Sage 300 Web Screens SDK

Code Generation Wizard

December 2016

The MIT License (MIT)

Copyright © 2016 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](#_Toc440882984)

[2. Accessing the Code Generation Wizard 5](#_Toc440882985)

[3. Using Code Generation Wizard 6](#_Toc440882986)

[3.1 Step 1: Select Code Type 6](#_Toc440882987)

[3.2 Step 2: Enter View and Credentials 8](#_Toc440882988)

[3.3 Step 2: Enter Dynamic Query Properties 9](#_Toc440882989)

[3.4 Step 2: Enter Report Properties 11](#_Toc440882990)

[3.5 Step 3: Enter Resource Name 13](#_Toc440882991)

[3.6 Step 4: Select Options 14](#_Toc440882992)

[3.7 Step 5: Generate Code 15](#_Toc440882993)

[3.8 Step 6: Generated Code 16](#_Toc440882994)

[4. Examining the Solution 17](#_Toc440882995)

[4.1 Solution Explorer 17](#_Toc440882996)

[4.2 Business Repository Project 17](#_Toc440882997)

[4.3 Interfaces Project 18](#_Toc440882998)

[4.4 Models Project 19](#_Toc440882999)

[4.5 Resources Project 19](#_Toc440883000)

[4.6 Services Project 20](#_Toc440883001)

[4.7 Web Project 21](#_Toc440883002)

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 2013.

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.

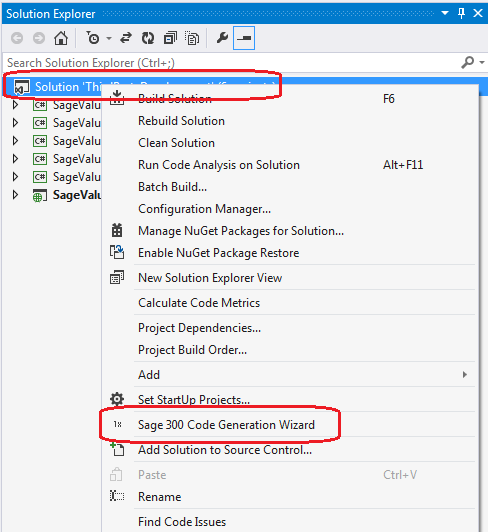
* Web

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.

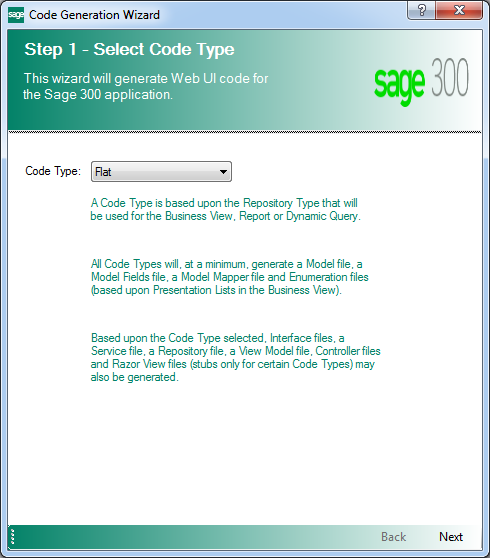
1. Accessing the Code Generation Wizard

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



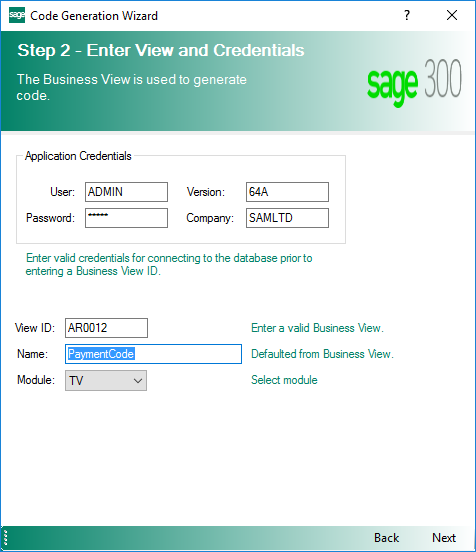
Code Types

Note: At time of publication, the Flat, Process, Dynamic Query, Report, and Inquiry code types are not fully implemented.

