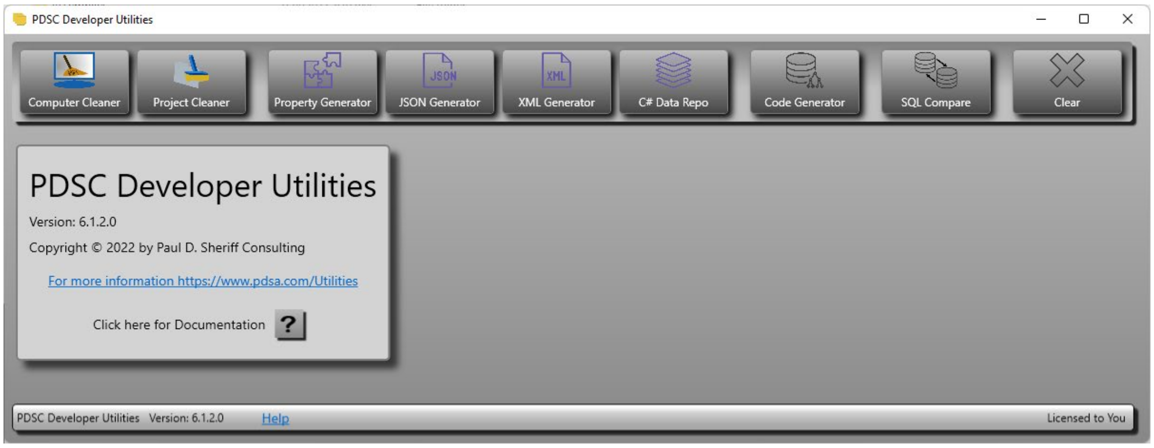
Chapter 2: PDSC Developer Utilities Usage

The PDSC Developer Utilities (Figure 1) is a set of tools to help you develop your .NET applications and keep your development environment clean and working as efficient as it can. This chapter gives you an overview of the various utilities and describes the installation of the tool.



PDSC Developer Utilities

Version: 6.1.2.0

Copyright © 2022 by Paul D. Sheriff Consulting

For more information https://www.pdsa.com/Utilities

Click here for Documentation ?

PDSC Developer Utilities Version: 6.1.2.0 Help Licensed to You

Figure 1: Screen shot of the PDSC Developer Utilities

Overview of the Developer Utilities

After installing the PDSC Developer Utilities you will have the following programs that you can run.

Utility	Description
Computer Cleaner	Visual Studio and .NET are great development environments for creating applications quickly. However, they tend to leave a lot of miscellaneous folders and files all over your hard drive. This utility recycles these folder and files to free up hard drive space.

Utility	Description
Project Cleaner	This tool goes through Visual Studio or VS Code project folders and recycles several folders that are not needed and can be regenerated automatically next time you build your application. You can optionally have it look in .SLN, VBProj, CSProj files and eliminate any references to source control. It can also remove any read-only attributes from the files. This utility is configurable so you can choose what folders and files you wish to recycle.
Property Generator	This utility generates C# or Visual Basic property statements. There are several templates (like the snippets in the Visual Studio editor) from which you can choose. You can also create your own templates to generate any type of property you want.
JSON Generator	This utility allows you to choose a table or view and generates a JSON file of the data.
XML Generator	This utility allows you to choose a table or view and generates an XML file of the data. Optionally, an XSD file of the schema of the table or view can also be generated.
C# Data Repo Generator	This utility generates a repository class that returns hard-coded data that you select from a table in one of your database tables. When creating exercises for a training class, or to test some functionality, but you don't want to have to connect to a database, it is nice to have a collection of hard-coded data that can be returned. Instead of you having to create all this data by hand, if you have a table with data, you can use this generator to select some data and have it hard-coded into a repository class.
Code Generator	This utility generates Entity, Repository, View Model and Search classes from a table. It can also generate a complete set of CRUD MVC pages.
C# Application Creator (COMING SOON)	This utility copies all the files and folders from the where you installed the PDSC Framework template project to a new folder and name that you specify. It then renames the appropriate files to the new application name you specify.
SQL Server Schema Compare	This utility compares two SQL Server databases to determine any tables, constraints, stored procedures, views, etc. that are missing between the two databases.

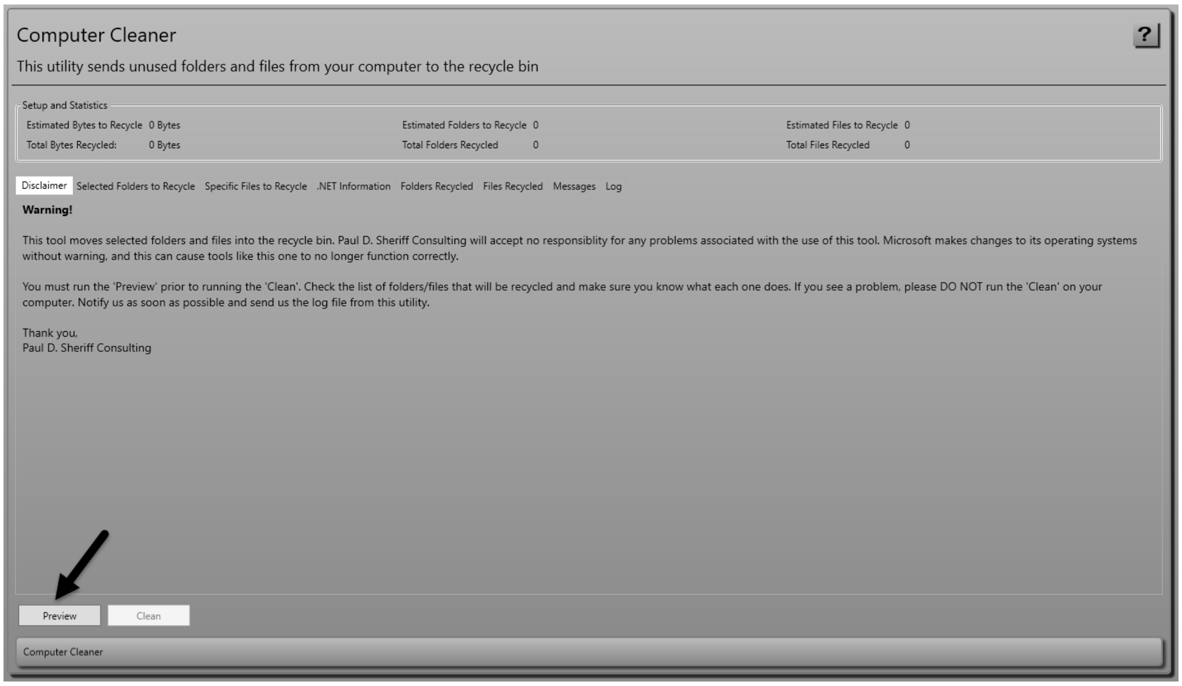
Table 1. List of PDSC Developer Utilities

Computer Cleaner

Visual Studio, Visual Studio Code, the .NET Framework, and .NET Core are great development environments for creating applications quickly. However, the sometimes leave a lot of miscellaneous files all over your hard drive. There are a few locations on your hard drive that you should check to see if there are left-over folders or files that you can delete. I have attempted to gather as much data as I can about the various versions of .NET and operating systems. Of course, your mileage may vary on the folders and files listed here. This utility attempts to find the various folders depending on which version(s) of Visual Studio, VS Code, the .NET Framework, and .NET Core you have installed on your machine.

Disclaimer Tab

When you first come into the Computer Cleaner utility, a disclaimer tab (Figure 2) is displayed. We provide you with the warning that this tool is going to recycles files on your computer into the Recycle Bin. It puts them into the Recycle Bin so you can retrieve them if necessary. As there is always the potential for things to go wrong as Microsoft changes their operating systems frequently, it is good to be able to recover these recycled files. Please note that PDSC does not take any responsibility for the use of this tool on your machine.


Computer Cleaner

This utility sends unused folders and files from your computer to the recycle bin

Setup and Statistics

Estimated Bytes to Recycle	0 Bytes	Estimated Folders to Recycle	0	Estimated Files to Recycle	0
Total Bytes Recycled:	0 Bytes	Total Folders Recycled	0	Total Files Recycled	0

Warning!

This tool moves selected folders and files into the recycle bin. Paul D. Sheriff Consulting will accept no responsiblity for any problems associated with the use of this tool. Microsoft makes changes to its operating systems without warning, and this can cause tools like this one to no longer function correctly.

You must run the 'Preview' prior to running the 'Clean'. Check the list of folders/files that will be recycled and make sure you know what each one does. If you see a problem, please DO NOT run the 'Clean' on your computer. Notify us as soon as possible and send us the log file from this utility.

Thank you,
Paul D. Sheriff Consulting

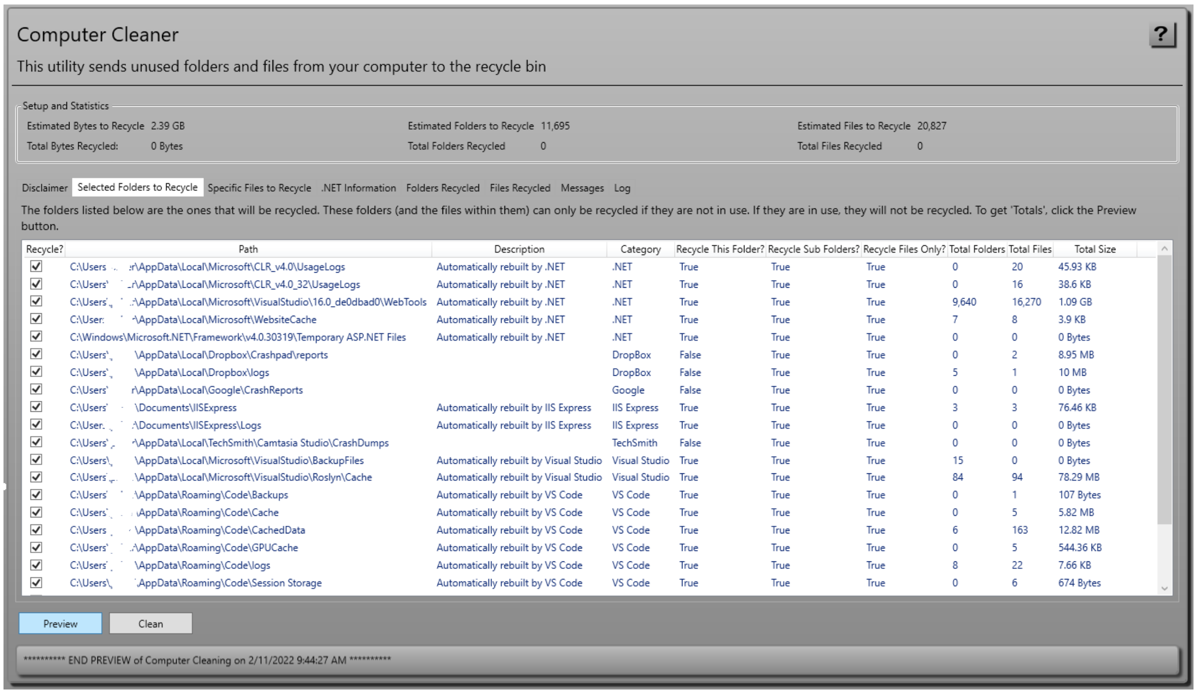
Figure 2: Disclaimer screen

The first thing you need to do is to click on the **Preview** button in the lower left-hand corner. This will provide you with a preview of the folders and/or files that are going to be recycled. The list of folders that are going to be recycled is contained in the **ComputerCleaner-FoldersToRecycle.xml** file located in the **PDSCDeveloperUtilities** folder in the **My Documents** folder on your computer.

NOTE:	Clicking on the Preview button can take a few minutes depending on how many folders and files are on your hard drive.

Selected Folders to Recycle Tab

After clicking on the **Preview** button, the list of folders that will be recycled (Figure 3). Be sure to review this list carefully. You may unselect any folders that you do not wish to recycle by unchecking the check box under the Recycle? Column next to the folder you don't want to recycle.



Computer Cleaner

This utility sends unused folders and files from your computer to the recycle bin

Setup and Statistics

Estimated Bytes to Recycle	2.39 GB	Estimated Folders to Recycle	11,695	Estimated Files to Recycle	20,827
Total Bytes Recycled:	0 Bytes	Total Folders Recycled	0	Total Files Recycled	0

The folders listed below are the ones that will be recycled. These folders (and the files within them) can only be recycled if they are not in use. If they are in use, they will not be recycled. To get 'Totals', click the Preview button.

Recycle?	Path	Description	Category	Recycle This Folder?	Recycle Sub Folders?	Recycle Files Only?	Total Folders	Total Files	Total Size
✓	C:\Users[illegible]r\AppData\Local\Microsoft\CLR_v4.0\UsageLogs	Automatically rebuilt by .NET	.NET	True	True	True	0	20	45.93 KB
✓	C:\Users[illegible]r\AppData\Local\Microsoft\CLR_v4.0_32\UsageLogs	Automatically rebuilt by .NET	.NET	True	True	True	0	16	38.6 KB
✓	C:\Users[illegible]\AppData\Local\Microsoft\VisualStudio\16.0_de0dbad0\WebTools	Automatically rebuilt by .NET	.NET	True	True	True	9,640	16,270	1.09 GB
✓	C:\User[illegible]\AppData\Local\Microsoft\WebsiteCache	Automatically rebuilt by .NET	.NET	True	True	True	7	8	3.9 KB
✓	C:\Windows\Microsoft.NET\Framework\v4.0.30319\Temporary ASP.NET Files	Automatically rebuilt by .NET	.NET	True	True	True	0	0	0 Bytes
✓	C:\Users[illegible]\AppData\Local\Dropbox\Crashpad\reports		DropBox	False	True	True	0	2	8.95 MB
✓	C:\Users[illegible]\AppData\Local\Dropbox\logs		DropBox	False	True	True	5	1	10 MB
✓	C:\Users[illegible]r\AppData\Local\Google\CrashReports		Google	False	True	True	0	0	0 Bytes
✓	C:\Users[illegible]\Documents\IISExpress	Automatically rebuilt by IIS Express	IIS Express	True	True	True	3	3	76.46 KB
✓	C:\User[illegible]\Documents\IISExpress\Logs	Automatically rebuilt by IIS Express	IIS Express	True	True	True	0	0	0 Bytes
✓	C:\Users[illegible]\AppData\Local\TechSmith\Camtasia Studio\CrashDumps		TechSmith	False	True	True	0	0	0 Bytes
✓	C:\Users[illegible]\AppData\Local\Microsoft\VisualStudio\BackupFiles	Automatically rebuilt by Visual Studio	Visual Studio	True	True	True	15	0	0 Bytes
✓	C:\Users[illegible]\AppData\Local\Microsoft\VisualStudio\Roslyn\Cache	Automatically rebuilt by Visual Studio	Visual Studio	True	True	True	84	94	78.29 MB
✓	C:\Users[illegible]\AppData\Roaming\Code\Backups	Automatically rebuilt by VS Code	VS Code	True	True	True	0	1	107 Bytes
✓	C:\Users[illegible]\AppData\Roaming\Code\Cache	Automatically rebuilt by VS Code	VS Code	True	True	True	0	5	5.82 MB
✓	C:\Users[illegible]\AppData\Roaming\Code\CachedData	Automatically rebuilt by VS Code	VS Code	True	True	True	6	163	12.82 MB
✓	C:\Users[illegible]\AppData\Roaming\Code\GPUCache	Automatically rebuilt by VS Code	VS Code	True	True	True	0	5	544.36 KB
✓	C:\Users[illegible]\AppData\Roaming\Code\logs	Automatically rebuilt by VS Code	VS Code	True	True	True	8	22	7.66 KB
✓	C:\Users[illegible]\AppData\Roaming\Code\Session Storage	Automatically rebuilt by VS Code	VS Code	True	True	True	0	6	674 Bytes

```
********** END PREVIEW of Computer Cleaning on 2/11/2022 9:44:27 AM **********
```

Figure 3: A list of the specified folders to recycle

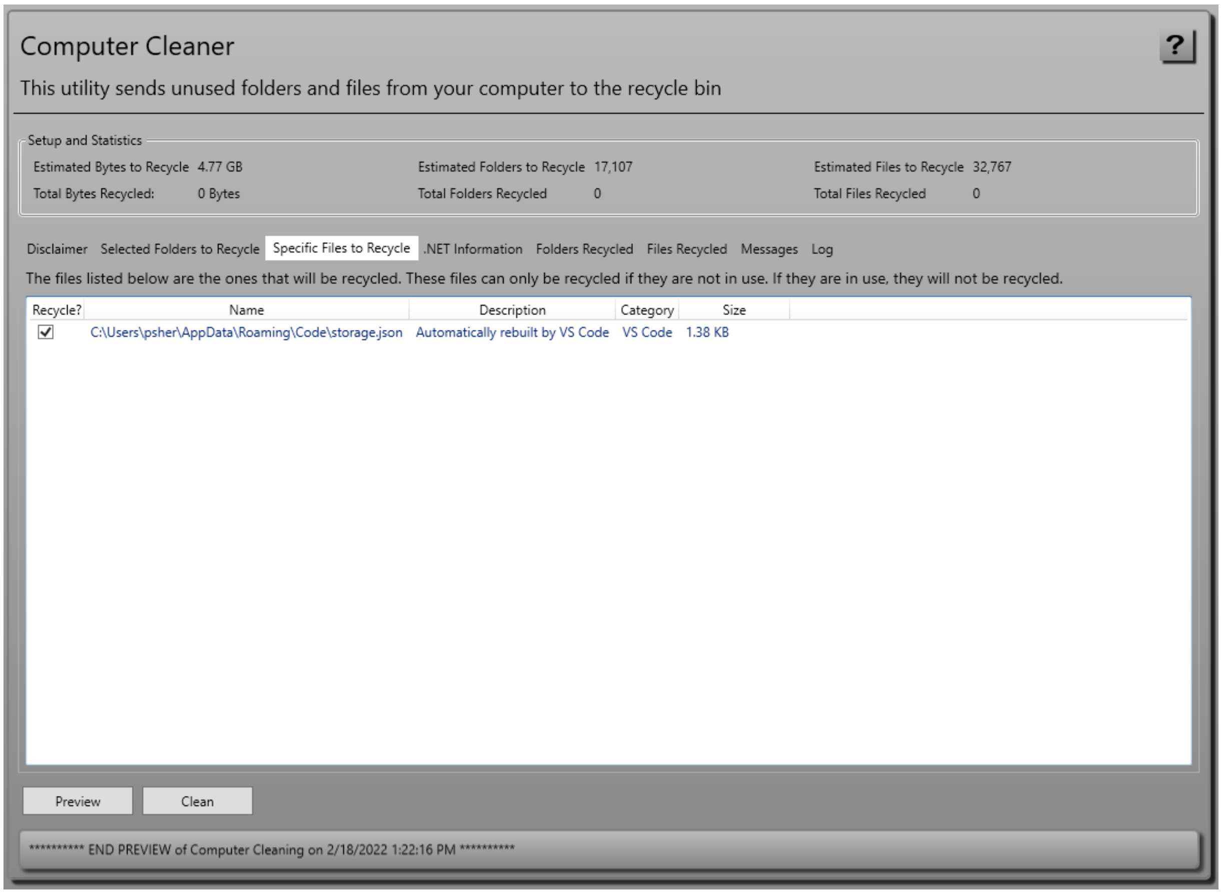
Column	Description
Recyle?	Check to recycle this folder and/or the files within this folder.
Path	The actual path to the folder/files to recycle
Description	A description of the folder.

