Sage 300 Web Screens SDK

Navigation Menu

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](#_Toc445285788)

[2. Navigation menu components 5](#_Toc445285789)

[3. Sequence of actions 6](#_Toc445285790)

[4. Module helpers 8](#_Toc445285791)

[Appendix: Menu classes diagram 10](#_Toc445285792)

1. Overview

This document provides an overview of how the navigation menu is created in Sage 300 web screens, based on criteria in each module and the associated XML configuration.

1. Navigation menu components

The following components define navigation menu items and their behavior in Sage 300 web screens:

* <module>MenuDetails.xml

Defines what menu items exist in this module and their behavior.

Example: ARMenuDetails.xml

* <module>MenuModuleHelper.cs

Shows or hides menu items based on the associated XML file, and on system conditions such as multicurrency, licensing, and so on.

Example: ARMenuModuleHelper.cs

* MenuModuleHelperManager.cs

Loads module helper files.

* MenuRepository.cs

Uses MenuModuleHelperManager.cs to retrieve all menu module helpers and combine results from each helper to create menu items as data.

* MenuService.cs

Layer to connect the controller and MenuRepository.cs.

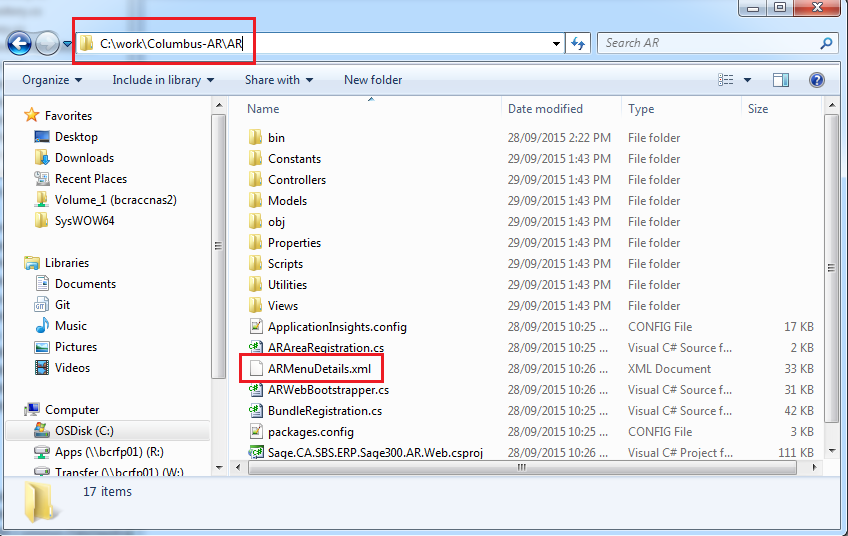
* Sage.CA.SBS.ERP.Sage300.<module>.Web

Project file; copies XML to the App\_Data\MenuDetail folder.

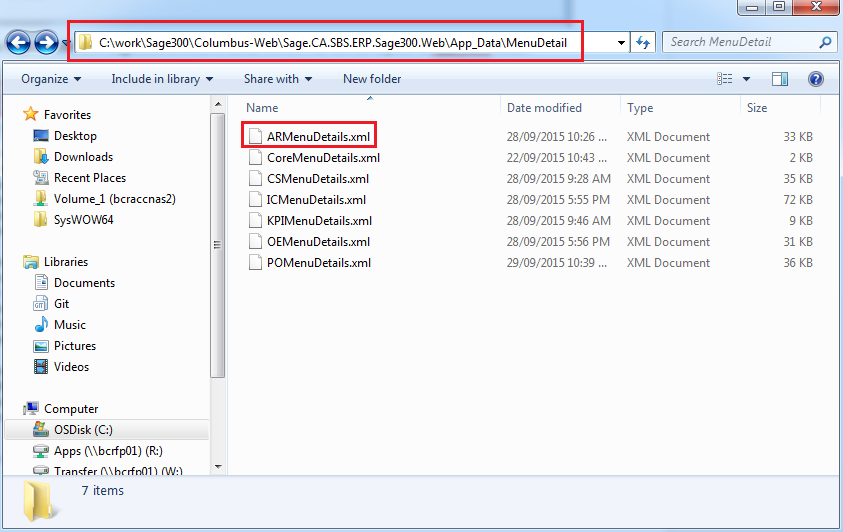
1. Sequence of actions

The Web project for each module should have a post-build event to copy <module>MenuDetails.xml to the MenuDetail folder.

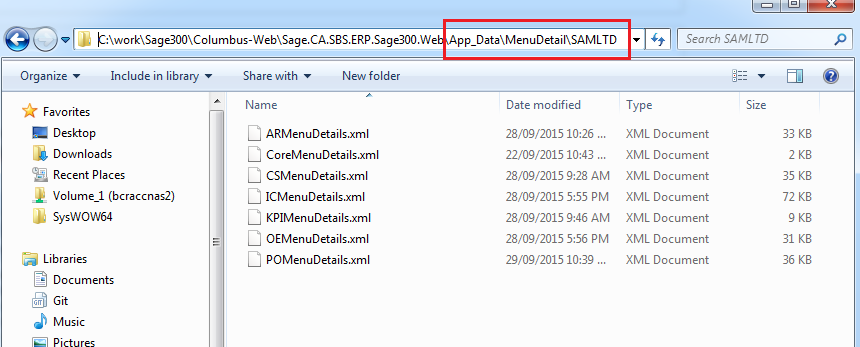
For example, using Accounts Receivable as an example, ARMenuDetails.xml should be located in the AR folder:



After Sage.CA.SBS.ERP.Sage300.AR.Web is built, the XML file should be copied to the App\_Data\MenuDetail folder:



When a user first signs in to the company, the system checks for the company folder in the App\_Data\MenuDetail folder. If the company folder is not found, the system creates the folder and copies the XML files into it.



1. Module helpers

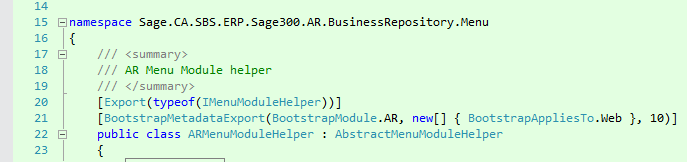
Each module helper makes use of MEF (Managed Extensibility Framework). When the system starts to run, MenuModuleHelperMangers.cs gathers the helpers and prepares them to be consumed by MenuRepository.cs.

Each module helper must have the following two attributes:

* Export specifies the type of export
* BootstrapMetadataExport defines the bootstrap module.

Each module helper must also extend from the AbstractMenuModuleHelper.cs base class so that all base functions are shared among all other helpers.

For example, in Accounts Receivable, ARMenuModuleHelper.cs is defined as follows:



For each module helper, two properties and two functions need to be overridden, as follows:

* Module property returns type of BootstrapModule.
* MenuDetailFileName property returns the name of the associated XML file.
* Initialize function is called each time the module is loaded.
* GetFilteredMenuItems function provides filter criteria depends on the needs of the module.

Here is the example for A/R:



* Initialize()

Inside this function, PrepareDataFile() needs to be called to load data from the XML file. At this point, it is a good idea to gather information about the system that could be used later to determinate if screens should be available or not.

* GetFilteredMenuItems()

Inside this function, return GetApplyFilterMenuItems() needs to be called with a criteriaList that defines which screens should be filtered out based on the current system conditions, such as optional field flags, multicurrency, and so on.

Appendix: Menu classes diagram

This diagram illustrates how menu classes are related.