* Flat
* To be used for simple setup screens
* To use the FlatRepository base class
* Will use a Business View to generate code files
* Process
* To be used for process screens
* To 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)
* To use the DynamicQueryRepository base class
* Report
* To be used for reports
* To use the ReportRepository base class
* Inquiry
* To be used for inquiry screens
* To use the InquiryRepository base class

Click Next to proceed.

* 1. Step 2: Enter View and Credentials



**Note:** This step is displayed for Flat, Process, and Inquiry Code types. If Dynamic Query or Report is selected as the Code Type, a different step 2 will be displayed.

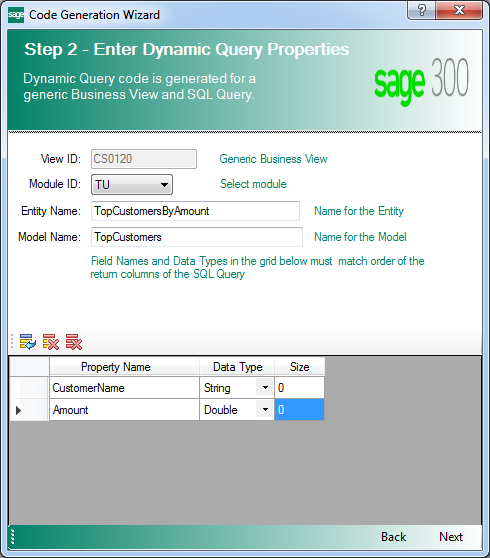
Application Credentials are defaulted and may be overridden in order to connect to the Business View.

Upon selecting a View ID, the Application Credentials will be used to access the Business View and the Name will be defaulted. The Name is made singular and may be overridden if required or desired.

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 or Back to go back to the previous step.

* 1. Step 2: Enter Dynamic Query Properties



**Note:** This step is displayed for the Dynamic Query Code Type. If Flat, Process, Report or Inquiry are selected as the Code Type, a different step 2 will be displayed.

Unlike other Code Types that are based upon a specific Business View, the Dynamic Query Code Type 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 is static.

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

The Entity Name field takes the place of the Business View description which is defaulted for other Code Type, but may be overridden. Therefore, this field must be entered, as there is no source to default it from.

The Model Name field is the name used for the Model created by the wizard. This is the model that is assigned data from the SQL query. It may be the same as the Entity Name if the developer chooses (in the Flat, Process and Inquiry code types, the Entity Name and the Model Name as the same). Therefore, this field must be entered as there is no source to default it from.

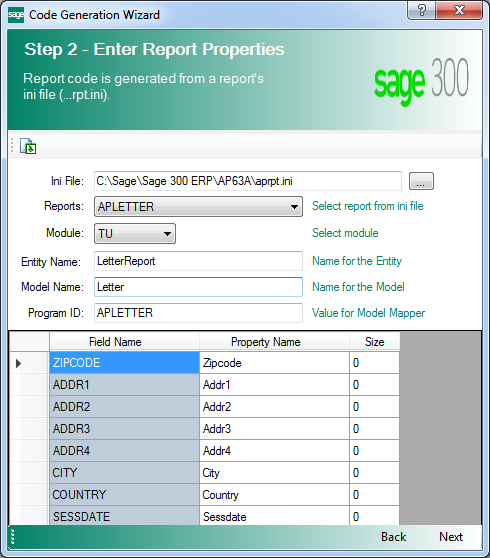
**Note:** A Model Mapper file will not be created generated as mapping is an abstract function of the repository class. Also, the Model Fields class will only have an Index class and not a Fields class as these model properties will only be referenced by index.

The Model Fields grid provides the properties required to generate the Model and related classes. The properties entered here are analogous to the Business View fields.

The grid requires a Field Name and Data Type. The order of the properties are 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.

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

* 1. Step 2: Enter Report Properties



**Note:** This step is displayed for the Report Code Type. If Flat, Process, Dynamic Query or Inquiry are selected as the Code Type, a different step 2 will be displayed.

The Report code type is based upon a Business View. However, the generation of the code files are based upon a Sage 300 report definition from an INI file.

Report definitions for Sage 300 are in INI files. Use the Browse (…) button to display an Open File dialog to search for an existing INI file, or enter the file name manually.

**Note:** If you enter the file name manually, you must press a toolbar button for the contents of the INI file to appear in the Reports list.