Category	What type of files, or the application, that created the files in this folder.
Recyle this Folder?	If set to true, this folder and all subfolders and files within it will be deleted. If the Description field reads "Automatically rebuilt by …" then this folder is safe to have deleted.
Recycle SubFolders?	If set to true, then any subfolders within this folder will be deleted.
Recycle Files Only?	If set to true, then any files within this folder and subfolders will be deleted.
Total Folders	After clicking the Preview button, this column displays the total number of folders found within this folder.
Total Files	After clicking the Preview button, this column displays the total number of files found within this folder.
Total Size	After clicking the Preview button, this column displays the total number of bytes of all files/folder found within this folder.

NOTE: The count of folders and files, and the total bytes, is just an estimate of what could potentially be recycled. If the folder/file is in use, then it can't be recycled.

Selected Files to Recycle Tab

On this tab (Figure 4) is a list of specific files to recycle. The list of files that are going to be recycled is contained in the file **[MyDocuments]\PDSCDeveloperUtilities\Xml\ComputerCleaner-FilesToRecycle.xml**. You can add as many files to this XML file as you wish.


Computer Cleaner

This utility sends unused folders and files from your computer to the recycle bin

Setup and Statistics

Estimated Bytes to Recycle 4.77 GB
Estimated Folders to Recycle 17,107
Estimated Files to Recycle 32,767
Total Bytes Recycled: 0 Bytes
Total Folders Recycled 0
Total Files Recycled 0

The files listed below are the ones that will be recycled. These files can only be recycled if they are not in use. If they are in use, they will not be recycled.

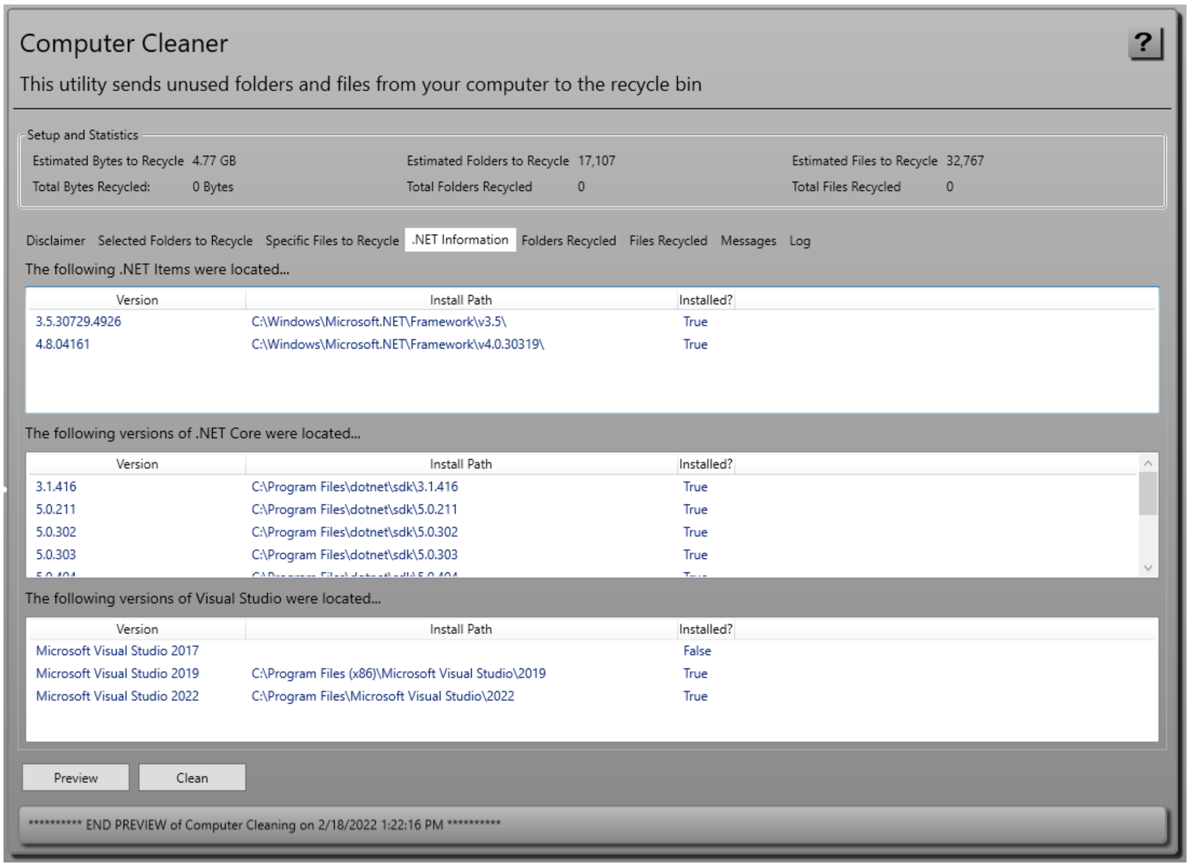
Recycle?	Name	Description	Category	Size
✓	C:\Users\psher\AppData\Roaming\Code\storage.json	Automatically rebuilt by VS Code	VS Code	1.38 KB

```
********** END PREVIEW of Computer Cleaning on 2/18/2022 1:22:16 PM **********
```

Figure 4: You can add additional files to recyle

.NET Information Tab

On this tab (Figure 5) is a list of .NET Framework and .NET Core versions located on your computer. The list of Visual Studio versions is also listed here.



Computer Cleaner

This utility sends unused folders and files from your computer to the recycle bin

Setup and Statistics

Estimated Bytes to Recycle	4.77 GB	Estimated Folders to Recycle	17,107	Estimated Files to Recycle	32,767
Total Bytes Recycled:	0 Bytes	Total Folders Recycled	0	Total Files Recycled	0

The following .NET Items were located...

Version	Install Path	Installed?
3.5.30729.4926	C:\Windows\Microsoft.NET\Framework\v3.5\	True
4.8.04161	C:\Windows\Microsoft.NET\Framework\v4.0.30319\	True

The following versions of .NET Core were located...

Version	Install Path	Installed?
3.1.416	C:\Program Files\dotnet\sdk\3.1.416	True
5.0.211	C:\Program Files\dotnet\sdk\5.0.211	True
5.0.302	C:\Program Files\dotnet\sdk\5.0.302	True
5.0.303	C:\Program Files\dotnet\sdk\5.0.303	True

The following versions of Visual Studio were located...

Version	Install Path	Installed?
Microsoft Visual Studio 2017		False
Microsoft Visual Studio 2019	C:\Program Files (x86)\Microsoft Visual Studio\2019	True
Microsoft Visual Studio 2022	C:\Program Files\Microsoft Visual Studio\2022	True

```
********** END PREVIEW of Computer Cleaning on 2/18/2022 1:22:16 PM **********
```

Figure 5: The list of .NET Frameworks, .NET Core and Visual Studio versions on your computer

Clean

Once you are satisfied with the list of folders and files to recycle from your hard-drive, click on the **Clean** button. This process can take several minutes depending on how many folders and files are on your hard drive. After this process is complete, the complete list of folders and files recycled is listed in the **Folders Recycled** and **Files Recycled** tabs.

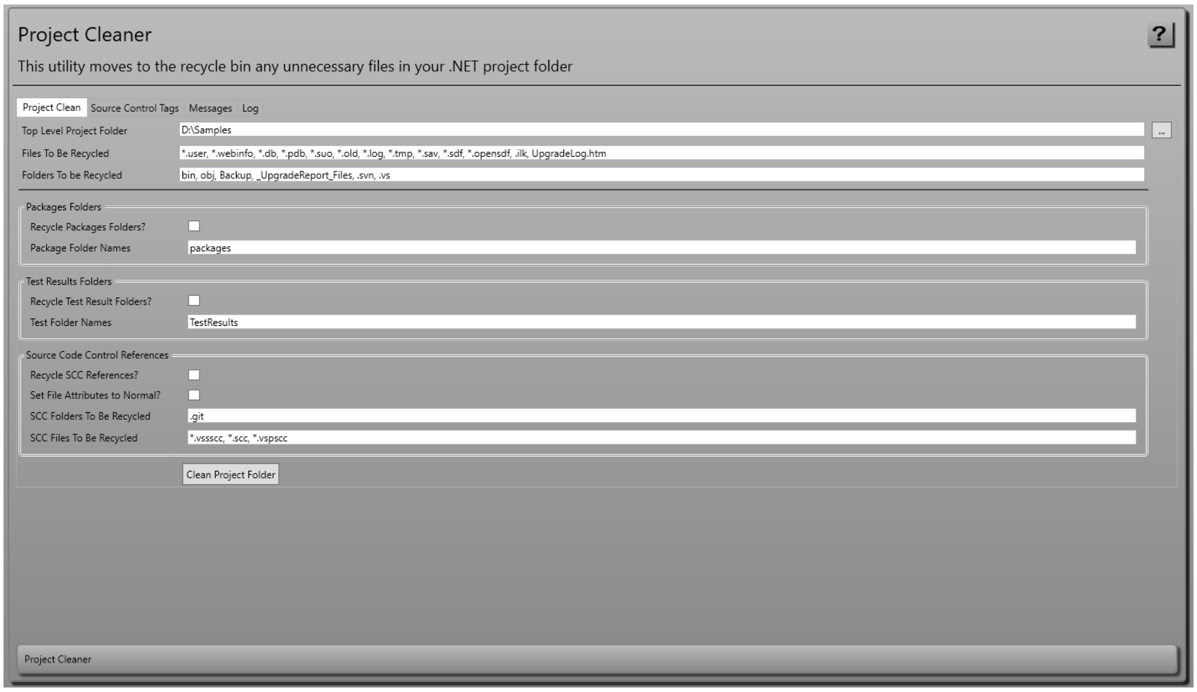
You can view the list of everything that happened on the **Messages** tab, and all these messages are written into a log file that is in your [**My Documents]\PDSCDeveloperUtilities\Log** folder.

It is perfectly normal to have some Error Messages display as well, as some folders/files may be in use and not able to be accessed. Or because of security constraints, they also may not be able to be accessed.

NOTE:	If after cleaning, something does not work correctly, go to your Recycle Bin and restore the folders/files that were recycled during this cleaning process.

Project Cleaner

When you create a project in Visual Studio, compile in different modes, and add the project to source control; a set of files and folders are created under your original project folder. Sometimes you might want to delete all these folders and files. For example, if you wish to give your project to someone else that is not on your network, does not have access to your source control, or you just want to clean up the folders under your project prior to adding your project for the first time to source control, you will want to eliminate all these extra files and folders using the Project Cleaner shown in Figure 6.



This utility moves to the recycle bin any unnecessary files in your .NET project folder

Top Level Project Folder	D:\Samples
Files To Be Recycled	*.user, *.webinfo, *.db, *.pdb, *.suo, *.old, *.log, *.tmp, *.sav, *.sdf, *.opensdf, .ilk, UpgradeLog.htm
Folders To be Recycled	bin, obj, Backup, _UpgradeReport_Files, .svn, .vs

Packages Folders

Recycle Packages Folders?	☐
Package Folder Names	packages

Test Results Folders

Recycle Test Result Folders?	☐
Test Folder Names	TestResults

Source Code Control References

Recycle SCC References?	☐
Set File Attributes to Normal?	☐
SCC Folders To Be Recycled	.git
SCC Files To Be Recycled	*.vssscc, *.scc, *.vspscc

Figure 6: Clean up files using the Project Cleaner utility

You will first enter a top-level folder and the Project Cleaner utility will iterate through all the lower-level folders and files underneath this folder and perform a series of operations. The operations performed will depend on what you fill in on the form in the following fields:

Field	Description
Top Level Project Folder	Enter the top level folder you wish to iterate through
Files to be Deleted	A list of file extensions that should be removed.
Folders to be Deleted	A list of folder names that should be deleted.
Remove Packages Folders?	Remove the “packages” folder.

Package Folder Names	Fill in the names of the packages folders to remove.
Remove Test Result Folders?	Check to remove any test result folders.
Test Folder Names	Fill in the names of the test result folders to remove.
Remove SCC References?	Check this is you wish this utility to remove the folders and files listed and to also open your .SLN and any .csproj or .vbproj files and remove the source control tags from these files.
Set File Attributes to Normal?	Check this to set the attribute of all files under the top-level Folder to normal.
SCC Folders to be Deleted	A list of source control folders that should be removed.
SCC Files to be Deleted	A list of source control file extensions that should be removed.

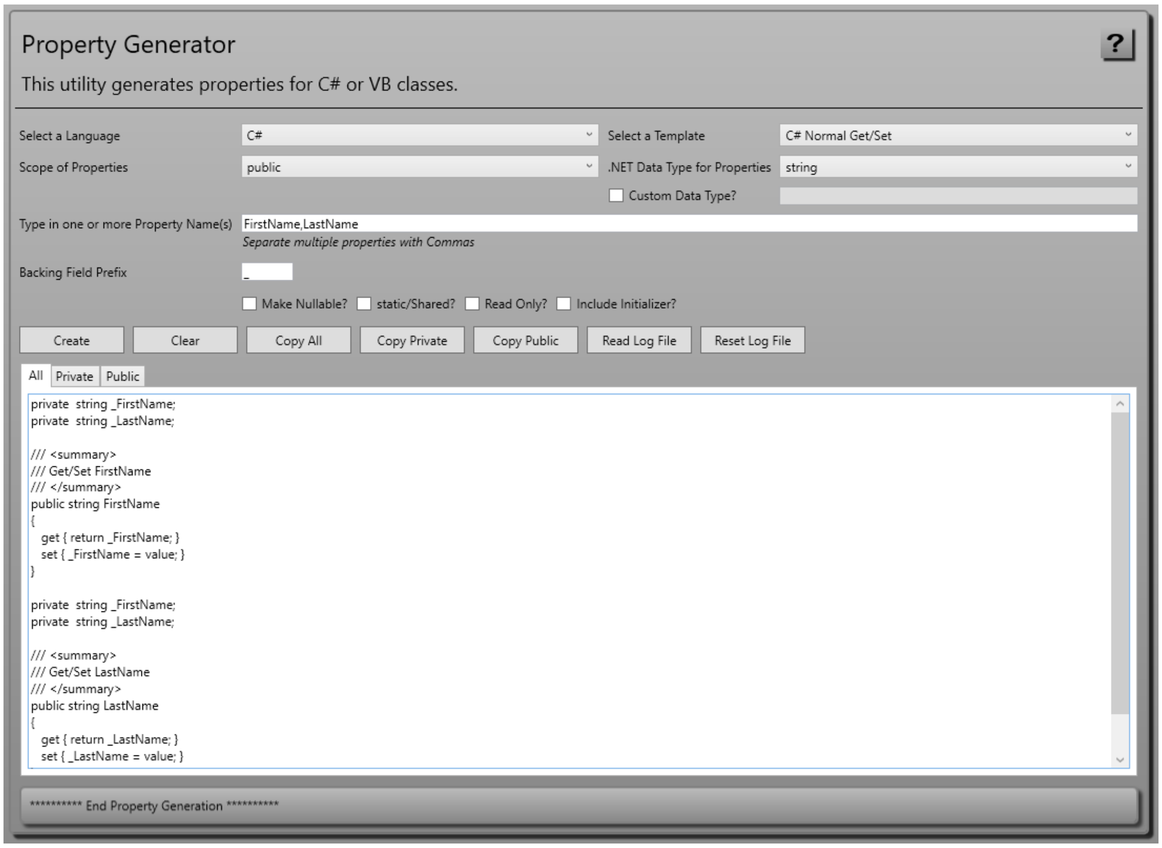
Table 2: Fields to fill in for cleaning projects.

NOTE: This utility only goes thru the folder and sub-folders specified in the **Top Level Folder** field. If the solution in the top-level folder points to another project in another folder structure, that other project will NOT have any of its attributes reset, or its source control references removed.

Property Generator

Visual Studio has code snippets that will let you create properties (Figure 7). These snippets such as **prop** and **propfull** are great for normal one-at-a-time properties. However, when you wish to create a lot of properties, or you need other types of properties, this is where the PDSC Property generator can help you out.

This tool will allow you to put in a comma-delimited list of property names, choose a scope and a data type and will then generate all the appropriate private variables and public property names in C# or Visual Basic. You will have a set of different templates to choose from that will allow you to create automatic properties, properties that raise the NotifyPropertyChanged event. You are also able to add your own templates to control how you generate the properties.


Property Generator

This utility generates properties for C# or VB classes.

