Sage 300 Web Screens SDK

Code Generation Wizard

August 2021

The MIT License (MIT)

Copyright © 2021 The Sage Group plc or its licensors. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

[1. Overview 4](#_Toc78902764)

[2. Accessing the Code Generation Wizard 5](#_Toc78902765)

[3. Using Code Generation Wizard 7](#_Toc78902766)

[3.1 Step 1: Select Code Type and Credentials 7](#_Toc78902767)

[3.2 Step 2: Add/Edit Entities 10](#_Toc78902768)

[3.2.1 Entity Tab 11](#_Toc78902769)

[3.2.2 Options Tab 13](#_Toc78902770)

[3.2.3 Properties Tab 15](#_Toc78902771)

[3.2.4 Composition Tab 16](#_Toc78902772)

[3.3 Step 3: Generate a UI Layout 17](#_Toc78902773)

[3.4 Step 4: Generate Code 18](#_Toc78902774)

[3.5 Step 5: Generated Code 19](#_Toc78902775)

[4. Examining the Solution 20](#_Toc78902776)

[4.1 Solution Explorer 20](#_Toc78902777)

[4.2 Business Repository Project 21](#_Toc78902778)

[4.3 Interfaces Project 21](#_Toc78902779)

[4.4 Models Project 22](#_Toc78902780)

[4.5 Resources Project 22](#_Toc78902781)

[4.6 Services Project 23](#_Toc78902782)

[4.7 Web Project 24](#_Toc78902783)

1. Overview

This document presents instructions for using the Code Generation Wizard to create Sage 300 Web User Interfaces (“UIs”), including screens, reports, inquiries, and so on.

The Code Generation Wizard is a Visual Studio Plugin that was developed in Visual Studio 2019 and is compatible with Visual Studio 2019.

The intent of the wizard is to generate code files in a Sage 300 Web UI solution. The Solution Wizard has already created a solution which provides the necessary scaffolding and structures to accommodate the generated code files by this wizard.

The wizard will create code files in the following projects:

* Business Repository

Contains the code files for the Entity Repositories, Entity Mappers, Menu Navigation XML, Security Constants, and so on.

* Interfaces

Contains the code files for the Entity Repository/Service Interfaces.

* Models

Contains the code files for the Models, Model Enumerations, and Model Fields.

* Resources

Contains the code files for the Resources (Resx files) for supported languages.

* Services

Contains the code files for the Services.

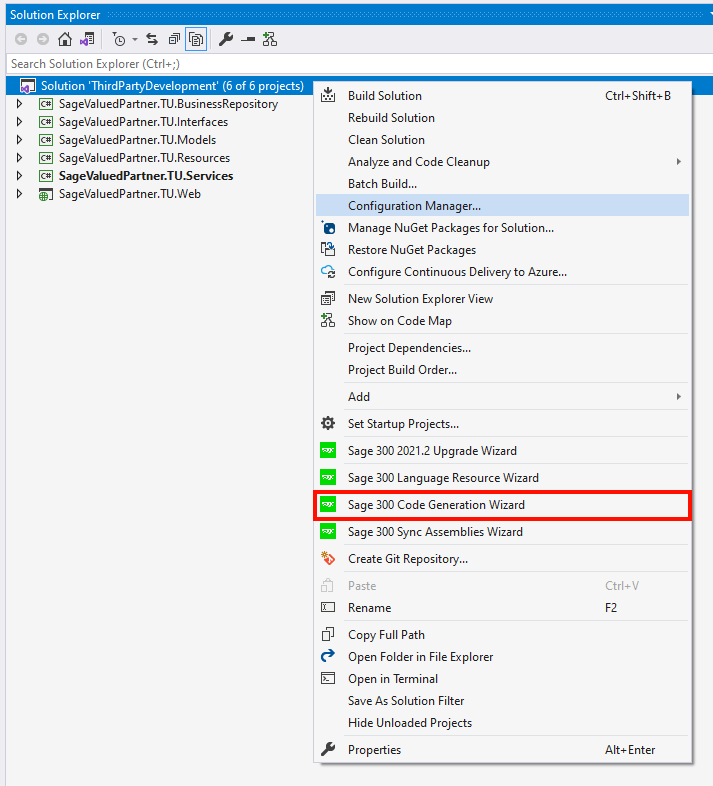
* Web

Contains the standard code files based upon MVC patterns (Controllers, View Models, JavaScript, Razor Views, Configuration files, and so on).