The Reports list is populated from the INI file. It contains the report names from the INI file in alphabetical order. The report name selected will be used to populate the Model Fields grid.

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

The Entity Name field takes the place of the Business View description which is defaulted for other Code Type, but may be overridden. Therefore, this field must be entered as there is no source to default it from.

The Model Name field is the name used for the Model created by the wizard. This is the model that is assigned data from the report. It may be the same as the Entity Name if the developer chooses (in the Flat, Process and Inquiry code types, the Entity Name and the Model Name as the same). Therefore, this field must be entered as there is no source to default it from.

The Program ID field is retrieved from the INI file and will be used in the Model Mapper class.

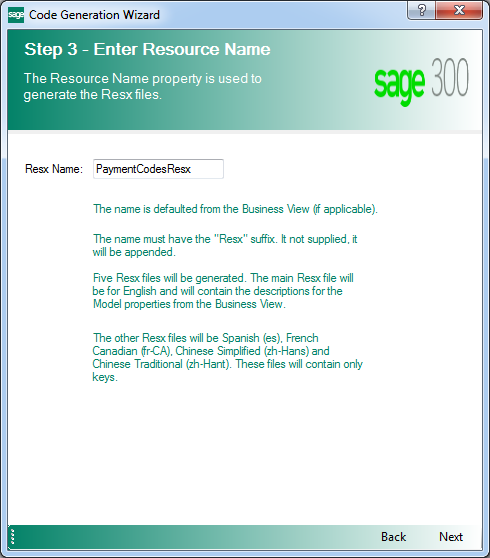
The Model Fields grid provides the properties required to generate the Model and related classes. The properties entered here are analogous to the Business View fields.

* The Field Name column is read-only and is the server field name.
* The Property Name column is the name for the model properties and a simple algorithm has been applied to this column based upon the server field name.
* The Property Name is editable and allows the developer to override the name if desired prior to code generation.
* 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.

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

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

* 1. Step 3: Enter Resource Name

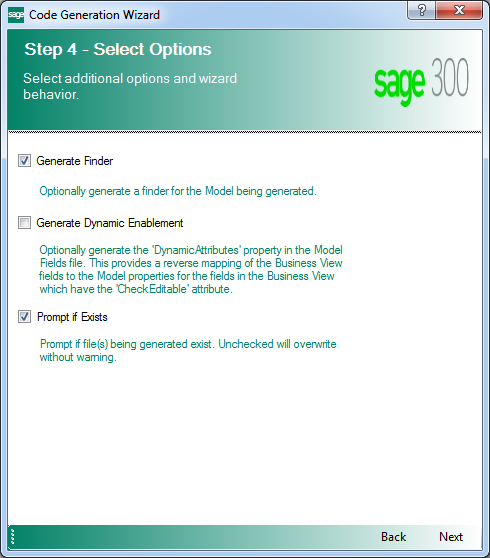


The Resource Name is defaulted from the previous step’s Name field and is appended with “Resx”.

This field may be overridden, but it must have the “Resx” suffix.

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

* 1. Step 4: Select Options



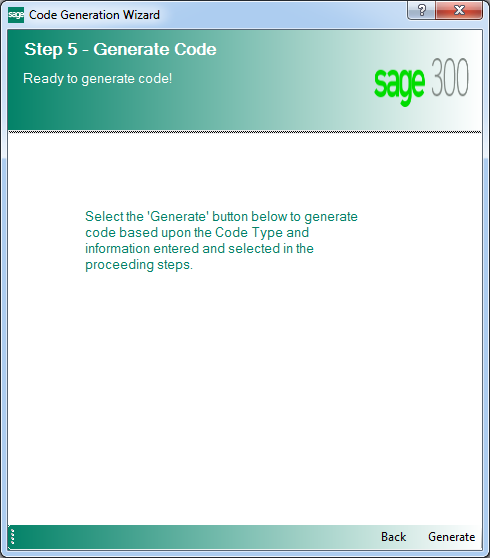
The Generate Finder check box is defaulted to true and will generate the internal controller class for certain Code Types.

The Generate Dynamic Enablement check box is defaulted to false and 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.

The Prompt If Exists check box is defaulted to true and 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.