Select a Language: C#
Select a Template: C# Normal Get/Set
Scope of Properties: public
.NET Data Type for Properties: string
Custom Data Type?
Type in one or more Property Name(s): FirstName,LastName
Separate multiple properties with Commas
Backing Field Prefix: _
Make Nullable? static/Shared? Read Only? Include Initializer?

```
private  string _FirstName;
private  string _LastName;

/// <summary>
/// Get/Set FirstName
/// </summary>
public string FirstName
{
  get { return _FirstName; }
  set { _FirstName = value; }
}

private  string _FirstName;
private  string _LastName;

/// <summary>
/// Get/Set LastName
/// </summary>
public string LastName
{
  get { return _LastName; }
  set { _LastName = value; }
```

********** End Property Generation **********

Figure 7: Property generator helps you create properties in many different styles

Adding Your Own Templates

Under the **[My Documents]\PDSCDeveloperUtilities\Xml** folder is a file named **PropertyGen-Templates.xml**. This file contains the list of template files you can use to generate properties. There is also a folder named **\Templates\PropertyGenerator** (Figure 8) in which there are several .txt files that hold the snippets for each of the types of properties that you can generate.

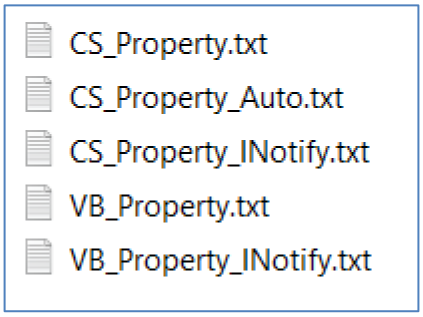

CS_Property.txt
CS_Property_Auto.txt
CS_Property_INotify.txt
VB_Property.txt
VB_Property_INotify.txt

Figure 8: All the property snippets are just contained in .txt files

You will find one XML node in the **PropertyGen-Templates.xml** file for each .txt file located in the **\Templates\PropertyGenerator** folder. To add a new template, you just copy one of the existing .txt files and give it a new name.

As an example, let's say you wanted to add a method call from every property "setter". You could copy the CS_Property.txt and call it CS_Test.txt. Open the CS_Test.txt in Notepad. It should look something like the following:

```
/// <summary>
/// Get/Set <|PROPERTY_NAME|>
/// </summary>
<|SCOPE|> <|STATIC|> <|LANGUAGE_DATA_TYPE|> <|PROPERTY_NAME|>
{
   get { return <|PRIVATE_FIELD_PREFIX|><|PROPERTY_NAME|>; }
   {|READ_ONLY|}set { <|PRIVATE_FIELD_PREFIX|><|PROPERTY_NAME|> =
value; }{/|READ_ONLY|}
}
```

You can now expand the "set" portion and add your own method call by changing this code to look something like the following:

```
/// <summary>
/// Get/Set <|PROPERTY_NAME|>
/// </summary>
<|SCOPE|> <|STATIC|> <|LANGUAGE_DATA_TYPE|> <|PROPERTY_NAME|>
{
   get { return <|PRIVATE_FIELD_PREFIX|><|PROPERTY_NAME|>; }
   {|READ_ONLY|}set
   {
     <|PRIVATE_FIELD_PREFIX|><|PROPERTY_NAME|> = value;
     MyMethod("<|PUBLICNAME|>");
   }{/|READ_ONLY|}
}
```

In the above template you broke up the "set" onto separate lines and then added a call to a method called MyMethod and you pass in as a string the public property name.

Next you need to add a new node to the **PropertyGen-Template.xml** file. Copy an existing node and paste it immediately after one of the descendant nodes. Modify the Description element to something you will recognize and the FileName element to the name of your new .txt file.

```
<PropertyTemplate>
  <Description>C# My Method Get/Set</Description>
  <FileName>CS_Test.txt</FileName>
  <Language>CSharp</Language>
  <GenPrivateVars>True</GenPrivateVars>
  <GenPublic>True</GenPublic>
</PropertyTemplate>
```

Now, restart the PDSC Developer Utilities and your new template for the property generator will appear.

Property Generator Tokens

In the .txt files that represent the code to generate for the properties you find a set of tokens in the format <|TOKEN_NAME|>. There are a few tokens that are recognized by our property generator that are different from the Code Generator tokens. Table 3 contains the list of the tokens that you can use in your templates.

Token	Description
{|READ_ONLY|} and {|/READ_ONLY|}	Wrap these tokens around your "set" property to remove the "set" if you choose "Read only" in the Property Genreator tool.
<|READ_ONLY|>	Used for Visual Basic to insert "ReadOnly" into a property name
<|SCOPE|>	Returns the scope specified in the property generator tool.
<|STATIC|>	Returns "static"
<|SHARED|>	Returns "Shared"
<|NO_PRIVATE_VARIABLES|>	Do not generate the private variables

Table 3. List of Tokens in Property generator

Other XML files for the Property Generator

There are a few other XML files that the property generator uses to assist with the generation. These files are located in the **[My Documents]\PDSCDeveloperUtilities\Xml** folder under the location where you installed the Developer Utilities.

Xml File name	Description
LanguageDataTypes	A list of data types for C# and Visual Basic.
PropertyGen-DotNetLanguages	The list of .NET languages.
PropertyGen-LanguageScope	A list of scopes for C# and Visual Basic.
PropertyGen-Templates.xml	The list of templates that can be generated

Table 4. List of XML files for the Property Generator

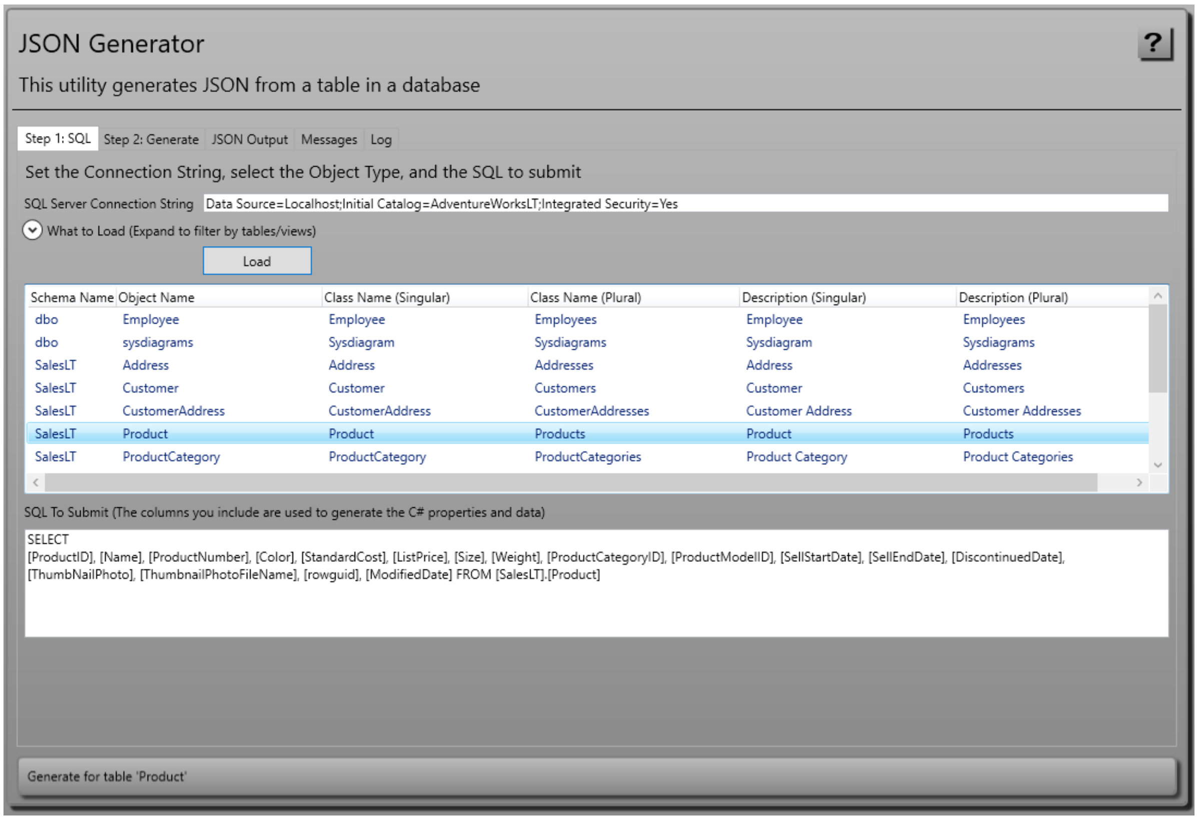
JSON Generator

JSON files are very handy for a lot of things. The PDSC JSON Generator utility builds a JSON file from the data within any table or view in your SQL Server database.

Step 1: SQL / Select Object to Generate

To start the JSON generation process, put in the appropriate connection string that will connect you to your database (Figure 9). You can optionally expand the "What to Load…" area and select the type of objects to load (Tables or Views). If you have a large collection of objects in your database, you may wish to fill in a Schema name (or partial schema name), and/or an object name (or partial object name) prior to clicking on the Load button.

Click on the Load button to load all objects in the database. Click on one of the objects in the list and the appropriate SQL statement will be generated in the text box below the object list. You can modify this SQL prior to moving to step 3 if you wish to generate different names for your element or attribute names.


JSON Generator
This utility generates JSON from a table in a database
?
Step 1: SQL
Step 2: Generate
JSON Output
Messages
Log
What to Load (Expand to filter by tables/views)
Load
Generate for table 'Product'

Set the Connection String, select the Object Type, and the SQL to submit

SQL Server Connection String: Data Source=Localhost;Initial Catalog=AdventureWorksLT;Integrated Security=Yes

Schema Name	Object Name	Class Name (Singular)	Class Name (Plural)	Description (Singular)	Description (Plural)
dbo	Employee	Employee	Employees	Employee	Employees
dbo	sysdiagrams	Sysdiagram	Sysdiagrams	Sysdiagram	Sysdiagrams
SalesLT	Address	Address	Addresses	Address	Addresses
SalesLT	Customer	Customer	Customers	Customer	Customers
SalesLT	CustomerAddress	CustomerAddress	CustomerAddresses	Customer Address	Customer Addresses
SalesLT	Product	Product	Products	Product	Products
SalesLT	ProductCategory	ProductCategory	ProductCategories	Product Category	Product Categories

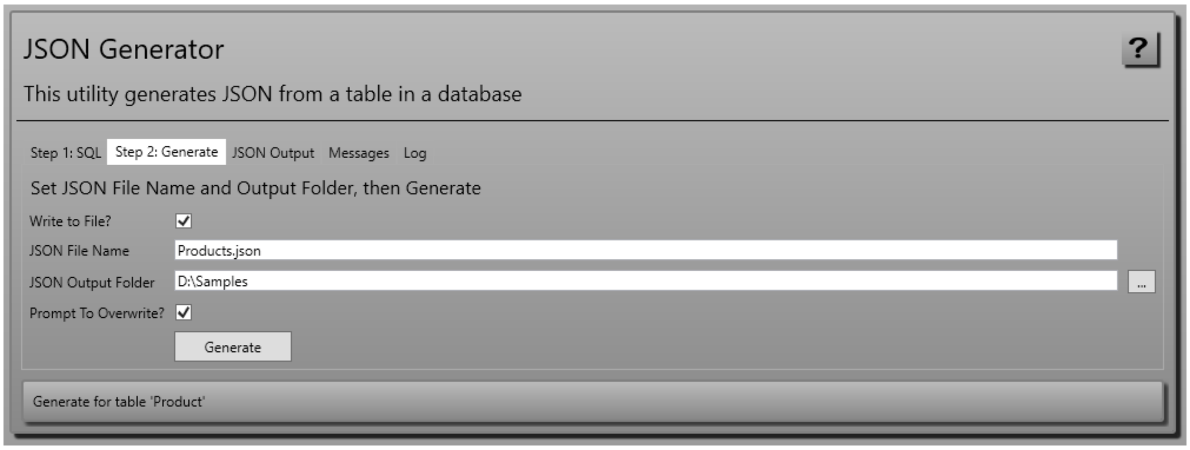
SQL To Submit (The columns you include are used to generate the C# properties and data)

```
SELECT
[ProductID], [Name], [ProductNumber], [Color], [StandardCost], [ListPrice], [Size], [Weight], [ProductCategoryID], [ProductModelID], [SellStartDate], [SellEndDate], [DiscontinuedDate],
[ThumbNailPhoto], [ThumbnailPhotoFileName], [rowguid], [ModifiedDate] FROM [SalesLT].[Product]
```

Figure 9: Step 1: JSON Generator SQL Tab

Step 2: Generate

Click on **Step 2: Generate** (Figure 10) to fill in information on how you wish to generate the JSON. You can either write to a file or not. If you write to a file, specify the name of the file and the folder for the JSON file. A ".json" file extension will automatically be added to the file name. You will be prompted to overwrite this file if you check the **Prompt to Overwrite?** check box.



JSON Generator

This utility generates JSON from a table in a database

Set JSON File Name and Output Folder, then Generate

Write to File? ✓

JSON File Name: Products.json

JSON Output Folder: D:\Samples

Prompt To Overwrite? ✓

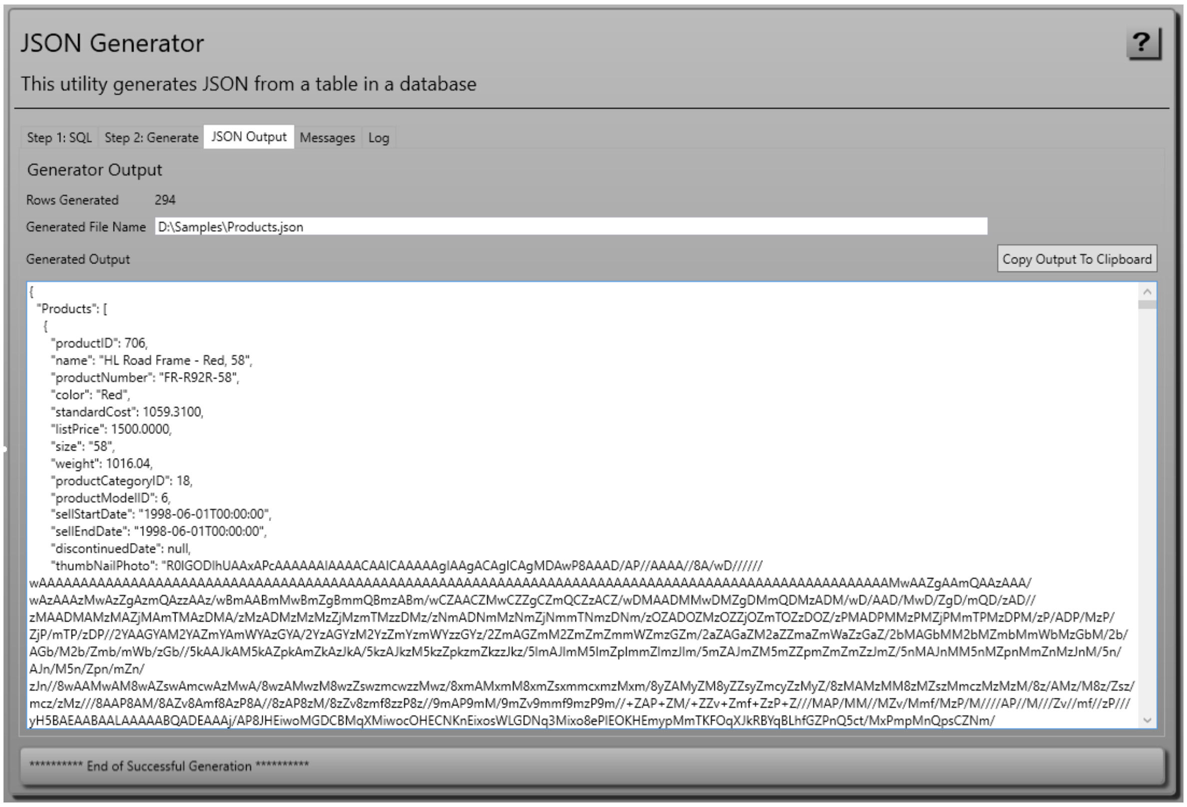
Generate for table 'Product'

Figure 10: Step 2: JSON Generator Generate Tab

Click the **Generate** button to start the generation process.

View the JSON Output

After you click on the **Generate** (Figure 11) button, you are presented with the screen shown in Figure 11. This screen shows you where the generated .json file is located and the JSON output.



