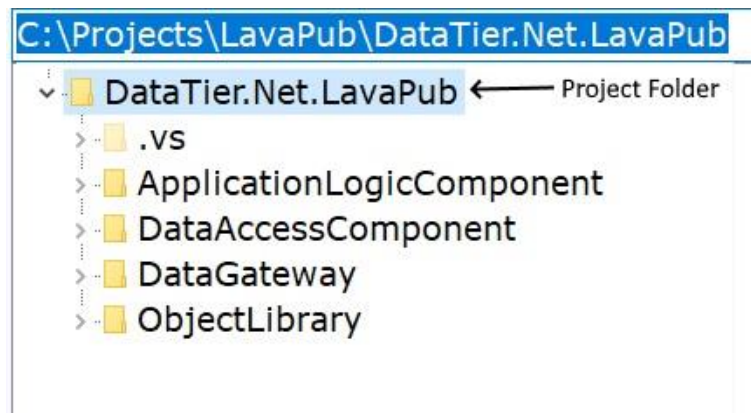
The Data Watcher is set after you load the data from SQL. Whenever data is saved or reloaded, the data watcher is set again to detect any changes.

## Data Service

The Data Service calls Gateway methods and creates an easy way for the website to communicate with the data tier.



## Project Structure of a DataTier.Net Project



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

## 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.

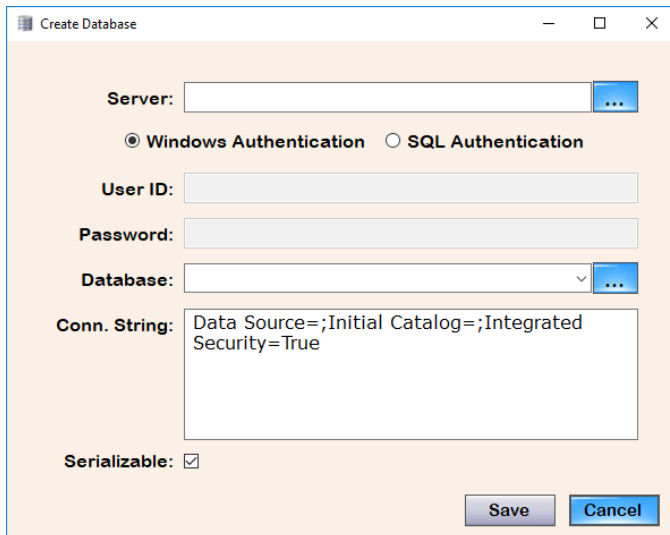
You should keep this option selected if this project was created from a DataTier.Net Template.

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 'Next Button' to select the database tab and click the Add button:

This will launch the Database Editor



The screenshot shows a 'Create Database' dialog box with the following fields and options:

- Server:** A text input field with a browse button (three dots) to its right.
- Authentication:** Two radio buttons: ☒ Windows Authentication and ☐ SQL Authentication.
- User ID:** A text input field.
- Password:** A text input field.
- Database:** A dropdown menu with a browse button (three dots) to its right.
- Conn. String:** A text area containing the default connection string: `Data Source=;Initial Catalog=;Integrated Security=True`.
- Serializable:** A checkbox that is currently checked.
- Buttons:** 'Save' and 'Cancel' buttons at the bottom right.