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

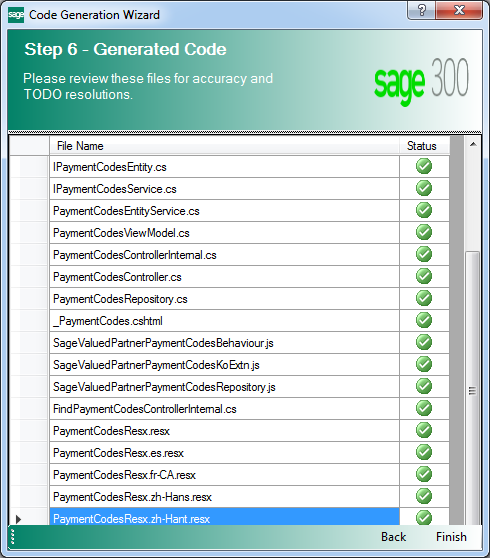
* 1. Step 5: Generate Code



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

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

* 1. Step 6: 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.

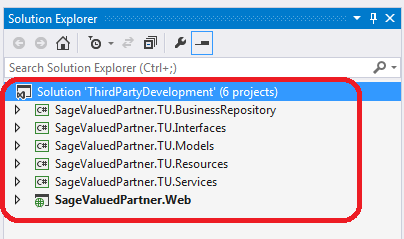
**Note:** TODO issues are C# comments (i.e. // TODO something…) that provide explicit instructions for a developer to resolve or verify wizard generated code.

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

1. Examining the Solution

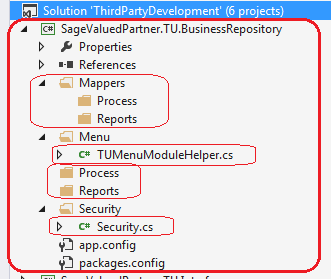
It is time to examine the solution and get familiar with the class and folder scaffolding.

* 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



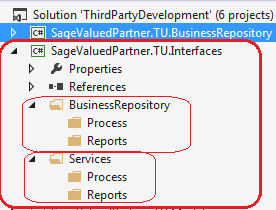
The wizard has created the required folders and a couple of classes that will be used by the Code Generation Wizard.

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.

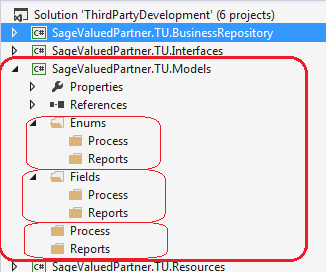
* 1. Interfaces Project



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.

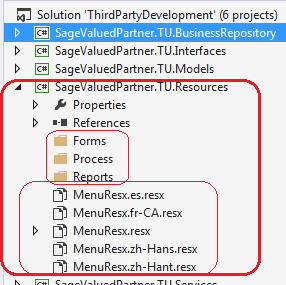
* 1. Models Project



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.

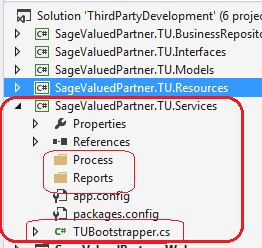
Note the generated Resx files:

* MenuResx.resx is the English resource.
* MenuResx.es.resx is the Spanish resource.
* MenuResx.fr-CA.resx is the French resource.
* MenuResx.zh-Hans.resx is the Chinese Simplified resource.
* MenuResx.zh-Hant.resx is the Chinese Traditional resource.

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

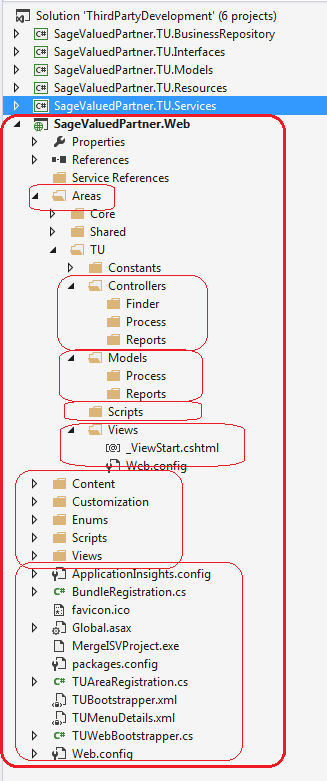


The wizard has created the required folders and a class that will be used by the Code Generation Wizard.

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.