Rows Generated 294

Generated File Name D:\Samples\Products.json

Generated Output

```
{
  "Products": [
    {
      "productID": 706,
      "name": "HL Road Frame - Red, 58",
      "productNumber": "FR-R92R-58",
      "color": "Red",
      "standardCost": 1059.3100,
      "listPrice": 1500.0000,
      "size": "58",
      "weight": 1016.04,
      "productCategoryID": 18,
      "productModelID": 6,
      "sellStartDate": "1998-06-01T00:00:00",
      "sellEndDate": "1998-06-01T00:00:00",
      "discontinuedDate": null,
      "thumbNailPhoto": "R0lGODlhUAAxAPcAAAAAAIAAAACAAICAAAAAgIAAgACAgICAgMDAwP8AAAD/AP//AAAA//8A/wD//////
wAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMwAAZgAAmQAAzAAA/
wAzAAAzMwAzZgAzmQAzzAAz/wBmAABmMwBmZgBmmQBmzABm/wCZAACZMwCZZgCZmQCZzACZ/wDMAADMMwDMZgDMmQDMzADM/wD/AAD/MwD/ZgD/mQD/zAD//
zMAADMAMzMAZjMAmTMAzDMA/zMzADMzMzMzZjMzmTMzzDMz/zNmADNmMzNmZjNmmTNmzDNm/zOZADOZMzOZZjOZmTOZzDOZ/zPMADPMMzPMZjPMmTPMzDPM/zP/ADP/MzP/
ZjP/mTP/zDP//2YAAGYAM2YAZmYAmWYAzGYA/2YzAGYzM2YzZmYzmWYzzGYz/2ZmAGZmM2ZmZmZmmWZmzGZm/2aZAGaZM2aZZmaZmWaZzGaZ/2bMAGbMM2bMZmbMmWbMzGbM/2b/
AGb/M2b/Zmb/mWb/zGb//5kAAJkAM5kAZpkAmZkAzJkA/5kzAJkzM5kzZpkzmZkzzJkz/5lmAJlmM5lmZplmmZlmzJlm/5mZAJmZM5mZZpmZmZmZzJmZ/5nMAJnMM5nMZpnMmZnMzJnM/5n/
AJn/M5n/Zpn/mZn/
zJn//8wAAMwAM8wAZswAmcwAzMwA/8wzAMwzM8wzZswzmcwzzMwz/8xmAMxmM8xmZsxmmcxmzMxm/8yZAMyZM8yZZsyZmcyZzMyZ/8zMAMzMM8zMZszMmczMzMzM/8z/AMz/M8z/Zsz/
mcz/zMz///8AAP8AM/8AZv8Amf8AzP8A//8zAP8zM/8zZv8zmf8zzP8z//9mAP9mM/9mZv9mmf9mzP9m//+ZAP+ZM/+ZZv+Zmf+ZzP+Z///MAP/MM//MZv/Mmf/MzP/M////AP//M///Zv//mf//zP///
yH5BAEAABAALAAAAABQADEAAAj/AP8JHEiwoMGDCBMqXMiwocOHECNKnEixosWLGDNq3Mixo8ePIEOKHEmypMmTKFOqXJkRBYqBLhfGZPnQ5ct/MxPmpMnQpsCZNm/
```

********** End of Successful Generation **********

Figure 11: JSON Generator Output tab

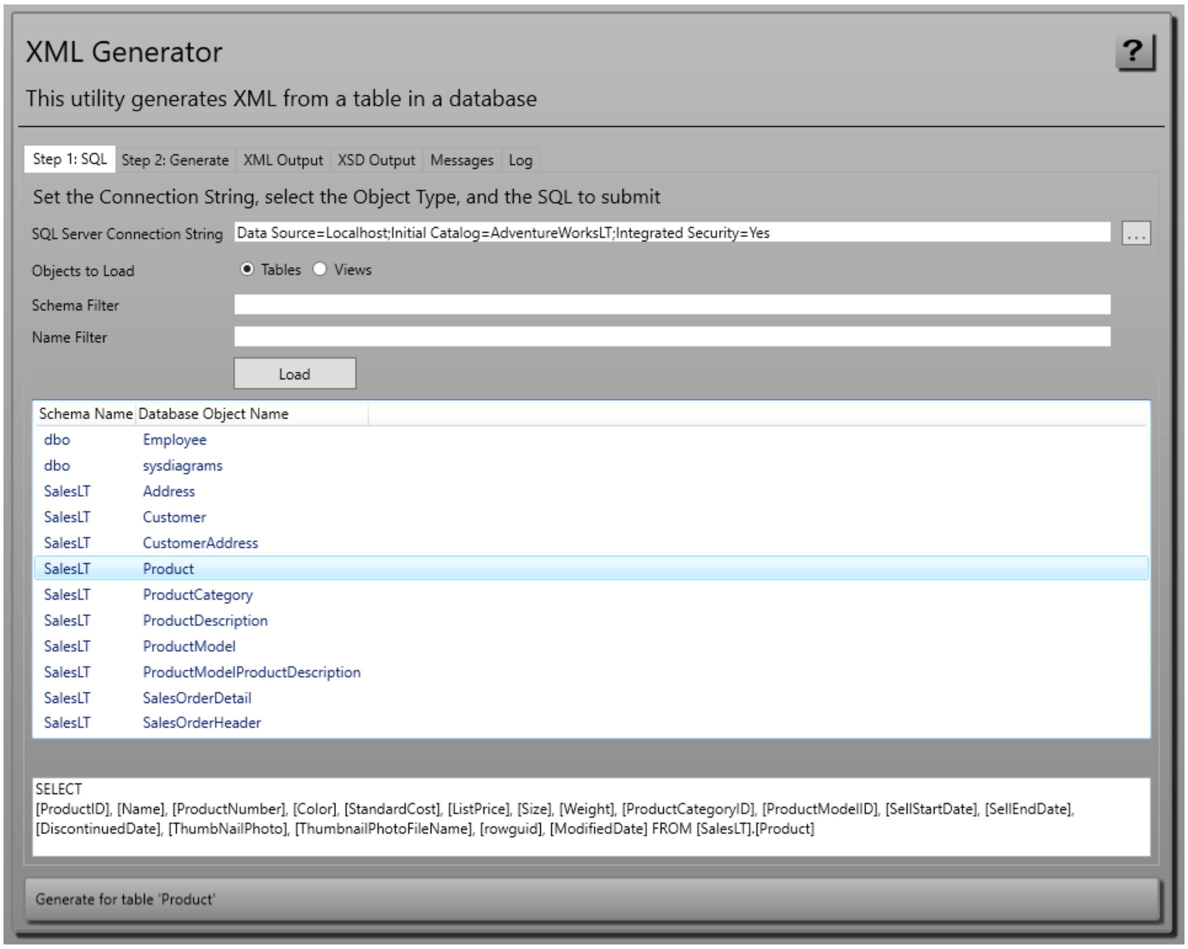
XML Generator

XML files are very handy for a lot of things. The PDSC XML Generator utility builds XML and XSD files from any table or view in your SQL Server database.

Step 1: SQL / Select Object to Generate

To start the XML generation process, put in the appropriate connection string that will connect you to your database (Figure 12). You can optionally expand the "What to Load…" area and select the type of objects to load (Tables or Views). If you have a large collection of objects in your database, you may wish to fill in a Schema name (or partial schema name), and/or an object name (or partial object name) prior to clicking on the Load button.

Click on the Load button to load all objects in the database. Click on one of the objects in the list and the appropriate SQL statement will be generated in the text box below the object list. You can modify this SQL prior to moving to step 3 if you wish to generate different names for your element or attribute names.


XML Generator
?
This utility generates XML from a table in a database
Step 1: SQL
Step 2: Generate
XML Output
XSD Output
Messages
Log
Set the Connection String, select the Object Type, and the SQL to submit
SQL Server Connection String
...
Objects to Load
Tables
Views
Schema Filter
Name Filter
Load
Generate for table 'Product'

SQL Server Connection String: Data Source=Localhost;Initial Catalog=AdventureWorksLT;Integrated Security=Yes

Objects to Load: [✓ Tables] Views

Schema Name	Database Object Name
dbo	Employee
dbo	sysdiagrams
SalesLT	Address
SalesLT	Customer
SalesLT	CustomerAddress
SalesLT	Product
SalesLT	ProductCategory
SalesLT	ProductDescription
SalesLT	ProductModel
SalesLT	ProductModelProductDescription
SalesLT	SalesOrderDetail
SalesLT	SalesOrderHeader

```
SELECT
[ProductID], [Name], [ProductNumber], [Color], [StandardCost], [ListPrice], [Size], [Weight], [ProductCategoryID], [ProductModelID], [SellStartDate], [SellEndDate],
[DiscontinuedDate], [ThumbNailPhoto], [ThumbnailPhotoFileName], [rowguid], [ModifiedDate] FROM [SalesLT].[Product]
```

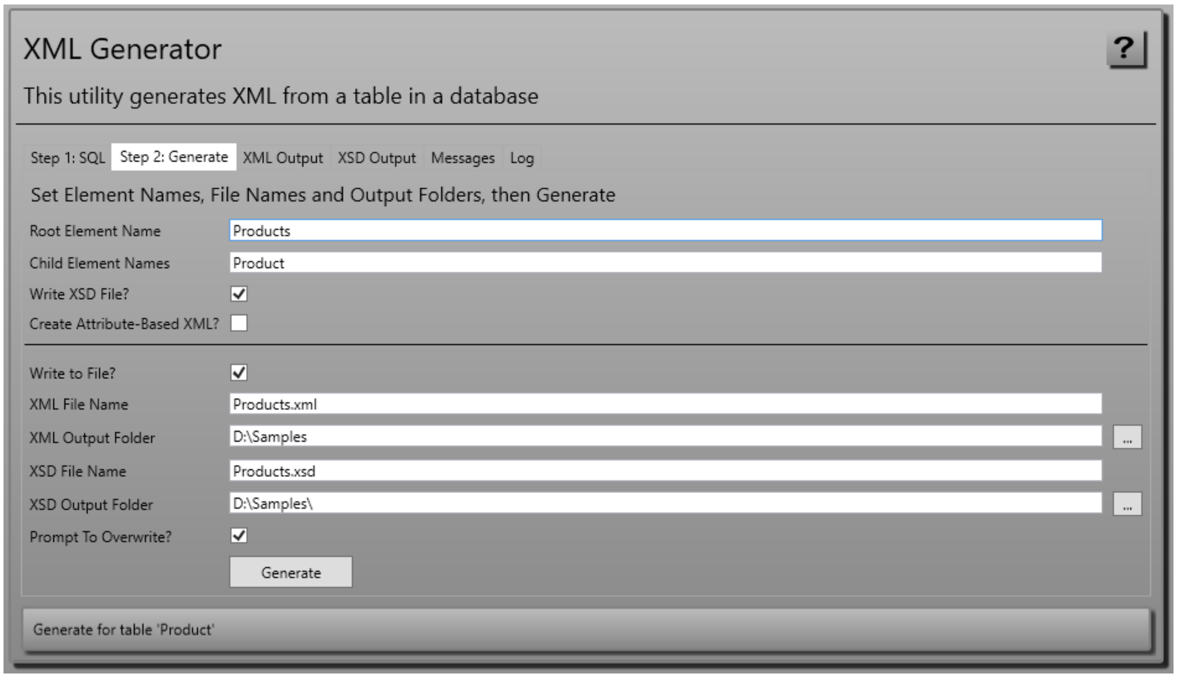
Figure 12: Step 1: XML Generator SQL Tab

Step 2: Generate

Click on **Step 2: Generate** (Figure 13) to fill in information on how you wish to generate the XML/XSD files. You can specify your Root Element Name, and for each row the Child Element Name to use. Check the **Write XSD File?** to generate an XSD file. Check the **Create Attribute-Based XML?** to generate attribute-based XML file.

You can either write to a file or not. Fill in the name of the XML file name and XML Output folder. Fill in the XSD file name and XSD output folder. You will be prompted to overwrite this file if you check the **Prompt To Overwrite?** check box.

Click the **Generate(s)** button to start the generation process.



XML Generator

This utility generates XML from a table in a database

Set Element Names, File Names and Output Folders, then Generate

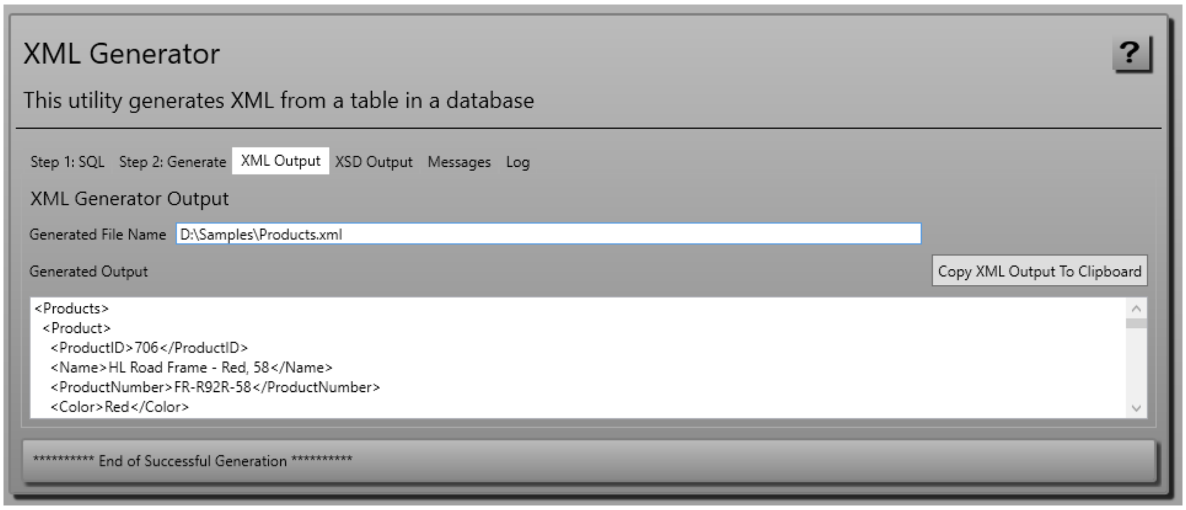
Root Element Name: Products
Child Element Names: Product
Write XSD File? ✓
Create Attribute-Based XML?
Write to File? ✓
XML File Name: Products.xml
XML Output Folder: D:\Samples
XSD File Name: Products.xsd
XSD Output Folder: D:\Samples\
Prompt To Overwrite? ✓

Generate for table 'Product'

Figure 13: Step 2: XML Generator Generate Tab

XML Output

After you click on the **Generates** button, you will be presented with the screen shown in Figure 14.



XML Generator

This utility generates XML from a table in a database

XML Generator Output

Generated File Name: D:\Samples\Products.xml

Generated Output

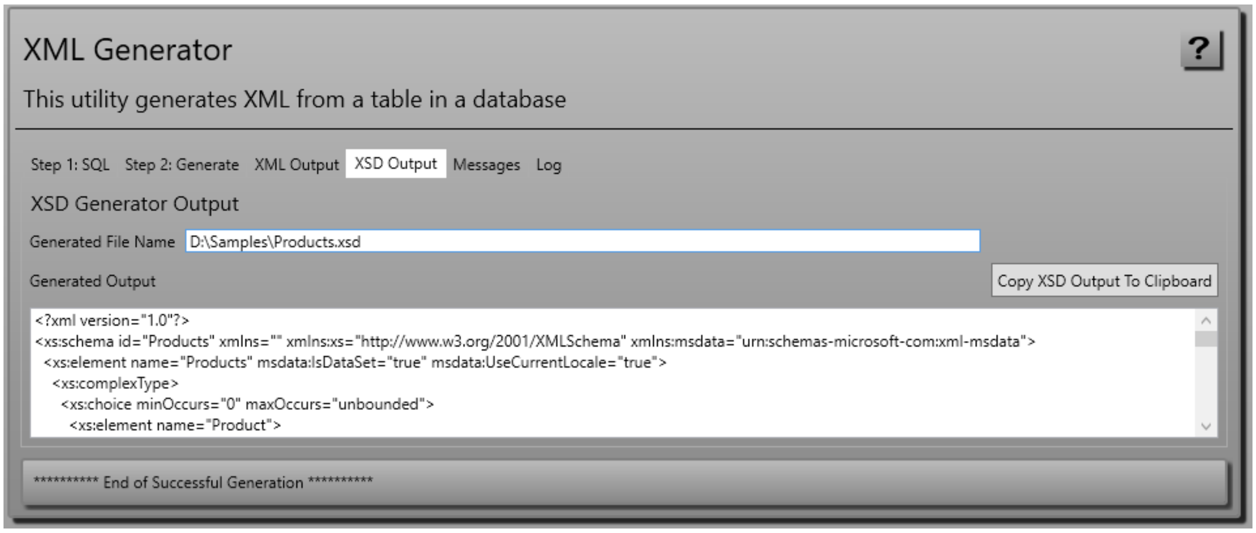
```
<Products>
 <Product>
  <ProductID>706</ProductID>
  <Name>HL Road Frame - Red, 58</Name>
  <ProductNumber>FR-R92R-58</ProductNumber>
  <Color>Red</Color>
```

********** End of Successful Generation **********

Figure 14: XML Output Tab

XSD Output

If you generated an XSD, you can view the XSD on the screen show in Figure 15.



Generated File Name: D:\Samples\Products.xsd

```
<?xml version="1.0"?>
<xs:schema id="Products" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
  <xs:element name="Products" msdata:IsDataSet="true" msdata:UseCurrentLocale="true">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Product">
```

```
********** End of Successful Generation **********
```

Figure 15: XSD Output tab

C# Data Repository Generator

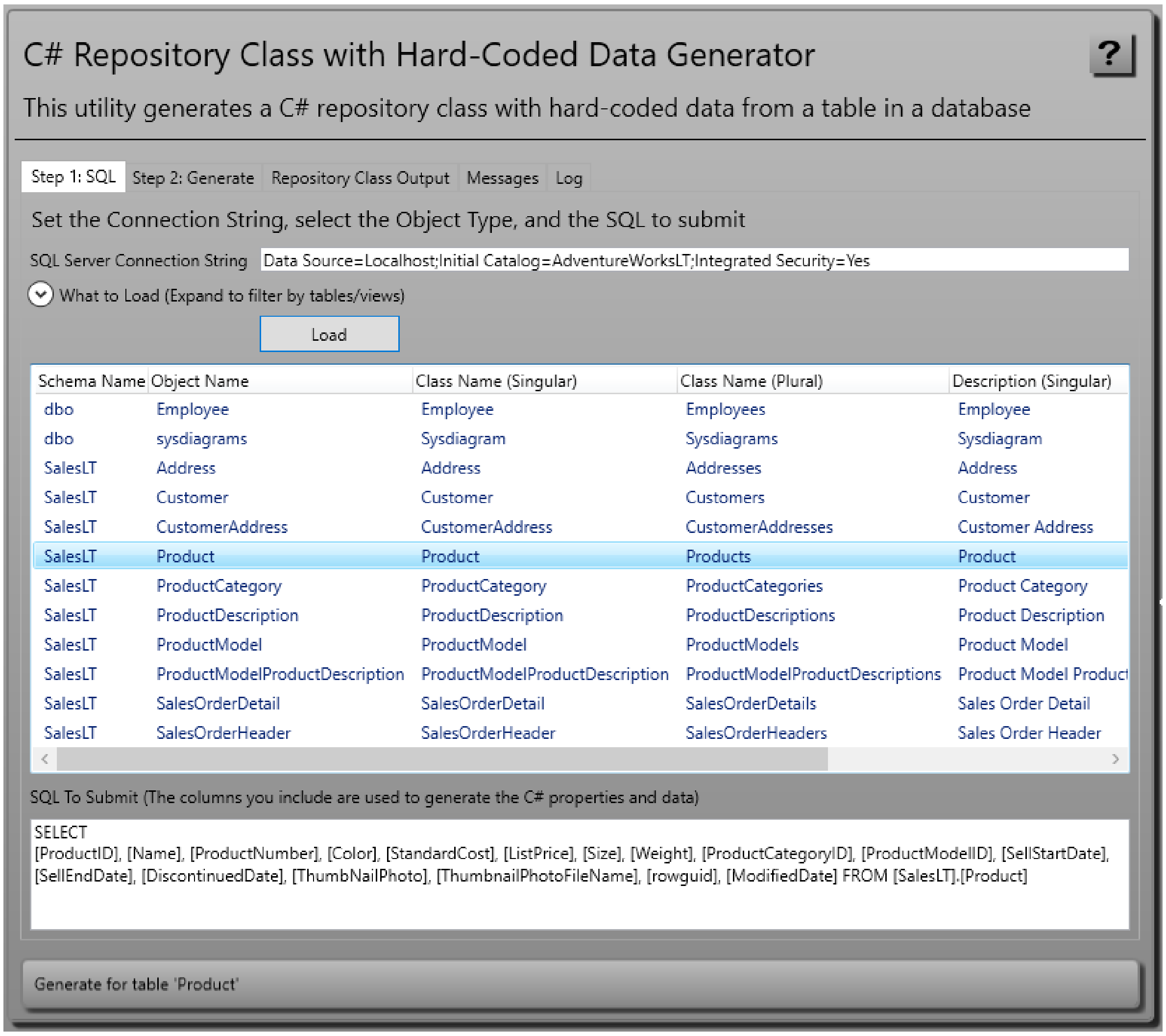
A repository class is one that has methods to return data from a data source. When creating exercises for a training class, or to test some functionality, but you don't want to have to connect to a database, it is nice to have a collection of hard-coded data that can be returned. Instead of you having to create all this data by hand, if you have a table with data, you can use this generator to select some data and have it hard-coded into a repository class.