**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.

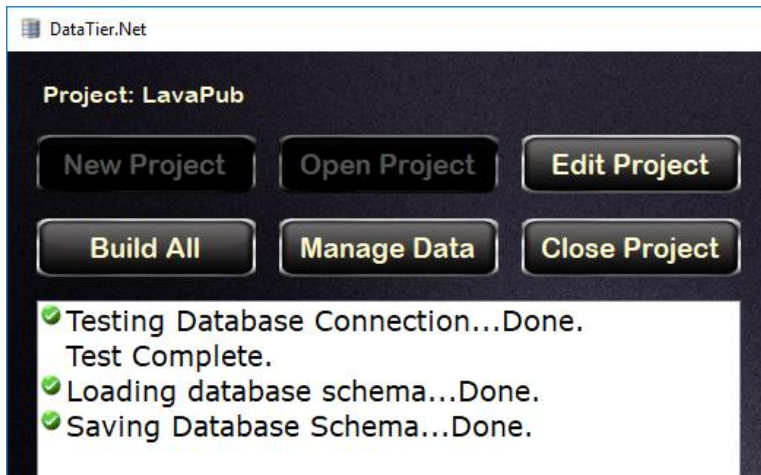
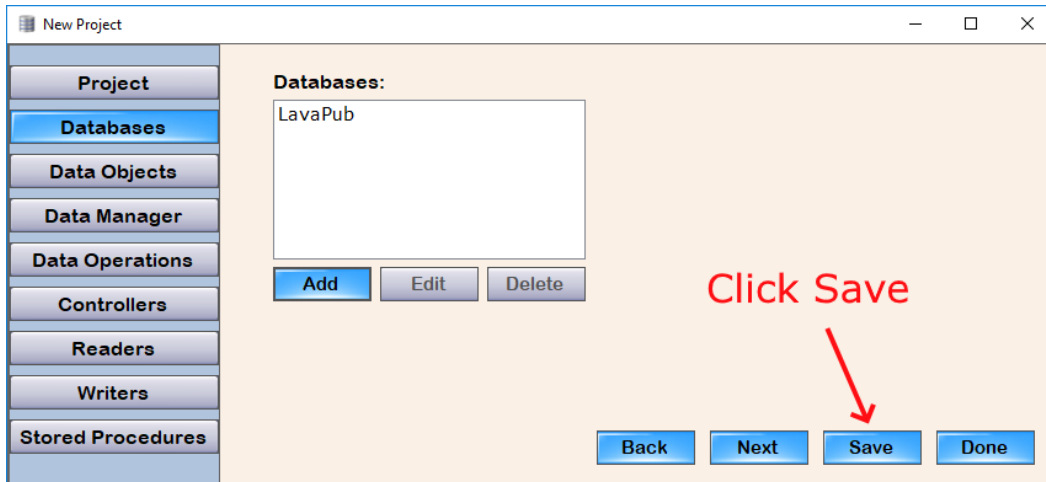
**Tip:** If you click the Server button to browse for Servers, you may need to add \SQLEXPRESS to the server name detected. This is a known issue.

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.

**Manage Data**

**Tables:**

- ☒ CustomReader
- ☒ DTNDatabase
- ☒ DTNField
- ☒ DTNProcedure
- ☒ DTNTable
- ☒ Enumeration
- ☒ FieldSet
- ☒ FieldSetField
- ☒ FieldSetFieldView
- ☒ Method
- ☒ **Project**
- ☒ ProjectReference
- ☒ ReferencesSet

Uncheck to exclude a table.

**Selected Table:**

Project

Create New Method

Manage Methods

Manage Field Sets

Manage Readers

Remove Table Code

Blazor Data Services

Create Callback: ☐

**Fields:**

- ☒ BindingCallbackOption
- ☒ ControllerFolder
- ☒ ControllerNamespace
- ☒ ControllerReferencesSetId
- ☒ DataManagerFolder
- ☒ DataManagerNamespace
- ☒ DataManagerReferencesSetId
- ☒ DataOperationsFolder
- ☒ DataOperationsNamespace
- ☒ DataOperationsReferencesSetId
- ☒ DataWriterFolder
- ☒ DataWriterNamespace
- ☒ DataWriterReferencesSetId
- ☒ DateModified
- ☒ DotNetCore
- ☒ EnableBlazorFeatures

Uncheck to exclude a field.

Save Done

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.