Like the Solution Wizard, this wizard is embedded directly into the Visual Studio IDE.

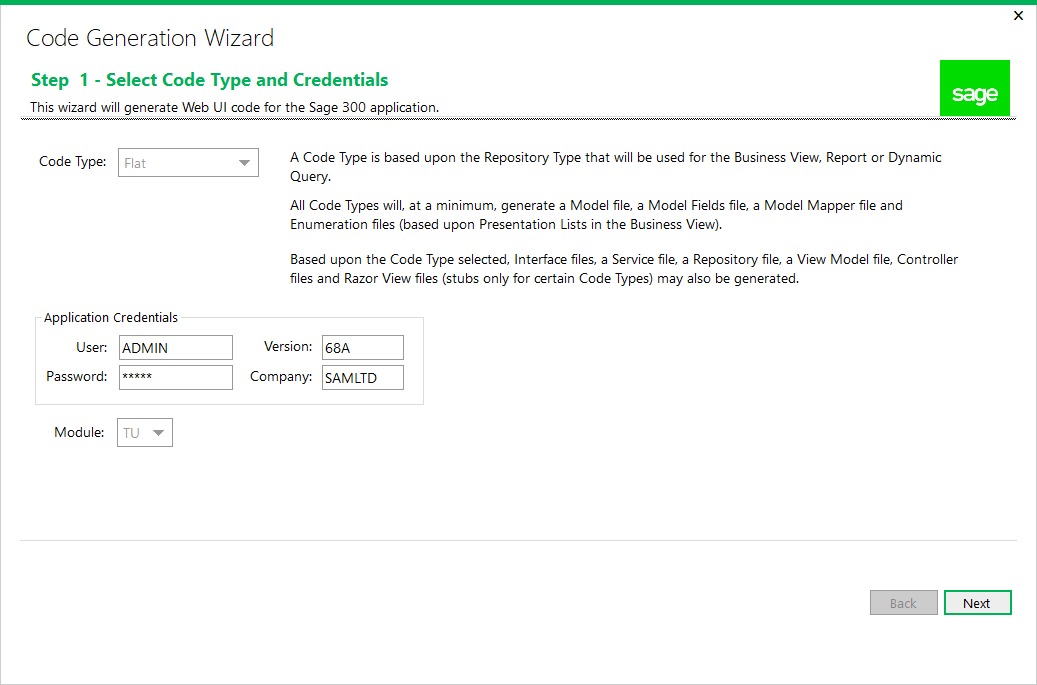
1. Accessing the Code Generation Wizard

The wizard is embedded in Visual Studio as a plugin. To open it, right-click the solution, and on the context menu, click Sage 300 Code Generation Wizard.



The wizard will perform validation to ensure that the invoking solution is a Sage 300 Web UI solution. If it is not (due to the required projects not being present), a message informs you that the solution does not include all required Sage 300 projects.

1. Using Code Generation Wizard
   1. Step 1: Select Code Type and Credentials



Code Types

At time of publication, the Dynamic Query, Report, Inquiry and Header Detail code types are not fully implemented. This means that while the code compiles successfully, there are components that have not yet been implemented (i.e. Razor Views, JavaScript, etc.)