Step 1: SQL and Object Selection

To start the C# repository class generation process, put in the appropriate connection string that will connect you to your database (Figure 16). Choose whether you wish to load Tables or Views by selecting the appropriate radio button. If you have a large collection of objects in your database, you may wish to fill in a Schema Filter (partial schema name), and/or a Name Filter (partial object name) prior to clicking on the Load button.

After clicking on the Load button, you will be presented with a list of database objects that match your specific filter. Click on one of the objects in the list and the appropriate SQL statement will be generated in the text box below the object list. You can modify this SQL prior to moving to step 3 if you wish to generate different names for your element or attribute names.

If you type in your own SQL, add you may add on "*SELECT **TOP 100***" or some other number if you don't want to generate all the records in the table/view.


C# Repository Class with Hard-Coded Data Generator

This utility generates a C# repository class with hard-coded data from a table in a database

Set the Connection String, select the Object Type, and the SQL to submit

SQL Server Connection String: Data Source=Localhost;Initial Catalog=AdventureWorksLT;Integrated Security=Yes

What to Load (Expand to filter by tables/views)

Schema Name	Object Name	Class Name (Singular)	Class Name (Plural)	Description (Singular)
dbo	Employee	Employee	Employees	Employee
dbo	sysdiagrams	Sysdiagram	Sysdiagrams	Sysdiagram
SalesLT	Address	Address	Addresses	Address
SalesLT	Customer	Customer	Customers	Customer
SalesLT	CustomerAddress	CustomerAddress	CustomerAddresses	Customer Address
SalesLT	Product	Product	Products	Product
SalesLT	ProductCategory	ProductCategory	ProductCategories	Product Category
SalesLT	ProductDescription	ProductDescription	ProductDescriptions	Product Description
SalesLT	ProductModel	ProductModel	ProductModels	Product Model
SalesLT	ProductModelProductDescription	ProductModelProductDescription	ProductModelProductDescriptions	Product Model Product
SalesLT	SalesOrderDetail	SalesOrderDetail	SalesOrderDetails	Sales Order Detail
SalesLT	SalesOrderHeader	SalesOrderHeader	SalesOrderHeaders	Sales Order Header

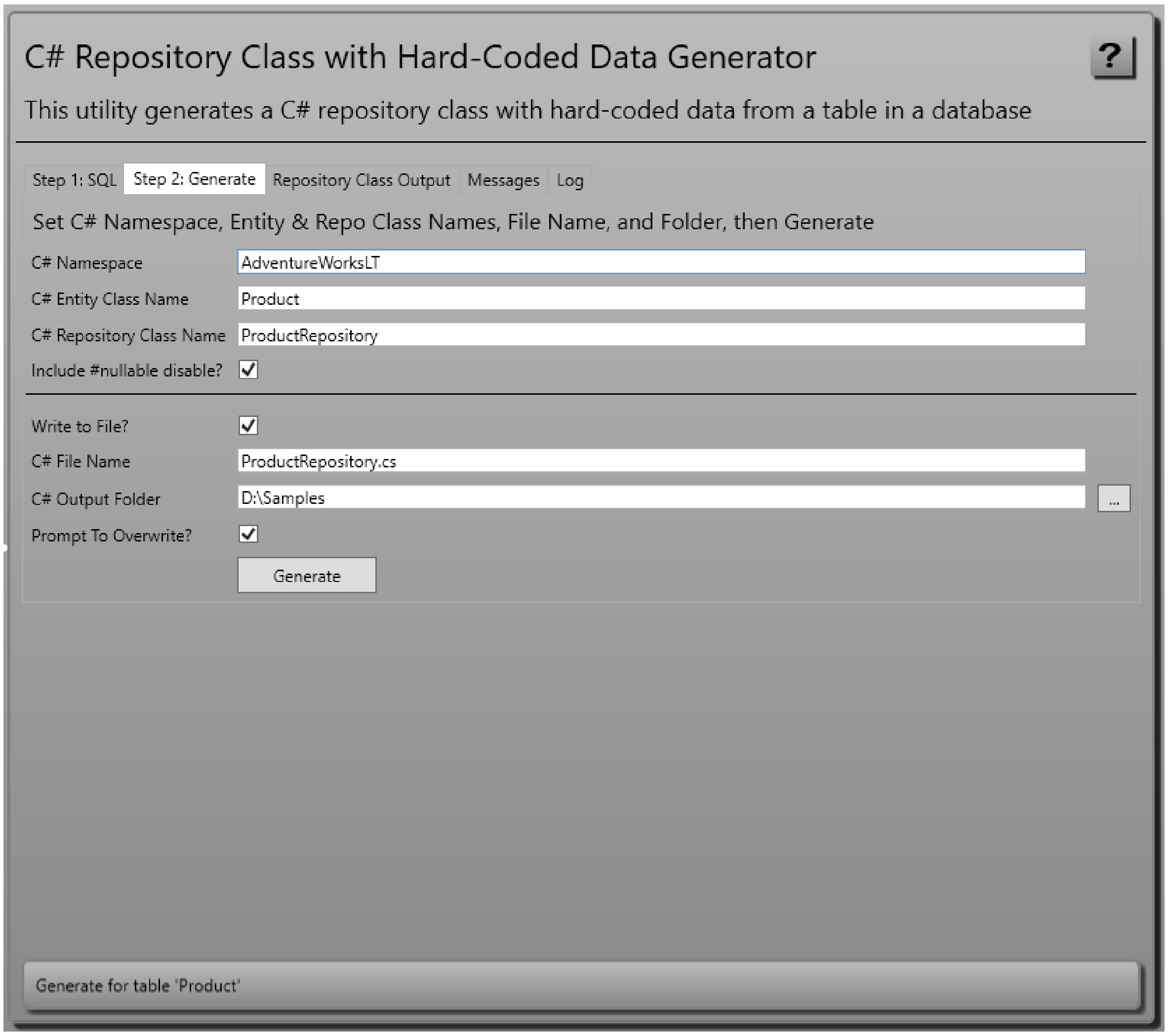
SQL To Submit (The columns you include are used to generate the C# properties and data)

```
SELECT
[ProductID], [Name], [ProductNumber], [Color], [StandardCost], [ListPrice], [Size], [Weight], [ProductCategoryID], [ProductModelID], [SellStartDate],
[SellEndDate], [DiscontinuedDate], [ThumbNailPhoto], [ThumbnailPhotoFileName], [rowguid], [ModifiedDate] FROM [SalesLT].[Product]
```

Figure 16: Step 1: C# Repository Class Generator SQL Tab

Step 2: Generate

Click on **Step 3: Generate** (Figure 17) to fill in information on how you wish to generate the C# repository class. Fill in the Namespace to use, the Entity Class Name, the Repository Class Name and if you wish to include the #nullable disable statement at the top of the file. You can either write to a file or not. Fill in the file name. and the output folder. Check the **Prompt to Overwrite?** check box if you want to be prompted before overwriting a previously written file. Click the **Generate** button to start the generation process.


C# Repository Class with Hard-Coded Data Generator

This utility generates a C# repository class with hard-coded data from a table in a database

Set C# Namespace, Entity & Repo Class Names, File Name, and Folder, then Generate

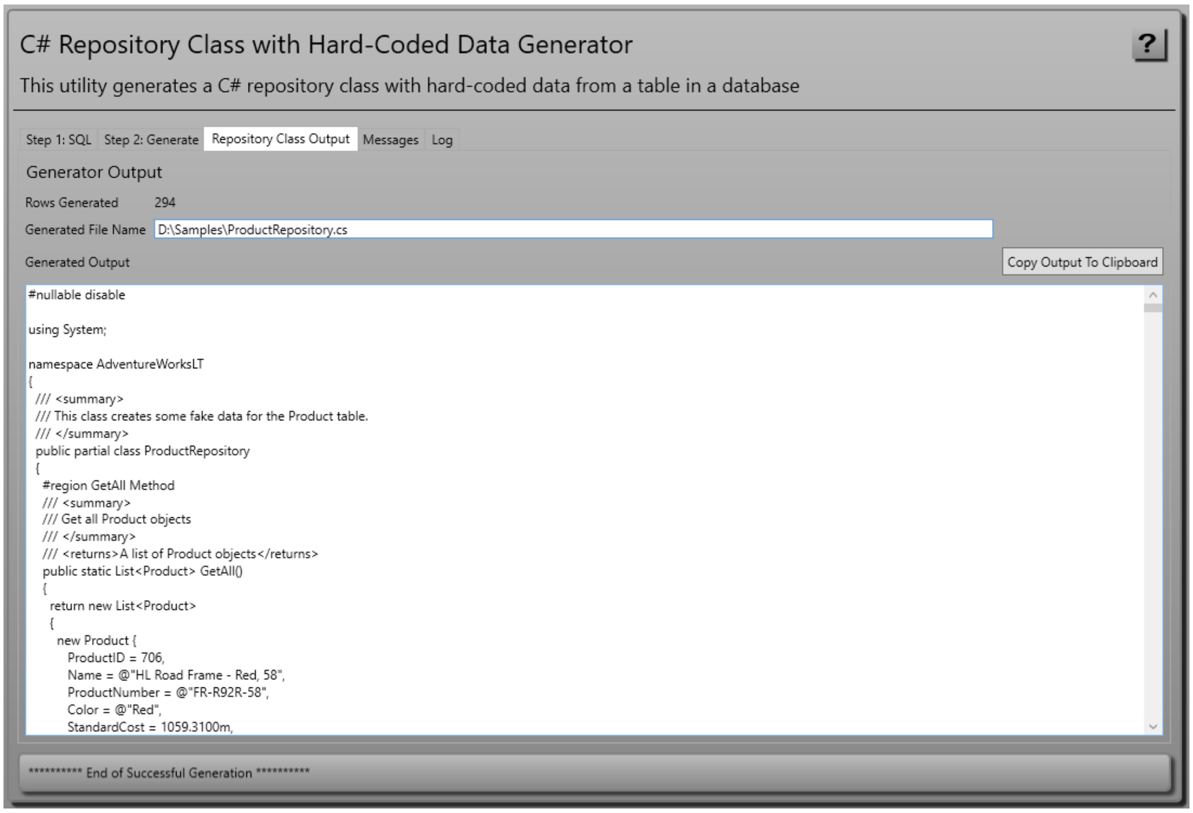
C# Namespace	AdventureWorksLT
C# Entity Class Name	Product
C# Repository Class Name	ProductRepository
Include #nullable disable?	✓
Write to File?	✓
C# File Name	ProductRepository.cs
C# Output Folder	D:\Samples
Prompt To Overwrite?	✓

Generate for table 'Product'

Figure 17: Step 2: C# Repository Class Generator Generate Tab

Step 3: Output

After you click on the **Generate** button, you will be presented with the screen shown in Figure 18. This screen tells you where the C# repository file is located and allows you to copy the entity class to the clipboard.


C# Repository Class with Hard-Coded Data Generator
This utility generates a C# repository class with hard-coded data from a table in a database
?
Step 1: SQL
Step 2: Generate
Repository Class Output
Messages
Log
Generator Output
Generated Output
Copy Output To Clipboard

Rows Generated 294

Generated File Name D:\Samples\ProductRepository.cs

```
#nullable disable

using System;

namespace AdventureWorksLT
{
  /// <summary>
  /// This class creates some fake data for the Product table.
  /// </summary>
  public partial class ProductRepository
  {
    #region GetAll Method
    /// <summary>
    /// Get all Product objects
    /// </summary>
    /// <returns>A list of Product objects</returns>
    public static List<Product> GetAll()
    {
      return new List<Product>
      {
        new Product {
          ProductID = 706,
          Name = @"HL Road Frame - Red, 58",
          ProductNumber = @"FR-R92R-58",
          Color = @"Red",
          StandardCost = 1059.3100m,
```

********** End of Successful Generation **********

Figure 18: Step 3: C# Repository Class Generator Output tab

Data Repository Generator Tokens

In the .txt files that represent the code to generate for the Hard Coded Data Repository class you find a few tokens in the format <|TOKEN_NAME|>. There are a few tokens that are recognized by our data repository generator that are different from the Code Generator tokens. Table 5 contains the list of the tokens that you can use in your templates.

Token	Description
<|HARD_CODED_DATA|>	Returns the generated values from all rows and columns and inserts them the generated output
<|UPDATE_COLUMNS|>	Returns the columns to update
<|CLASS_NAME_REPOSITORY|>	Returns the name of the repository class as specified in the Repository Generator tool

Table 5. List of Tokens in Data Repository generator

Code Generator

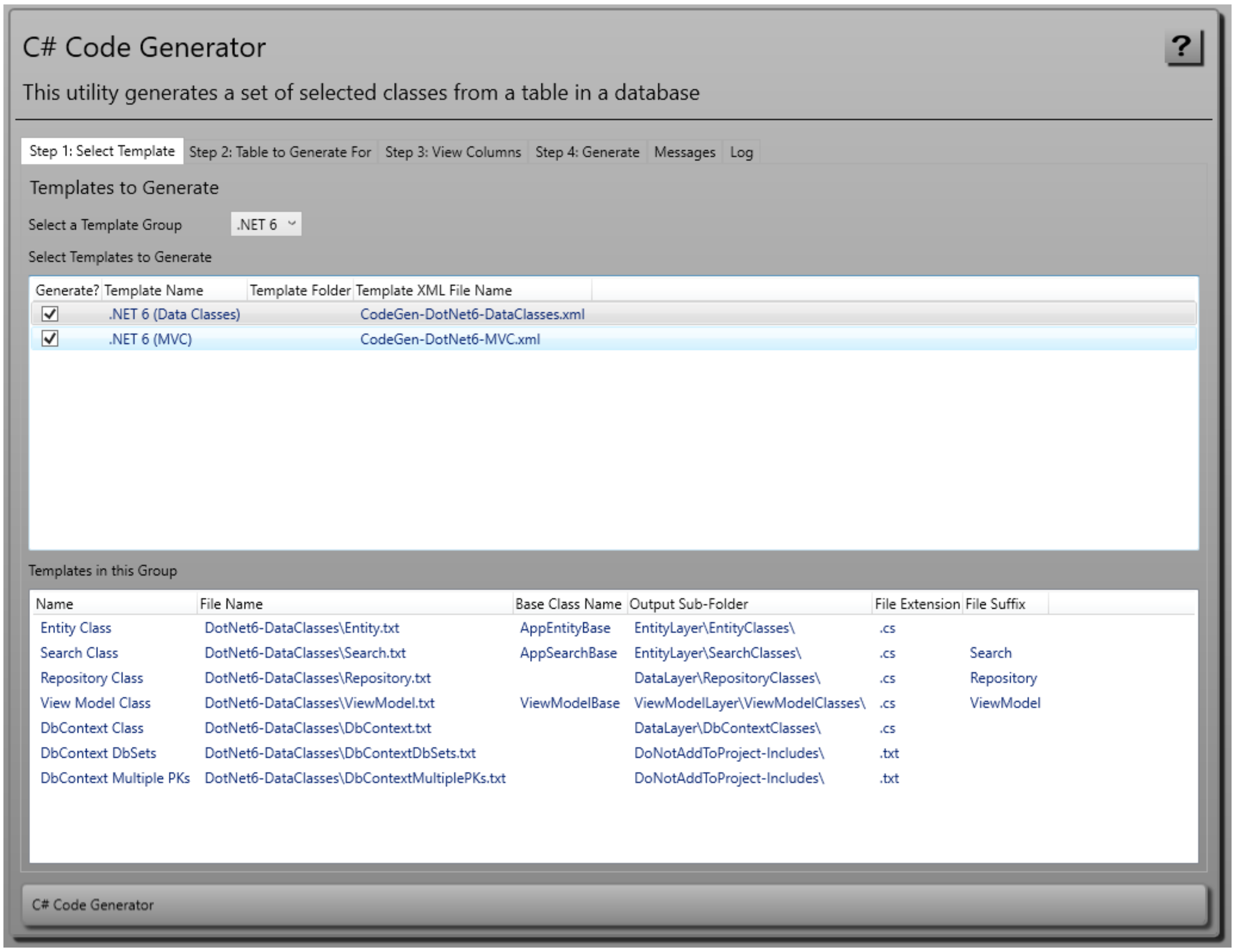
The PDSC Code Generator generates a set of classes and MVC pages for a single table in your SQL Server database. For example, with the supplied templates, you can generate Entity, Repository, Search and View Model classes. You can also generate a complete CRUD MVC set of pages. The following table explains what you can generate with the Code Generator

Template	Description
.NET 6 (Data Classes)	Generates the appropriate classes to get all, get a single, search for, add, edit, and delete data from a single table. The code uses the Entity Framework for all this functionality. It generates a DbContext class to which you can add additional generated code for more tables. A view model class is created for use with MVC applications.
.NET 6 (MVC)	Generates an MVC controller class, an index page, and partials pages for listing, searching, adding, editing, and deleting data from a table. A Program.cs file and an appsettings.Development.json file are also generated to connect to the DbContext and repository classes generated from the template ".NET 6 (Data Classes)".
.NET Framework 4.x (Data Classes)	Generates the appropriate classes to get all, get a single, search for, add, edit, and delete data from a single table. The code uses the Entity Framework for all this functionality. It generates a DbContext class to which you can add additional generated code for more tables. A view model class is created for use with WPF or MVC applications.
.NET Framework 4.x (MVC)	Generates an MVC controller class, an index page, and partials pages for listing, searching, adding, editing, and deleting data from a table. A Web.config file is generated to store the connection string that is supplied to the DbContext class generated from the template ".NET Framework 4.x (Data Classes)".