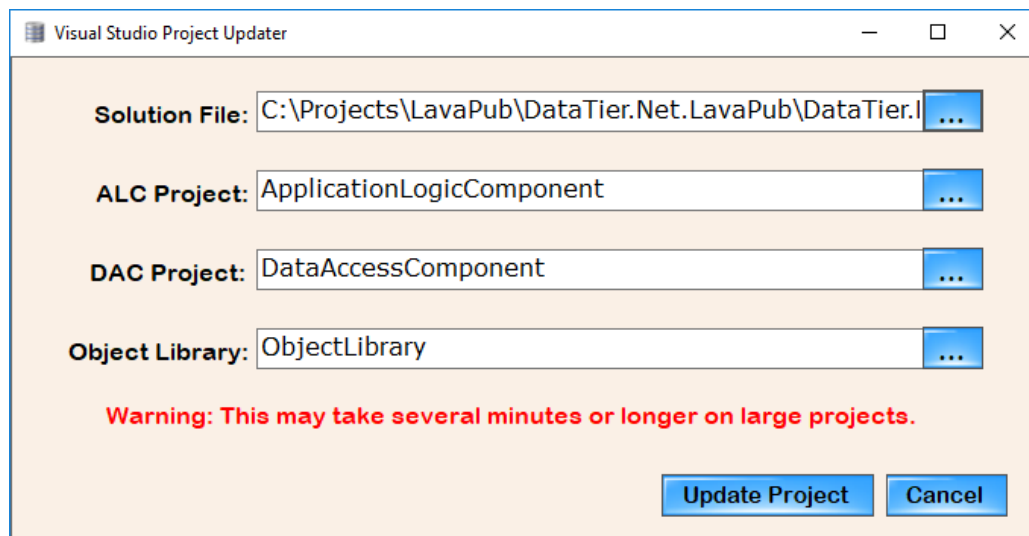
If DataTier.Net determines that new files were added, the Visual Studio Project Updater will launch after you build.

## Using the Visual Studio Project Updater Control

### .Net Framework Only

Step 1: Select the DataTier.Net.ClassLibrary 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.

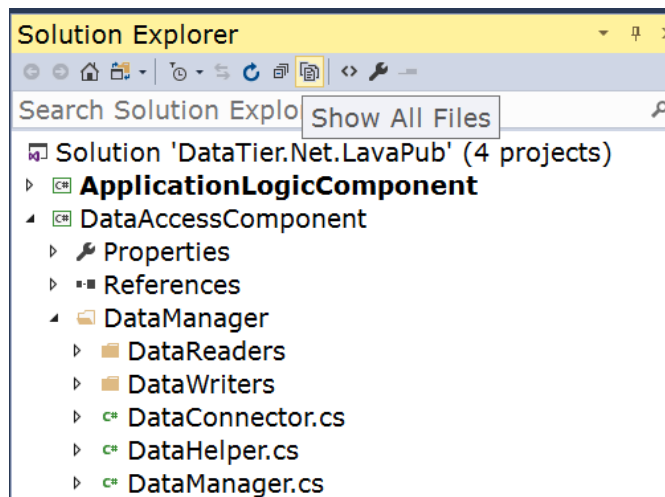
Often, I would forget to click the Include Project Files button, and I would have to manually include all the required project files, and it is quite a tedious job.

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

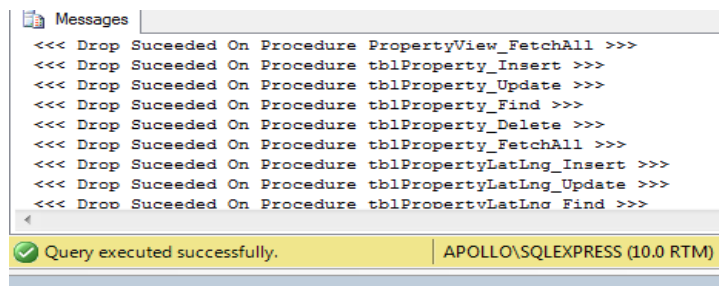
### Update 4.2.2019: Click the link 'Stored Procedures.sql' in DataTier.Net

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 point 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.