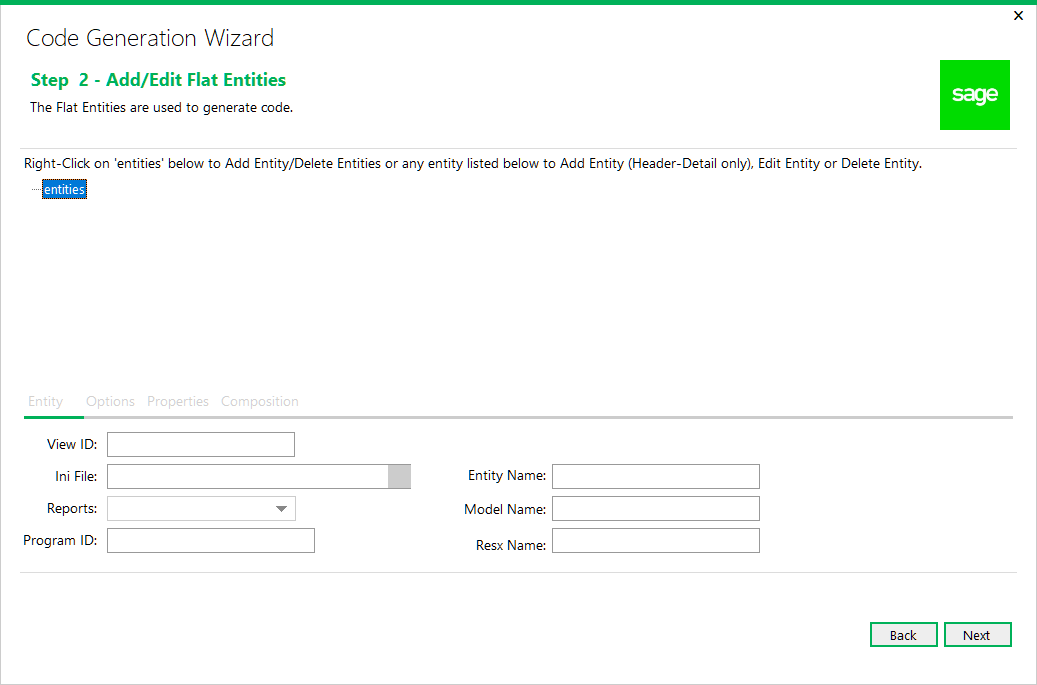
* Flat
* To be used for simple setup screens
* Will use the FlatRepository base class
* Will use a Business View to generate code files
* Process
* To be used for process screens
* Will use the ProcessRepository base class
* Will use a Business View to generate code files
* Dynamic Query
* To be used for SQL statement screens (i.e. KPIs)
* Will use the DynamicQueryRepository base class
* Will use a manually created model to generate code files
* Report
* To be used for reports
* Will use the ReportRepository base class
* Will use a Report INI file to generate code files
* Inquiry
* To be used for inquiry screens
* Will use the InquiryRepository base class
* Will use a Business View to generate code files
* Header Detail
* To be used for screens with business views with a header/detail relationship
* Will use the BaseHeaderDetailRepository base class
* Will use Business Views to generate code files

Application Credentials are defaulted and may be overridden to connect to the Business View.

The Module list contains a list of Modules discovered in the solution. If only one Module is discovered, it is defaulted.

Click Next to proceed.

* 1. Step 2: Add/Edit Entities



As of 2018.1, this step combines the actions for all code types with certain fields and functionality disabled based upon the code type.

The wizard is now able to generate multiple entities which is required to support the header detail code type. Only the header detail code type allows for hierarchical entities to be created.

Right-click on the ‘**entities**’ tree node and select **Add Entity** to add a new entity or select **Delete Entities** to delete all entities already added. If the code type is **HeaderDetail**, select **Edit Container Name** to display a dialog for specifying the name of the container for the header detail entities (i.e. Receipt for a Receipt Entry screen).

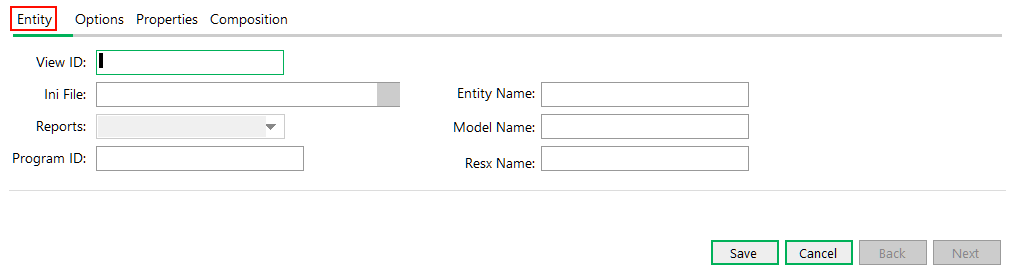
Right-click on **{entity}** in the tree and select **Edit Entity** to edit an already added entity.

Right-click on **{entity}** in the tree and select **Delete Entity** to delete an already added entity.

If the code type is **HeaderDetail**, child entities may be added by right-clicking on **{entity}** in the tree and select **Add Entity**.

Click Next to proceed or Back to go back to the previous step.

* + 1. Entity Tab

****

The Entity Tab has properties specific to the entity:

**View ID**

If the code type is **Report**, the generation of the code files are based upon a Sage 300 report definition from an INI file. Therefore, the View ID is a randomly generated static GUID and the field is disabled.