Table 6. List of Standard Template in the Code Generator

Step 1: Select Template(s)

On the first tab (Figure 19) you select which version of .NET you wish to generate from the drop-down list box. Next, select one or more of the Templates to generate.



C# Code Generator

This utility generates a set of selected classes from a table in a database

Templates to Generate

Select a Template Group: .NET 6

Select Templates to Generate

Generate?	Template Name	Template Folder	Template XML File Name
✓	.NET 6 (Data Classes)		CodeGen-DotNet6-DataClasses.xml
✓	.NET 6 (MVC)		CodeGen-DotNet6-MVC.xml

Templates in this Group

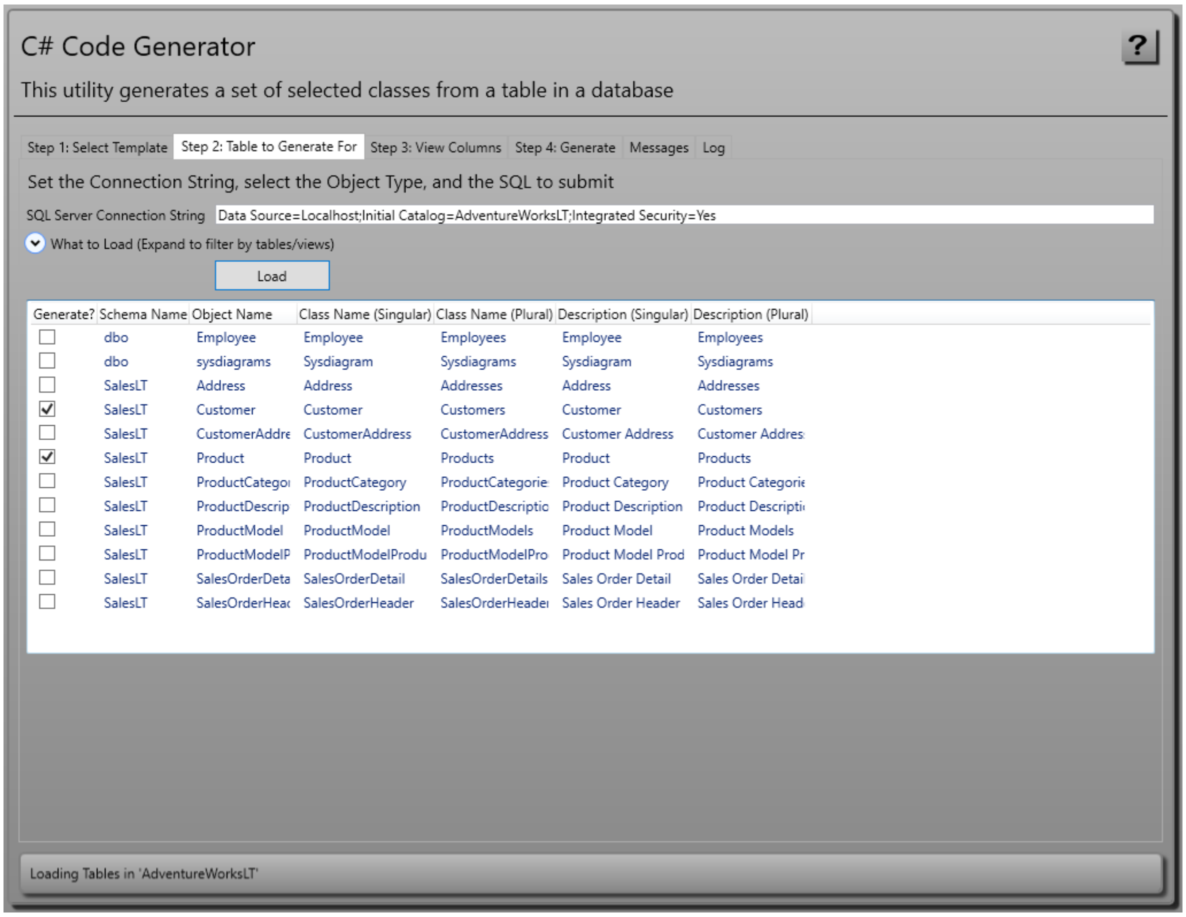
Name	File Name	Base Class Name	Output Sub-Folder	File Extension	File Suffix
Entity Class	DotNet6-DataClasses\Entity.txt	AppEntityBase	EntityLayer\EntityClasses\	.cs	
Search Class	DotNet6-DataClasses\Search.txt	AppSearchBase	EntityLayer\SearchClasses\	.cs	Search
Repository Class	DotNet6-DataClasses\Repository.txt		DataLayer\RepositoryClasses\	.cs	Repository
View Model Class	DotNet6-DataClasses\ViewModel.txt	ViewModelBase	ViewModelLayer\ViewModelClasses\	.cs	ViewModel
DbContext Class	DotNet6-DataClasses\DbContext.txt		DataLayer\DbContextClasses\	.cs	
DbContext DbSets	DotNet6-DataClasses\DbContextDbSets.txt		DoNotAddToProject-Includes\	.txt	
DbContext Multiple PKs	DotNet6-DataClasses\DbContextMultiplePKs.txt		DoNotAddToProject-Includes\	.txt	

Figure 19: Step 1: C# Code Generator Select Template Tab

Step 2: Table(s) to Generate For

Put in the appropriate connection string that will connect you to your database (Figure 20). You can optionally expand the "What to Load…" area and select the type of objects to load (Tables or Views). If you have a large collection of objects in your database, you may wish to fill in a Schema name (or partial schema name), and/or an object name (or partial object name) prior to clicking on the Load button. Click on the Load button to load all objects in the database.

Click on the check box in the **Generate?** Column next to the table for which you wish to generate code. If you click on one of the tables (not on the check box), you can then view the columns and the meta-data about that specific table in the **Step 3: View Columns** tab.



C# Code Generator

This utility generates a set of selected classes from a table in a database

Step 1: Select Template | Step 2: Table to Generate For | Step 3: View Columns | Step 4: Generate | Messages | Log

Set the Connection String, select the Object Type, and the SQL to submit

SQL Server Connection String: Data Source=Localhost;Initial Catalog=AdventureWorksLT;Integrated Security=Yes

What to Load (Expand to filter by tables/views)

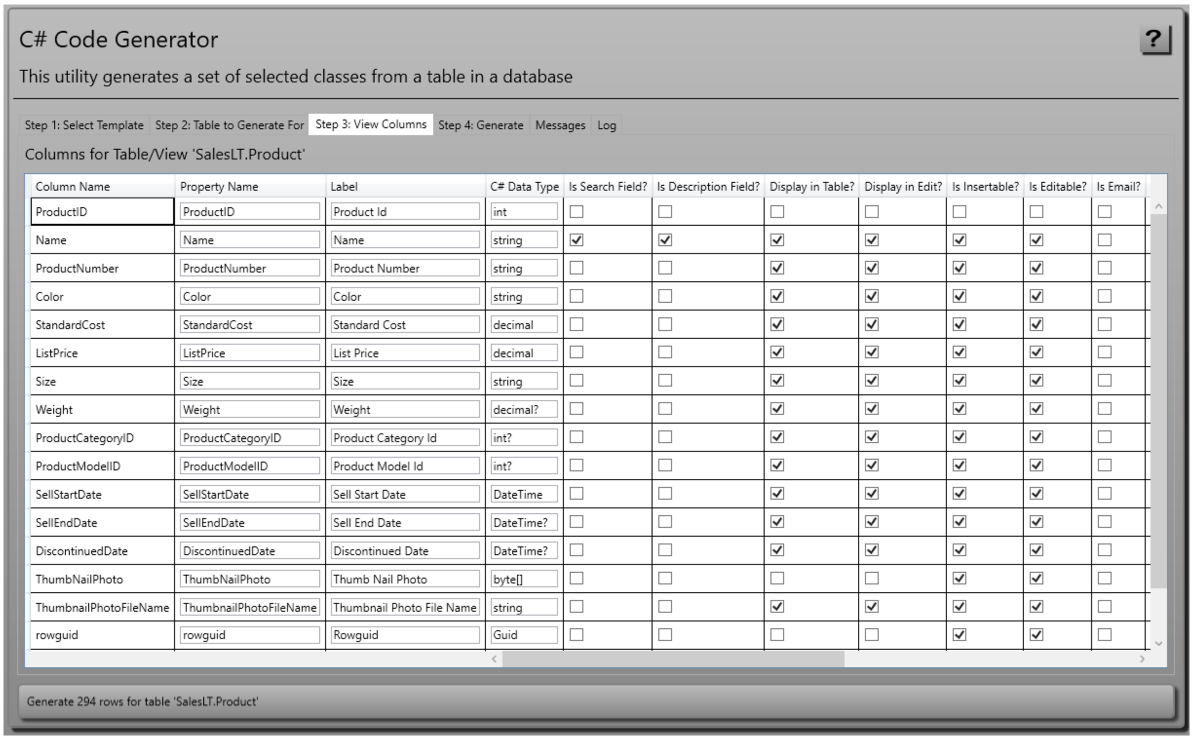
Generate?	Schema Name	Object Name	Class Name (Singular)	Class Name (Plural)	Description (Singular)	Description (Plural)
☐	dbo	Employee	Employee	Employees	Employee	Employees
☐	dbo	sysdiagrams	Sysdiagram	Sysdiagrams	Sysdiagram	Sysdiagrams
☐	SalesLT	Address	Address	Addresses	Address	Addresses
✓	SalesLT	Customer	Customer	Customers	Customer	Customers
☐	SalesLT	CustomerAddre	CustomerAddress	CustomerAddress	Customer Address	Customer Addres
✓	SalesLT	Product	Product	Products	Product	Products
☐	SalesLT	ProductCatego	ProductCategory	ProductCategorie	Product Category	Product Categorie
☐	SalesLT	ProductDescrip	ProductDescription	ProductDescriptio	Product Description	Product Descripti
☐	SalesLT	ProductModel	ProductModel	ProductModels	Product Model	Product Models
☐	SalesLT	ProductModelP	ProductModelProdu	ProductModelPro	Product Model Prod	Product Model Pr
☐	SalesLT	SalesOrderDeta	SalesOrderDetail	SalesOrderDetails	Sales Order Detail	Sales Order Detai
☐	SalesLT	SalesOrderHea	SalesOrderHeader	SalesOrderHeade	Sales Order Header	Sales Order Head

Loading Tables in 'AdventureWorksLT'

Figure 20: Step 2: C# Code Generator Table to Generate For Tab

Step 3: View Columns

On this tab (Figure 21) you can view all the columns for a selected table on the previous tab. If you have not clicked on a table, this list will be blank. For the selected table you may check or uncheck the appropriate usage of each of the columns. You may also modify the Property Name, Label and C# data type. **NOTE**: If you change any of these values, they are not saved after the generation occurs.



C# Code Generator

This utility generates a set of selected classes from a table in a database

Columns for Table/View 'SalesLT.Product'

Column Name	Property Name	Label	C# Data Type	Is Search Field?	Is Description Field?	Display in Table?	Display in Edit?	Is Insertable?	Is Editable?	Is Email?
ProductID	ProductID	Product Id	int	☐	☐	☐	☐	☐	☐	☐
Name	Name	Name	string	✓	✓	✓	✓	✓	✓	☐
ProductNumber	ProductNumber	Product Number	string	☐	☐	✓	✓	✓	✓	☐
Color	Color	Color	string	☐	☐	✓	✓	✓	✓	☐
StandardCost	StandardCost	Standard Cost	decimal	☐	☐	✓	✓	✓	✓	☐
ListPrice	ListPrice	List Price	decimal	☐	☐	✓	✓	✓	✓	☐
Size	Size	Size	string	☐	☐	✓	✓	✓	✓	☐
Weight	Weight	Weight	decimal?	☐	☐	✓	✓	✓	✓	☐
ProductCategoryID	ProductCategoryID	Product Category Id	int?	☐	☐	✓	✓	✓	✓	☐
ProductModelID	ProductModelID	Product Model Id	int?	☐	☐	✓	✓	✓	✓	☐
SellStartDate	SellStartDate	Sell Start Date	DateTime	☐	☐	✓	✓	✓	✓	☐
SellEndDate	SellEndDate	Sell End Date	DateTime?	☐	☐	✓	✓	✓	✓	☐
DiscontinuedDate	DiscontinuedDate	Discontinued Date	DateTime?	☐	☐	✓	✓	✓	✓	☐
ThumbNailPhoto	ThumbNailPhoto	Thumb Nail Photo	byte[]	☐	☐	☐	☐	✓	✓	☐
ThumbnailPhotoFileName	ThumbnailPhotoFileName	Thumbnail Photo File Name	string	☐	☐	✓	✓	✓	✓	☐
rowguid	rowguid	Rowguid	Guid	☐	☐	☐	☐	✓	✓	☐

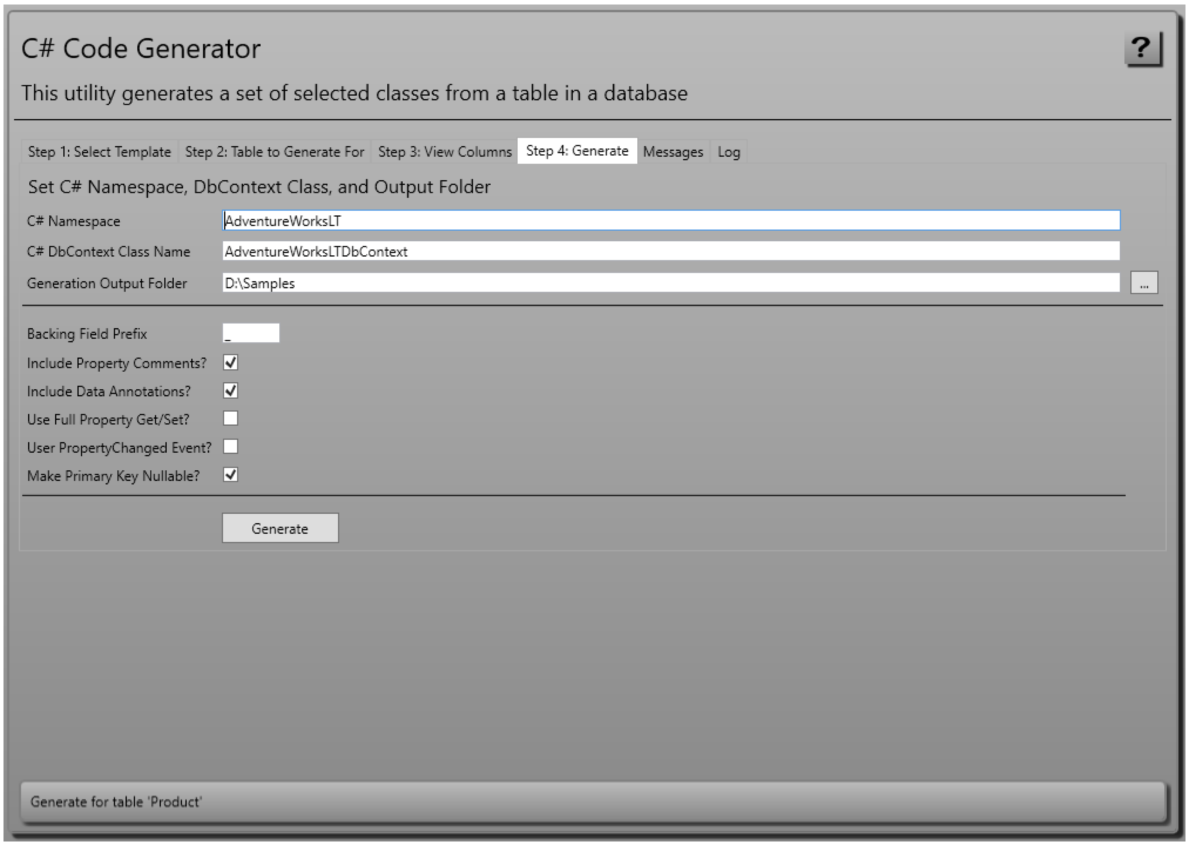
Generate 294 rows for table 'SalesLT.Product'

Figure 21: Step 3: C# Code Generator View Columns Tab

Step 4: Generate

Click on **Step 4: Generate** (Figure 22) to fill in information on how you wish to generate the C# classes. Fill in the Namespace to use, the entity class name, and if you wish to generate Data Annotations for each property and if you wish to include the #nullable disable statement at the top of the file.

You can either write to a file or not. Fill in the C# File Name, the C# Output Folder and check the **Prompt to Overwrite?** check box if you want to be prompted before overwriting a previously written file. Click the **Generate** button to start the generation process.