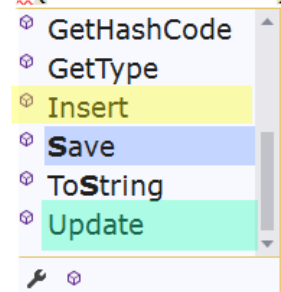
## Save Method

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.

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

## Tutorial Videos

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.

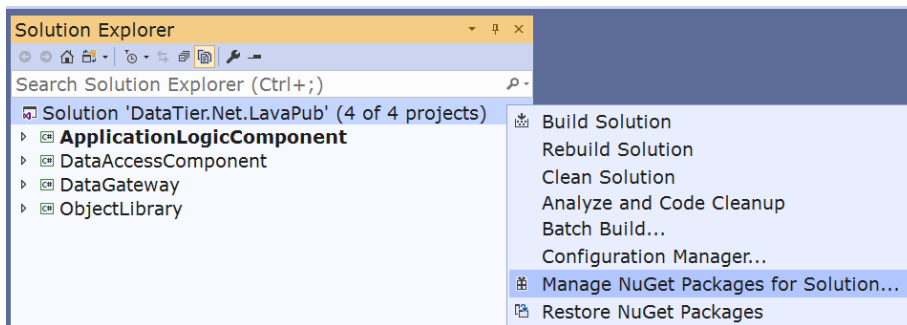
<https://www.youtube.com/DataJuggler>

**Legacy info for .NET Framework. Not Needed for .NET Core.**

## Install DataJuggler.Net NuGet Package

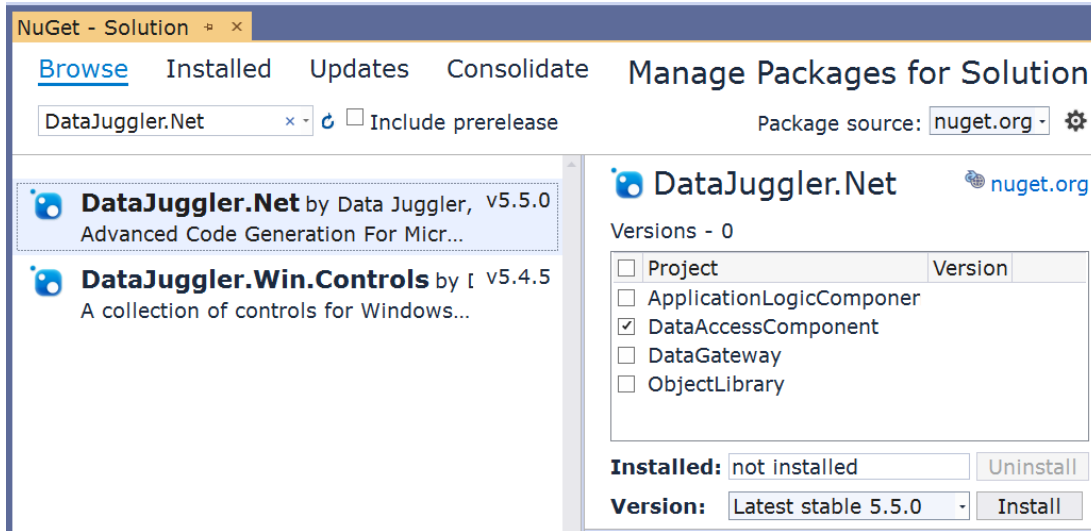
.Net Framework Only. Dot Net Core packages are already installed.

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.

**Sample Projects built using DataTier.Net.**

**DB Compare** <https://github.com/DataJuggler/DBCompare>



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** <https://github.com/DataJuggler/XmlMirror>



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.

I hope you enjoy DataTier.Net