If the code type is **DynamicQuery**, the **View ID** is associated with a generic Business View (CS0120), which allows a SQL query to be submitted to the Business View for data retrieval. Therefore, the View ID field is disabled.

When editing an entity, the field is disabled.

Upon selecting a View ID, the Application Credentials from the previous step will be used to access the Business View and the **Entity** Name, Model Name and Resx Name will be defaulted. These may be overridden but the Resx Name must have the “Resx” suffix.

**Ini File**

If the code type is **Report**, Report definitions for Sage 300 are in INI files. Click on the magnifying glass button to display an Open File dialog to search for an existing INI file, or enter the file name manually.

**Reports**

If the code type is **Report**, this list is populated from the INI file. It contains the report names in alphabetical order. The report name selected will be used to populate the Model Fields grid in the Properties tab.

Program ID

If the code type is **Report**, this field is assigned from the INI file and will be used in the Model Mapper class.

**Entity Name**

This field is defaulted from the retrieved Business View and may be overridden as required. It is used as the prefix name for the code files that will be created. For **DynamicQuery** and **Report** code types, this field must be manually entered as there is no source to default it from.

**Model Name**

This field is defaulted from the retrieved Business View and may be overridden as required. It is used as the name for the model that will be created. For **DynamicQuery** and **Report** code types, this field must be manually entered as there is no source to default it from.

If the code type is **DynamicQuery**, a Model Mapper file will not be generated as mapping is an abstract function of the repository class. Also, the Model Fields class will only have an Index class and not a Fields class as these model properties will only be referenced by index.

**Resx Name**

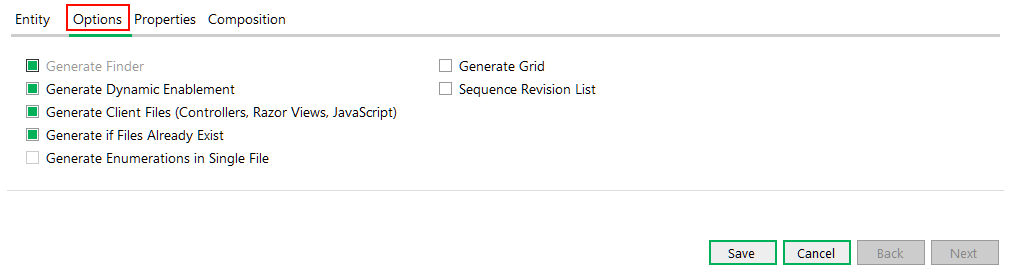
This field is defaulted from the retrieved Business View and may be overridden as required. It is used as the name for the Resx file that will be created. For **DynamicQuery** and **Report** code types, this field must be manually entered. It has a requirement that it must have the “Resx” suffix. This field will also be synchronized with any changes made to the entity name.

Up to five Resx files will be generated based upon the languages selected in the Solution Wizard. The main Resx file is English and will contain the descriptions for the Model properties and Enumerations from the Business View.

The other Resx files, if selected in the Solution Wizard, are Spanish (es), French (fr), Chinese Simplified (zh-Hans) and Chinese Traditional (zh-Hant). These files will contain only keys and will require manual translation.

When done adding or editing an entity, click **Save** to save the entity when finished or **Cancel** to disregard any changes.

* + 1. Options Tab

****

The Options Tab has options specific to the entity:

**Generate Finder**

If selected, this option will generate a finder for the entity. Certain code types have this option disabled as a finder is not applicable to all code types.

Generate Dynamic Enablement

If selected, this option provides a reverse mapping of the Business View fields. Within the Sage accounting modules, the Inventory Control, Order Entry, and Purchase Orders Business Views contain more functionality for determining if a field is editable or not. This will generate the code for determining if the UI or Business View is responsible for checking the editable state. Certain code types have this option disabled.

Generate Client Files

If selected, this option will generate the client files for a Business View (controllers, Razor Views, and JavaScript files) and thus provides for the ability to generate certain code files without the client files being generated. Certain code types have this option disabled.

Generate if Files Already Exist

If selected, this option provides a warning dialog if a code file to be generated already exists. The developer may choose to override an existing file, skip the file from being generated, or exit the wizard all together.

Generate Enumerations in Single File

If selected this option will generate all enumerations in a single file instead of in individual files.

This option is not yet enabled and is for future use.