This utility generates a set of selected classes from a table in a database

Set C# Namespace, DbContext Class, and Output Folder

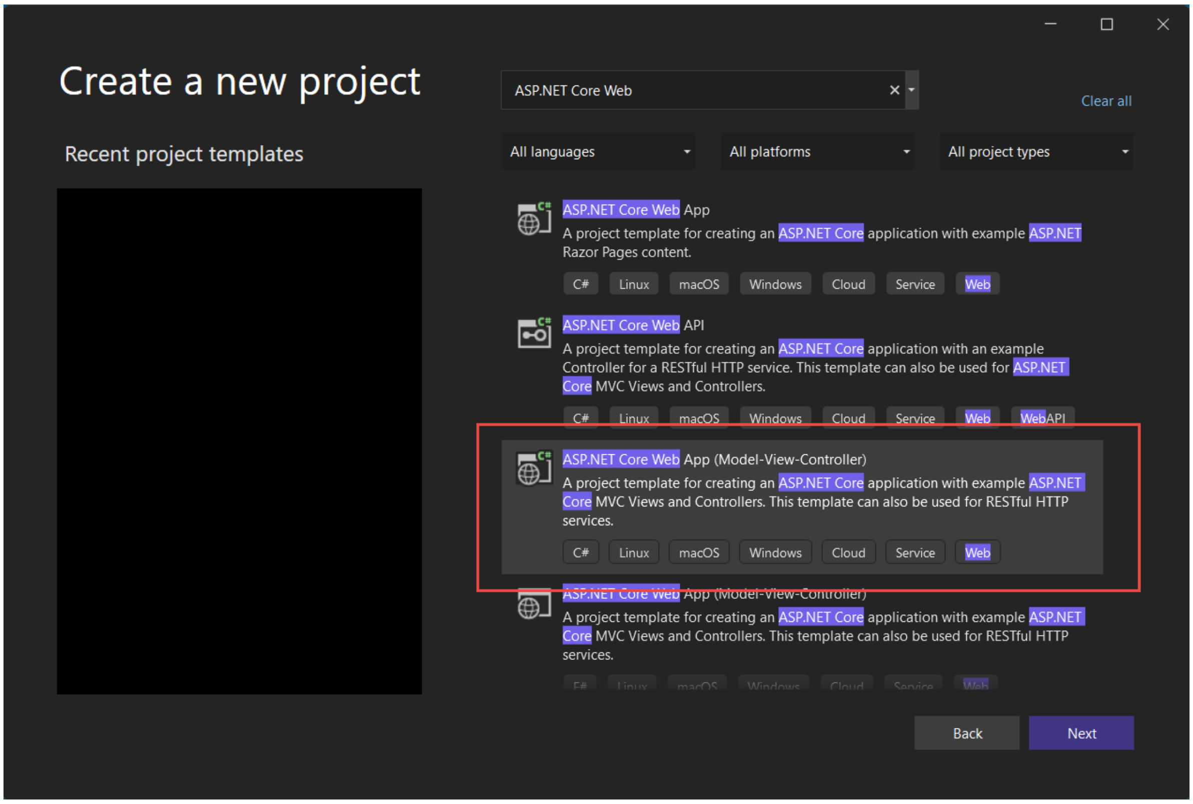
C# Namespace	AdventureWorksLT
C# DbContext Class Name	AdventureWorksLTDbContext
Generation Output Folder	D:\Samples
Backing Field Prefix	_
Include Property Comments?	✓
Include Data Annotations?	✓
Use Full Property Get/Set?	
User PropertyChanged Event?	
Make Primary Key Nullable?	✓

Generate for table 'Product'

Figure 22: Step 4: C# Code Generator Generate Tab

Create a .NET 6 Project

If you generated the code for a .NET 6 application, follow the instructions in this section. Start Visual Studio 2022 and create a new MVC Core Web App (Model-View-Controller) application as shown in Figure 23.


Create a new project

ASP.NET Core Web

Clear all

Recent project templates

[redacted]

All languages | All platforms | All project types

ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web

ASP.NET Core Web API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.
C# Linux macOS Windows Cloud Service Web WebAPI

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# Linux macOS Windows Cloud Service Web

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
F# Linux macOS Windows Cloud Service Web

Figure 23: Create an ASP.NET Core Web App

Click the **Next** button. On the next page set the **Project name** to the same name as your database as that is what the Code Generator is going to use as the default namespace. You can always modify this later, but for now, just name it this.

Click the **Next** button. On this screen select *.NET 6.0 (Long-term support)* from the **Framework** drop down. Select *None* from the **Authentication type** drop down. *Uncheck* the **Configure for HTTPS** check box. Click the **Create** button to create the new application.

Add Entity Framework to your Project

Right mouse-click on the Project folder in Visual Studio and select **Manage NuGet Packages…** from the menu. Click on the **Browse** tab and type in **Microsoft.EntityFrameworkCore.SqlServer**. Install this package into your project.

Generate the MVC and Data Classes Code

In the PDSC Developer Utilities, open the Code Generator and select the .NET 6 (Data Classes) template. Click on the **Step 2: Table to Generate For** tab. Fill in the database name you wish to use to generate from (this should be the same database name as the project name you just created). Click the **Load** button. Click on one of your tables you wish to generate from.

Click on the **Step 3: View Columns** tab. You are only allowed to check or uncheck any of the check boxes in this table. You should not change any of the first 3 columns (Is Nullable, Is Primary Key, and Is Auto Increment), but feel free to modify any of the others. For this first time, you should just leave everything as the default values.

Click on the **Step 4: Generate** tab. The **C# Namespace** field should be filled in with the name of your database. Change the **Generation Output Folder** to a valid folder on your hard drive to where you wish to generate the files. You should not change anything else this first time trying this out. Click the **Generate** button.

Now, click back on the **Step 1: Select Template** tab. Click on the **.NET 6 (MVC)** template. Click back to **Step 4: Generate** tab and click on the **Generate** button again.

Using Windows Explorer, navigate to the folder where you just generated all these files. The folders and files should look like Figure 24.

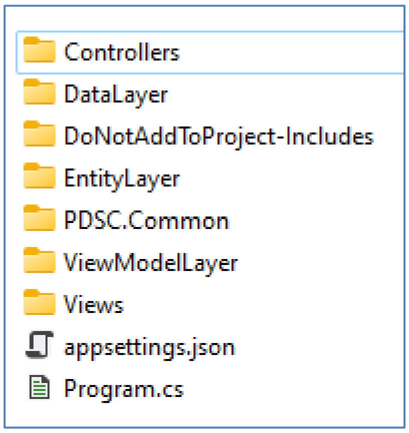

Controllers
DataLayer
DoNotAddToProject-Includes
EntityLayer
PDSC.Common
ViewModelLayer
Views
appsettings.json
Program.cs

Figure 24: The generated folders and files for the .NET 6 MVC project

Select all folders (except for any that start with **DoNotAddToProject**) and files and copy them to the Windows clipboard. Go the project you created in Visual Studio and click on the Project. Paste everything from the Windows clipboard into your project. When prompted if you want to overwrite any existing folders and files, answer **Yes**.

Try it Out

If everything was copied into the correct place, you should be able to compile the project. Open the \Views\Shared_Layout.cshtml file and locate the <li> tag that is used to display the Home menu as shown in the code snippet below.

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-controller="Home"
asp-action="Index">Home</a>
</li>
```

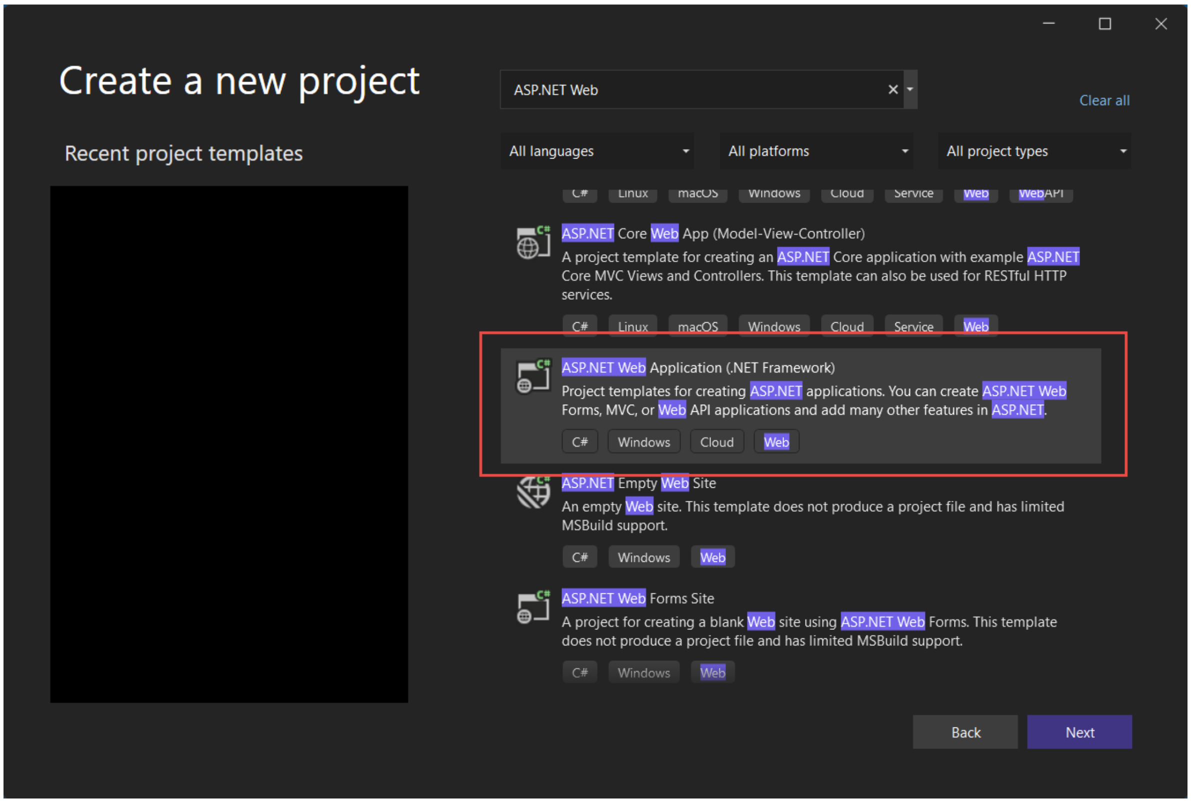
Copy the complete <li>…</li> tag to the clipboard, then paste it back in just below the closing </li>. Modify the <li> tag to use the name of your table.

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-controller="[YOUR
TABLE NAME]" asp-action="[YOUR TABLE NAME]Index">[YOUR TABLE
NAME]</a>
</li>
```

Run the application and click on your new link and you should see a complete CRUD page appear!

Create a .NET Framework 4.x Project

If you generated the code for a .NET Framework 4.x application, follow the instructions in this section. Create a new ASP.NET Web Application (.NET Framework) as shown in Figure 25.



Create a new project

ASP.NET Web

Clear all

Recent project templates

[redacted]

All languages | All platforms | All project types

C# | Linux | macOS | Windows | Cloud | Service | Web | WebAPI

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# | Linux | macOS | Windows | Cloud | Service | Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.
C# | Windows | Cloud | Web

ASP.NET Empty Web Site
An empty Web site. This template does not produce a project file and has limited MSBuild support.
C# | Windows | Web

ASP.NET Web Forms Site
A project for creating a blank Web site using ASP.NET Web Forms. This template does not produce a project file and has limited MSBuild support.
C# | Windows | Web

Back | Next

Figure 25: Create an ASP.NET Web Application (.NET Framework)

Click the **Next** button. On the next page set the **Project name** to the same name as your database as that is what the Code Generator is going to use as the default namespace. You can always modify this later, but for now, just name it this. Make sure you set the **Framework** drop down to *.NET Framework 4.8*.

Click the **Next** button. On this screen select *MVC* from the list of project types. Set the *Authentication* to *None*. *Uncheck* the **Configure for HTTPS** check box. Click the **Create** button to create the new application.

Add Entity Framework to your Project

Right mouse-click on the Project folder in Visual Studio and select **Manage NuGet Packages…** from the menu. Click on the **Browse** tab and type in **EntityFramework**. Install this package into your project.

Generate the MVC and Data Classes Code

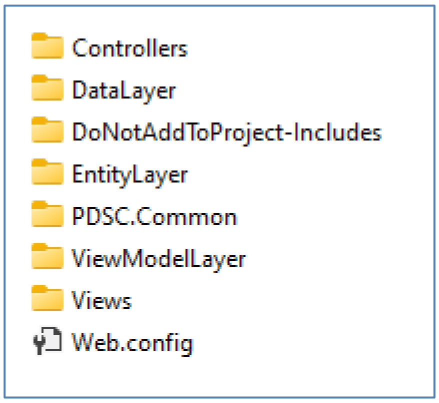
In the PDSC Developer Utilities, open the Code Generator and select the .NET Framework 4.x (Data Classes) template. Click on the **Step 2: Table to Generate For** tab. Fill in the database name you wish to use to generate from (this should be the same database name as the project name you just created). Click the **Load** button. Click on one of your tables you wish to generate from.

Click on the **Step 3: View Columns** tab. You are only allowed to check or uncheck any of the check boxes in this table. You should not change any of the first 3 columns (Is Nullable, Is Primary Key, and Is Auto Increment), but feel free to modify any of the others. For this first time, you should just leave everything as the default values.

Click on the **Step 4: Generate** tab. The **C# Namespace** field should be filled in with the name of your database. Change the **Generation Output Folder** to a valid folder on your hard drive to where you wish to generate the files. You should not change anything else this first time trying this out. Click the **Generate** button.

Now, click back on the **Step 1: Select Template** tab. Click on the **.NET Framework 4.x (MVC)** template. Click back to **Step 4: Generate** tab and click on the **Generate** button again.

Using Windows Explorer, navigate to the folder where you just generated all these files. The folders and files should look like Figure 26.



Controllers
DataLayer
DoNotAddToProject-Includes
EntityLayer
PDSC.Common
ViewModelLayer
Views
Web.config

Figure 26: The generated folders and files for the .NET Framework 4.x MVC project

Select all folders (except for any that start with **DoNotAddToProject**) and files and copy them to the Windows clipboard. Go the project you created in Visual Studio and click on the Project. Paste everything from the Windows clipboard into your project. When prompted if you want to overwrite any existing folders and files, answer **Yes**.

Try it Out

If everything was copied into the correct place, you should be able to compile the project. Open the \Views\Shared_Layout.cshtml file and locate the <li> tag that is used to display the Home menu as shown in the code snippet below.

```
<li>@Html.ActionLink("Home", "Index", "Home")</li>
```

Copy the complete <li>…</li> tag to the clipboard, then paste it back in just below the closing </li>. Modify the <li> tag to use the name of your table.

```
<li>@Html.ActionLink("[YOUR TABLE NAME]", "[YOUR TABLE NAME]Index",
"[YOUR TABLE NAME]")</li>
```

Run the application and click on your new link and you should see a complete CRUD page appear!

What's in the DoNotAddToProject Folder

In the DoNotAddToProject-Includes folder (Figure 27) is where you will find some .txt file that have code you can manually add to some of your previously generated files in your project. For example, you will find a **DbContextDbSets.txt** file that contains the public DbSet<T> property for any tables you have generated. Open this .txt file and copy the code in there and put it into your [NAMESPACE]DbContext.cs file in your application.

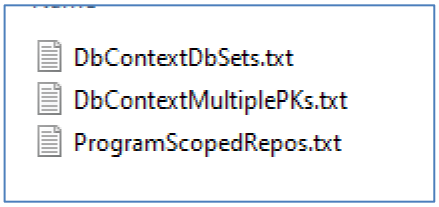

DbContextDbSets.txt
DbContextMultiplePKs.txt
ProgramScopedRepos.txt

Figure 27: Examples of additional code to add to previously generated files

If using .NET 6

You will find a **DbContextMultiplePKs.txt**. If you have any tables that have more than one field for the primary key, you need to open this file and copy the code into your [NAMESPACE]DbContext.cs file.

You will find a **ProgramScopedRepos.txt** file. This file contains the code to add to the **Program.cs** file your newly generated repositories as a service to be used for dependency injection.

Code Generator Tokens

This section explains the various tokens and how they are used throughout the various template files.

Looping Tokens

The following tokens are used to loop through columns and tables.

Token	Description
{|FOR EACH COLUMN|}	Loop through all columns for the table
{|FOR EACH COLUMN:IsEditable|}	Loop through all columns marked for Editing
{|FOR EACH COLUMN:IsInsertable|}	Loop through all columns marked for Inserting
{|FOR EACH COLUMN:IsPrimaryKey|}	Loop through all primary key columns
{|FOR EACH COLUMN:IsNotPrimaryKey|}	Loop through all columns that are not primary keys
{|FOR EACH COLUMN:IsDescriptionField|}	Loop through all columns marked as Description
{|FOR EACH COLUMN:IsSearchField|}	Loop through all columns marked as Search fields
{|FOR EACH COLUMN:DisplayInTable|}	Loop through all columns marked for displaying in table
{|FOR EACH COLUMN:DisplayInEdit|}	Loop through all columns marked for displaying in edit page
{|FOR EACH TABLE|}	Loop through all selected tables
{|END_LOOP|}	Each of the above must be terminated with this. Note, you can NOT nest {|FOR EACH |} tokens

Table 7: Looping Tokens

Remove Tokens

The following tokens can be used to remove blocks of code if a certain condition is not matched.

Token	Description
{|REMOVE_WHEN:IsAutoIncrement|}	Removes the block of code when the Primary key is and IDENTITY property
{|REMOVE_WHEN:IsNotAutoIncrement|}	Removes the block of code when the Primary key is NOT an IDENTITY property
{|REMOVE_WHEN:IsDotNet6OrLater|}	Removes the block of code when the Template is marked as .NET 6 or Later
{|REMOVE_WHEN:IsNotDotNet6OrLater|}	Removes the block of code when the Template is NOT marked as .NET 6 or Later
{|REMOVE_WHEN:IsPrimaryKeyInteger|}	Removes the block of code when the Primary key is an Integer
{|REMOVE_WHEN:IsPrimaryKeyNotInteger|}	Removes the block of code when the Primary key is NOT an Integer
{|REMOVE_WHEN:IsPrimaryKeyString|}	Removes the block of code when the Primary key is a string

{|REMOVE_WHEN:IsPrimaryKeyNotString|}	Removes the block of code when the Primary key is NOT a string
{|REMOVE_WHEN:IsPrimaryKeyGuid|}	Removes the block of code when the Primary key is a unique identifier
{|REMOVE_WHEN:IsPrimaryKeyNotGuid|}	Removes the block of code when the Primary key is NOT a unique identifier
{|REMOVE_WHEN:OnlyOnePrimaryKey|}	Removes the block of code when there is only 1 primary key column
{|END_REMOVE|}	Each of the above must be terminated with this. Note, you can NOT nest {|REMOVE WHEN |} tokens

Table 8: Remove Tokens

Column Tokens

The following tokens are used as you are looping through the list of columns.