Generate Grid

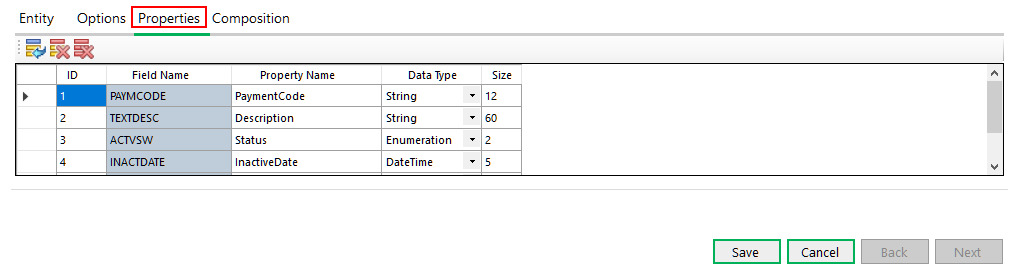
[TODO] This is the description text for the ‘Generate Grid’ checkbox.

Sequence Revision List

[TODO] This is the description text for the ‘Sequence Revision List’

When done adding or editing an entity, click **Save** to save the entity when finished or **Cancel** to disregard any changes.

* + 1. Properties Tab

****

The Properties Tab has the fields that are associated with the Business View.

The above screenshot shows properties for the **AR0012** Business View.

For **DynamicQuery** and **Report** code types, this information must be manually entered.

For all other code types that use a Business View for the View ID, this grid allows for the modification of the Property Name before the code is generated. This can be very useful if the default names are not sufficient or the developer wishes to use another name. Additionally, fields may be added or deleted as required by the developer to add fields to the model that are not found in the Business View.

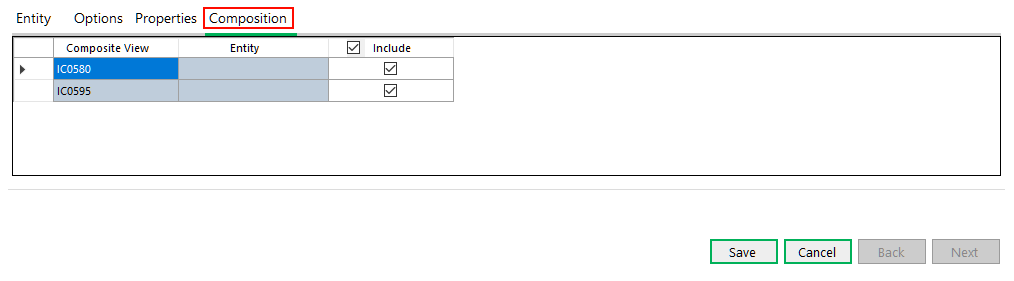
If the code type is **DynamicQuery**, the order of the properties is important and must match the columns returned by the SQL query. The Size column is used for the Display annotation on the Model property and for the Finder attributes if the option to generate a Finder is selected.

If the code type is Report, the Field Name column is read-only and is the server field name whereas the Property Name column is the name for the model properties. A simple algorithm will be applied to the property name column based upon the server field name. The Size column is used for the Display annotation on the Model property and for the Finder attributes if the option to generate a Finder is selected.

All properties are defined as String data type based upon the requirements of the report engine.

When done adding or editing an entity, click **Save** to save the entity when finished or **Cancel** to disregard any changes.

* + 1. Composition Tab

****

The Composition Tab has the compositions that are associated with the Business View.

This tab is only available if the code type is **HeaderDetail**.

The above screenshot shows compositions for the **IC0590** Business View.

The compositions are defaulted to be included and these should be included for any header entities.

The **Include** option, if selected, will ensure that the **Composite View** is associated with an entity specified in the list of entities. Once a Composite View has been specified in the list of entities, the Entity will be assigned from the **Entity Name** field of that entity. The validation logic will ensure this functionality.

The **Include** option, if not selected, will not require the entity to be in the list of entities while still allowing the composition code to assign a Null value to the entity in the composition routine. In the above **IC0590** example, when adding the child to this entity (**IC0595**), there will be numerous Composite Views specified, however, not all should be included.

Refer to the Receipt Sample in the Sage 300 Web SDK for the composition routine generated by the wizard.

Familiarity with the compositions is a requirement for the developer using the **HeaderDetail** code type.

When done adding or editing an entity, click **Save** to save the entity when finished or **Cancel** to disregard any changes.

* 1. Step 3: Generate a UI Layout

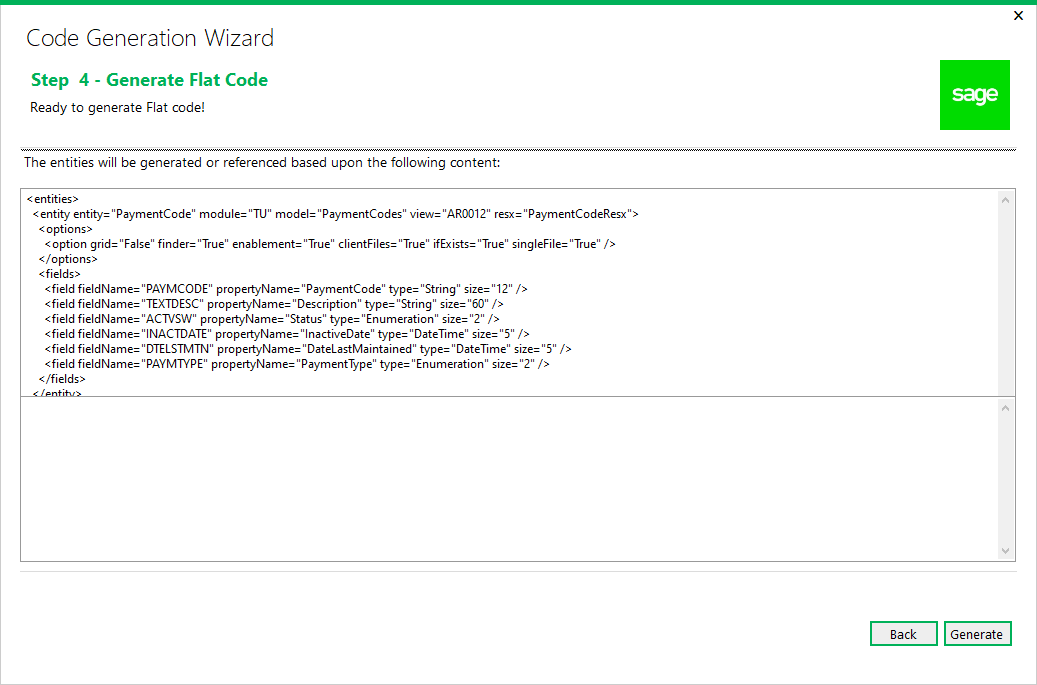
Table

Description automatically generated

TODO – Add correct text here.

Click Generate to proceed or Back to go back to the previous step.