Token	Description
<|MAX_LENGTH|> and {|MAX_LENGTH|}	Returns the maximum length marked for a string column in SQL Server
<|PRECISION|> and {|PRECISION|}	Returns the numeric precision for a column in SQL Server
<|SCALE|> and {|SCALE|}	Returns the numeric scale for a column in SQL Server
<|DATETIME_PRECISION|> and {|DATETIME_PRECISION|}	Returns the datetime precision for a column in SQL Server
<|DATA_TYPE|> and {|DATA_TYPE}}	Returns the data type in SQL Server
<|COLUMN_NAME|> and {|COLUMN_NAME|}	Returns the column name in SQL Server
<|PRIMARY_KEY_FIELD|> and {|PRIMARY_KEY_FIELD|}	Returns the primary key column name in SQL Server
<|DESCRIPTION_FIELD|>	Returns the description property for the current column
<|LANGUAGE_DATA_TYPE|>	Returns the data type for the current column for the selected template language
<|PK_PROPERTY_NAME|>	Returns the property name for the primary key field for a table
<|PK_LANGUAGE_DATA_TYPE|>	Returns the data type (and possibly nullable type) of the primary key field for a table for the selected template language

<|PK_LANGUAGE_DATA_TYPE_NEVER_NULLABLE|>	Returns just the data type of the primary key field for a table for the selected template language
<|PRIVATE_FIELD_PREFIX|>	Returns the specified backing field prefix such as an underscore
<|PROPERTY_NAME|>	Returns the property name for the current column
<|PROPERTY_NAME_LOWER_FIRSTCHAR|>	Returns the property name with the first character as lower-case for the current column
<|PROPERTY_NAME_ALL_LOWER|>	Returns the lower-case version of the property name for the current column
<|PROPERTY_LABEL|>	Returns the Label for the current column
<|PROPERTY_SEARCH_PATTERN|>	Returns the Search Pattern used in the View Model templates. Comes from LanguageDataType XML File.
<|PROPERTY_INITIALIZER|>	Returns the Property Initializer that can be used for initializing property values for the data type of the current column. Comes from LanguageDataType.xml File
{|DATA_ANNOTATION:[AnnotationName]|}	Returns a single Data Annotation by looking at the Name and Language elements in the CodeGen-DataAnnotations.xml file

Table 9: Column Tokens

Table Tokens

The following tokens are used for table information, or as you are looping through the list of tables.

Token	Description
<|TABLE_DATA_ANNOTATION|>	Returns the data annotation to apply to an entity class for use with the Entity Framework
<|NAMESPACE|> and {|NAMESPACE|}	Returns the Namespace specified in the code generator
<|CLASS_NAME|> and {|CLASS_NAME|}	Returns the Class Name property for the current table
<|CLASS_NAME_SINGULAR|>	Returns the Singular version of the Class Name for the current table
<|CLASS_NAME_PLURAL|>	Returns the Pluralized version of the Class Name for the current table

<|CLASS_DESC_SINGULAR|>	Returns the Singular version of the Description for the current table
<|CLASS_DESC_PLURAL|>	Returns the Pluralized version of the Description for the current table
<|CATALOG_NAME|>	Returns the Catalog for the current table
<|SCHEMA_NAME|>	Returns the Schema for the current table
<|TABLE_NAME|>	Returns the Table name for the current table
<|SELECT_SQL|>	Returns the SQL used to select all columns for the current table

Table 10: Table Tokens

Code Generation Template Tokens

The following tokens are used for information about the current template.

Token	Description
<|BASE_CLASS_NAME|>	Returns the value from the BaseClassName element from current template XML file. These files generally follow the pattern "CodeGen-?.xml" file
<|OUTPUT_PREFIX|>	Returns the value from the OutputFilePrefix element from current template XML file. These files generally follow the pattern "CodeGen-?.xml" file
<|OUTPUT_SUFFIX|>	Returns the value from the OutputFileSuffix element from current template XML file. These files generally follow the pattern "CodeGen-?.xml" file
<|OUTPUT_FILE_EXTENSION|>	Returns the value from the OutputFileExtension element from current template XML file. These files generally follow the pattern "CodeGen-?.xml" file
<|PROPERTIES|>	Generates all Properties for use with the Entity Framework entity class. These properties are generated using all the appropriate data annotations
<|DB_CONTEXT|>	The DbContext class name as specified in the code generator
<|CONNECTION_STRING|>	Returns the Connection string as specified in the code generator

Table 11: Code Generation Tokens

Miscellaneous Generation Tokens

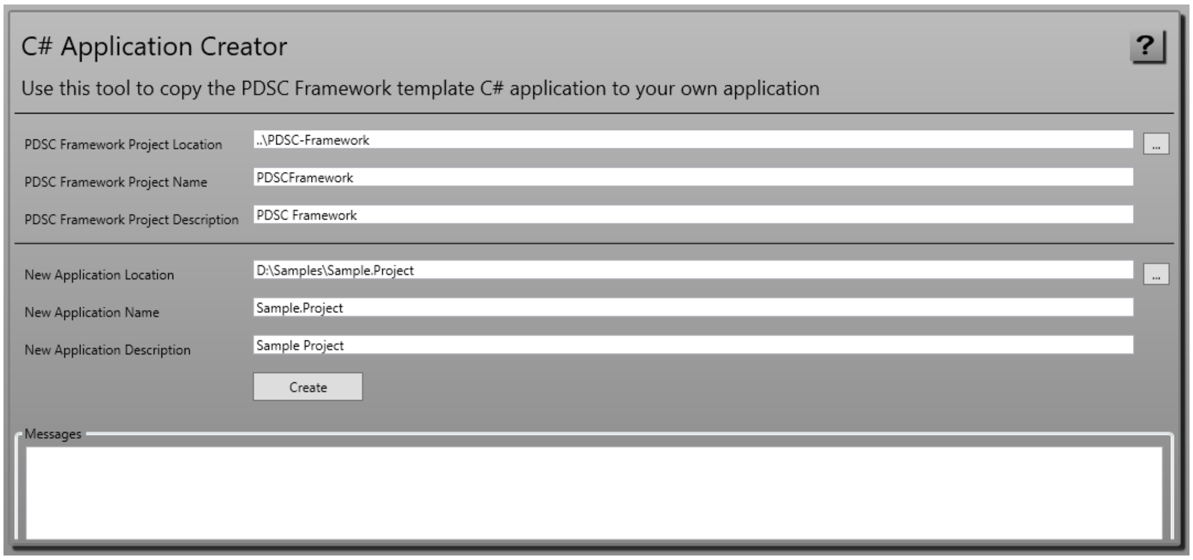
The following tokens are used for various functionality while generating code.

Token	Description
<|CRLF|> and {|CRLF|}	Returns a carriage return, line feed
<|LOGICAL_AND|>	Returns single ampersand (&) after the first time through a loop, and following a <|LOGICAL_AND_RESET|>
<|LOGICAL_AND_RESET|>	Resets "<|LOGICAL_AND|>" token to an empty string
<|LOGICAL_OR|>	Returns single vertical bar (|) after the first time through a loop, and following a <|LOGICAL_OR_RESET|>
<|LOGICAL_OR_RESET|>	Resets "<|LOGICAL_OR|>" token to an empty string
<|CONDITIONAL_AND|>	Returns double ampersands (&&) after the first time through a loop, and following a <|CONDITIONAL_AND_RESET|>
<|CONDITIONAL_AND_RESET|>	Resets "<|CONDITIONAL_AND|>" token to an empty string
<|CONDITIONAL_OR|>	Returns double vertical bars (||) after the first time through a loop, and following a <|CONDITIONAL_OR_RESET|>
<|CONDITIONAL_OR_RESET|>	Resets "<|CONDITIONAL_OR|>" token to an empty string
<|COMMA|>	Returns a comma (,) after the first time through a loop, and following a <|COMMA_RESET|>
<|COMMA_RESET|>	Resets "<|COMMA|>" token to an empty string
<|AND|>	Returns the word "And" after the first time through a loop, and following a <|AND_RESET|>
<|AND_RESET|>	Resets "<|AND|>" token to an empty string
<|OR|>	Returns the word "Or" after the first time through a loop, and following a <|OR_RESET|>
<|OR_RESET|>	Resets "<|OR|>" token to an empty string
<|REMOVE_LINE|>	Do not add the current line to the output
{|INCLUDE:FILENAME|}	Allows you to include one template file into the specified location of another template file. This included template file is inserted into the location after all other templates have been generated.

Table 12: Miscellaneous Generation Tokens

C# Application Creator (COMING SOON)

Click on the C# App Creator menu to see a screen that looks like Figure 28. Modify the "New Application Location" to a valid hard drive, and folder, on your system and click the **Create** button. If you get an error that one of the paths is incorrect, fix it up, then click the Create button again. In just a few seconds you should receive a message that the process is complete, and you will see a bunch of messages as seen at the bottom of the screen.



C# Application Creator

Use this tool to copy the PDSC Framework template C# application to your own application

PDSC Framework Project Location	..\PDSC-Framework
PDSC Framework Project Name	PDSCFramework
PDSC Framework Project Description	PDSC Framework
New Application Location	D:\Samples\Sample.Project
New Application Name	Sample.Project
New Application Description	Sample Project

Figure 28: The PDSC C# Application Creator helps you build a new MVC project.

You can now go to the "New Application Location" folder and view the results of running this tool as shown in Figure 29.

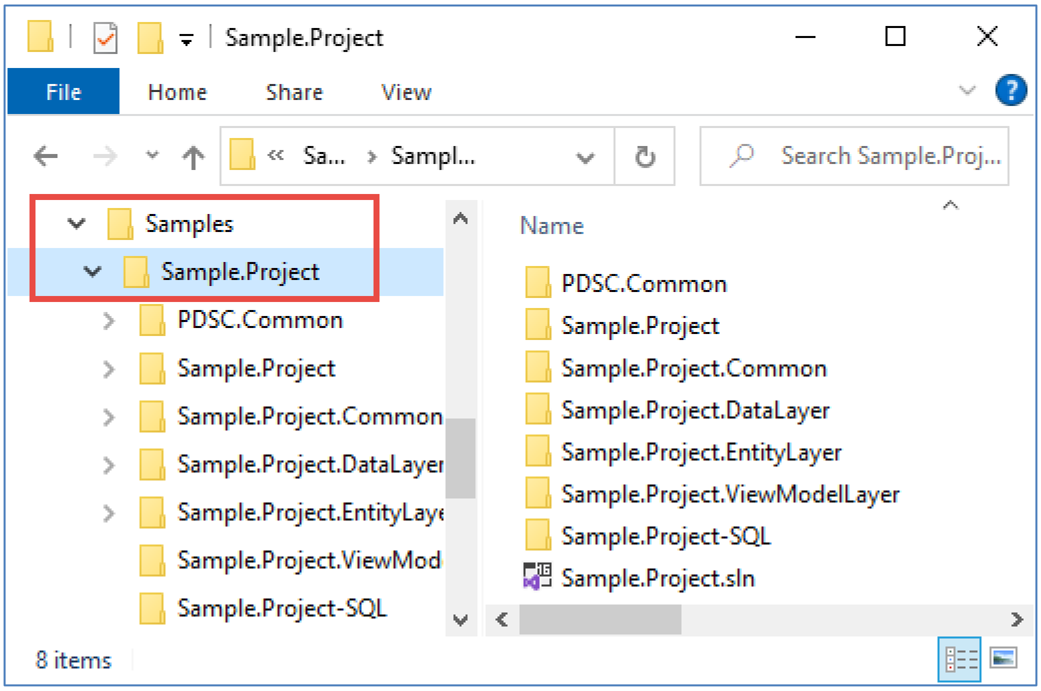


Figure 29: In the Application Location folder you find a folder structure like that of the PDSC Framework template.

Create a Sample Database

Open the SQL Server Management Studio and create a new database named **Sample.Project**. Open the "Sample.Project-SQL" folder and locate the **Sample.Project.sql** file (Figure 30) and load that file into SQL Server Management Studio. Run this script to create the database objects. Open the **Sample.Project-Data.sql** file in SQL Server Management Studio and run this script to add data to the database objects.

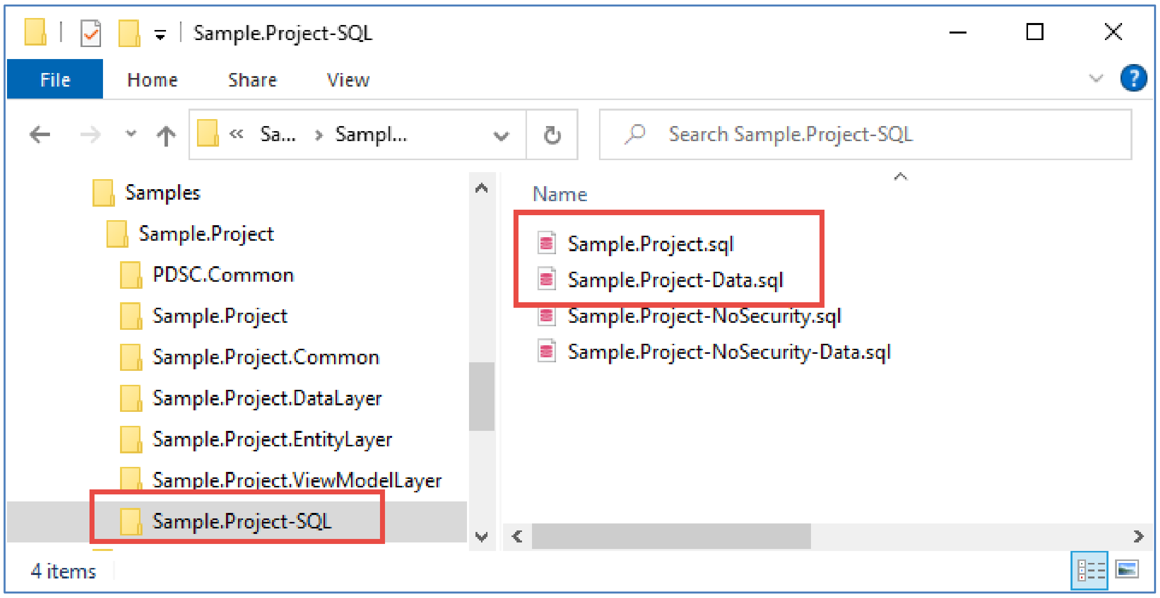


Figure 30: Locate the .SQL files to install in your new application folder.

Run the Sample Project

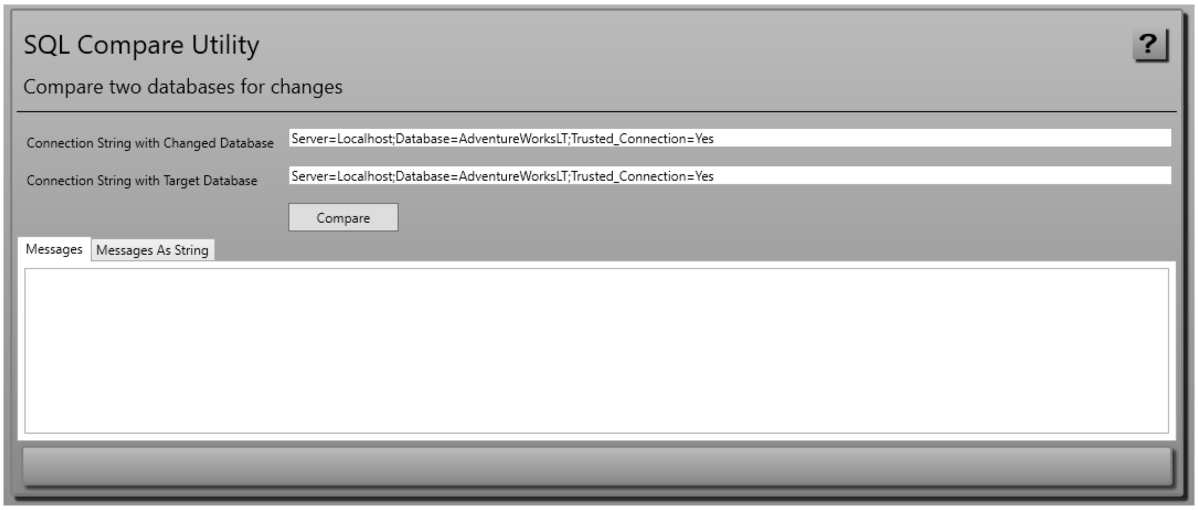
Go to the **\Sample.Project** folder and double-click on the **Sample.Project.sln** file. Run this project. When you get to the home page, click on the Login link and login with 'bill@microsoft.com' and the password 'P@ssw0rd'. If you have done everything correctly, you should now be logged into your sample application.

Next Steps

Please read the chapter on the **Haystack Code Generator** for information on how to generate add/edit/delete pages for your new project.

SQL Compare

When you run the SQL Compare utility, you put in two different connections string that point to similar databases. For example, maybe you need to find out what you changed in your QA database compared to your Production database. Click the Compare button (shown in Figure 31) and a complete list of missing or changed objects will appear in the messages tabs at the bottom of the screen.



Connection String with Changed Database: Server=Localhost;Database=AdventureWorksLT;Trusted_Connection=Yes

Connection String with Target Database: Server=Localhost;Database=AdventureWorksLT;Trusted_Connection=Yes

Figure 31: Get missing objects from one database to another via the SQL Compare Utility.

Summary

The PDSC Developer Utilities will help increase your productivity while developing your applications. We hope you enjoy using this product.