* 1. Step 4: Generate Code

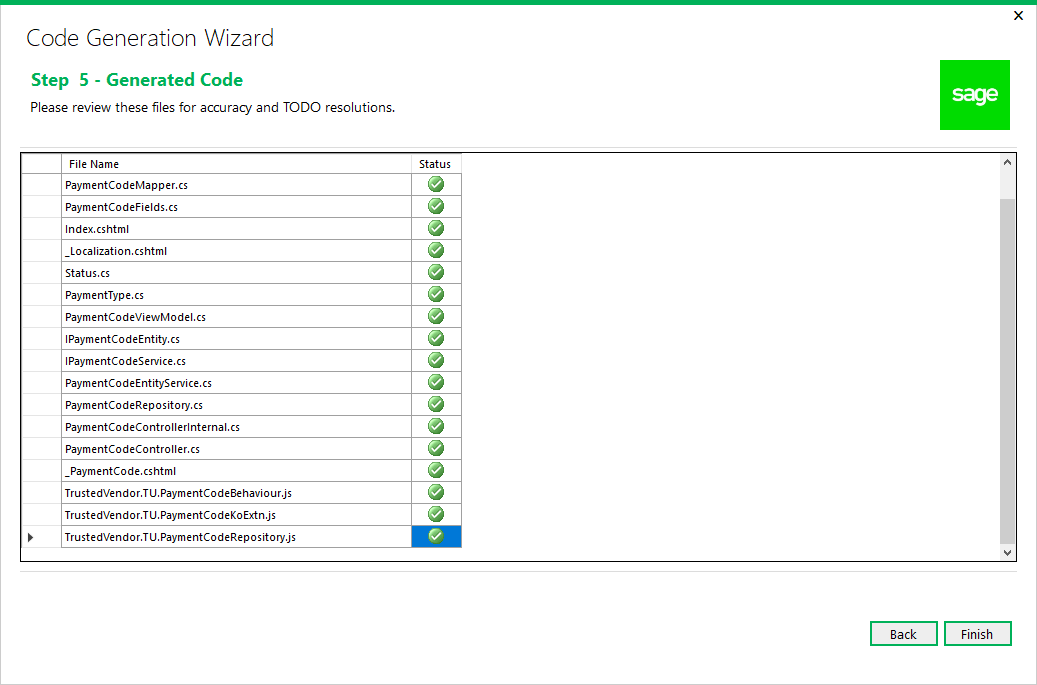


This is the confirmation prior to the wizard generating the code based upon the wizard inputs.

The wizard will generate code based on the XML shown in the preview.

Click Generate to proceed or Back to go back to the previous step.

* 1. Step 5: Generated Code



The wizard displays the code files that were generated. It is now time to review the generated files for accuracy and resolve any TODO issues.

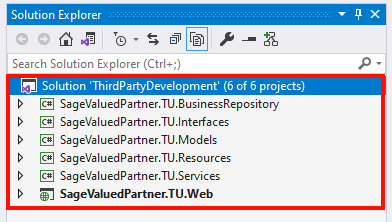
TODO issues are C# comments (i.e. // TODO something…) that provide explicit instructions for a developer to resolve or verify wizard generated code.

Click Finish to exit the wizard or Back to go back to the previous step.

1. Examining the Solution

It is time to examine the solution and get familiar with the class and folder scaffolding.

* 1. Solution Explorer



Note the following information:

* The solution name supplied in the New Project dialog.
* The namespace of the projects supplied in the wizard dialog.
  1. Business Repository Project



The wizard has created the required folders and a couple of classes that will be used by the Code Generation Wizard.

Note the presence of the Process and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the appropriate Mapper and Repository class will be generated in these subfolders as opposed to the root folders.

The TUMenuModuleHelper.cs file already has generated code for Menu Navigation.

The Security.cs file already has constants generated for the Import and Export constants for the generated Module ID.

* 1. Interfaces Project



The wizard has created the required folders that will be used by the Code Generation Wizard.

Note the presence of the Process and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the appropriate Interface classes will be generated in these subfolders as opposed to the root folders.

* 1. Models Project



The wizard has created the required folders that will be used by the Code Generation Wizard.

Note the presence of the Process and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the appropriate Enumerations, Fields and Model classes will be generated in these subfolders as opposed to the root folders.

* 1. Resources Project



The wizard has created the required folders that will be used by the Code Generation Wizard.

Note the presence of the Forms, Process, and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the appropriate Resx files will be generated in these subfolders as opposed to the Forms folders.

Note the generated Resx files:

* MenuResx.resx is the English resource.
* MenuResx.es.resx is the Spanish resource, if Spanish is included via the Solution Wizard Resource Files Step.
* MenuResx.fr.resx is the French resource, if French is included via the Solution Wizard Resource Files Step.
* MenuResx.zh-Hans.resx is the Chinese Simplified resource, if Chinese Simplified is included via the Solution Wizard Resource Files Step.
* MenuResx.zh-Hant.resx is the Chinese Traditional resource, if Chinese Traditional is included via the Solution Wizard Resource Files Step.

Only the English Resx file is marked as Public. All other resx files are marked as No Code Generation.

Any Resx file that is common to all screens/reports/whatever in a module is to be placed in the root folder.

* 1. Services Project



The wizard has created the required folders and a class that will be used by the Code Generation Wizard.

Note the presence of the Process and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the Service class will be generated in these subfolders as opposed to the root folders.

The TUBootstrapper.cs file already has generated code for Unity Dependency Injection which will be added to by the Code Generation Wizard.

* 1. Web Project



The wizard has created the required folders, files and classes that will be used by the Code Generation Wizard.

Note the presence of the Finder, Process, and Reports folders. In the Code Generation Wizard, if a Code Type of Process or Reports is selected, the appropriate classes will be generated in these subfolders as opposed to the root folders. If selected in the Code Generation Wizard Options, the Finder will be created in the Finder folder.

The TUBootstrapper.cs, TUAreaRegistration.cs, TUMenuDetails.xml and TUWebBootstrapper.cs files already have generated code for Unity Dependency Injection, which will be added to by the Code Generation Wizard.