Sage 300 Web Screens SDK

Developing a New Application

June 2019

The MIT License (MIT)

Copyright © 2019 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](#_Toc9420936)

[2. Bootstrap configuration files 5](#_Toc9420937)

[3. Menu configuration files 7](#_Toc9420938)

[4. JavaScript and View files 9](#_Toc9420939)

[5. Compiling and testing the developed application 10](#_Toc9420940)

[5.1 Compile views 10](#_Toc9420941)

[5.2 Copy files to the Sage 300 web directory 10](#_Toc9420942)

[5.3 Compile and test the solution 10](#_Toc9420943)

1. Overview

This guide covers steps to follow for third-party developers creating new applications for Sage 300 web screens:

* Define Bootstrap configuration files
* Define menu configuration files
* Define JavaScript and View files
* Compile and test the developed solution

1. Bootstrap configuration files

To enable the application to add, register, and resolve types in the Unity container, you will need to define the Bootstrap configuration file Bootstrapper.xml in the web directory.

The configuration file should contains the assemblies that contains classes that define Bootstrap. The file also defines the menu localization resource file assembly and name.

The file format should be as follows:

<bootstrapper>

<assemblies>

<add assembly="{assembly name}.dll" />

…

</assemblies>

<menuresx>

<add assembly="{menu resource assembly name}" name="{menu resource name}" />

</menuresx>

</bootstrapper>

For example, in the demo project, Bootstrapper.xml is defined as follows:

<bootstrapper>

<assemblies>

<add assembly="TPA.TU.BusinessRepository.dll" />

<add assembly="TPA.TU.Services.dll" />

<add assembly="TPA.Web.dll" />

</assemblies>

<menuresx>

<add assembly="TPA.TU.Resources" name="MenuResx" />

</menuresx>

</bootstrapper>

Or the file can use a wildcard character to represent the assembly:

<bootstrapper>

<assemblies>

<add assembly="TPA\*.dll" />

</assemblies>

<menuresx>

<add assembly="TPA.TU.Resources" name="MenuResx" />

</menuresx>

</bootstrapper>

Important! The menu resource file code should be generated with a public access modifier. It should not use wildcard characters (\*.\*), which will cause errors and be ignored by the Sage 300 Bootstrap system.

1. Menu configuration files

To hook up the plugin menu to the Sage 300 navigation menus, you will need to define the menu configuration files and copy them to the Sage 300 web directory App\_Data\MenuDetail and subdirectories.

The menu localization files are defined in resource assembly menuResx.resx and language resource files. This file assembly and name should also be defined in the bootstrap.xml file for menu plugins and localization.

In the demo project, this is defined in TPA.TU.Resources MenuResx.resx and related localization language files.

Typically, the content for each menu item is defined as an XML element, as follows:

<item>

<MenuID>TU4001</MenuID> //menu id definition

<MenuName>TU Account Set 1</MenuName> // menu name

<ResourceKey>TU\_Account\_Set\_1</ResourceKey> // resource id for menu caption.

<ParentMenuID>TU3000</ParentMenuID> // parent menu menuId

<IsGroupHeader>false</IsGroupHeader> // whether the menu is group header

<ScreenURL>TU/AccountGroup</ScreenURL> //screen url

<MenuItemLevel>4</MenuItemLevel> // menu level, see below menu image

<MenuItemOrder>2</MenuItemOrder> // menu order, see below menu image

<ColOrder>1</ColOrder>

<SecurityResourceKey>TUAccountSet1</SecurityResourceKey>

<IsReport>false</IsReport> // whether the menu is report menu

<IsActive>true</IsActive>

<IsGroupEnd>false</IsGroupEnd>

<IsWidget>false</IsWidget> // whether is widget menu

<Isintelligence>false</Isintelligence> //whether is used for Sage intelligence

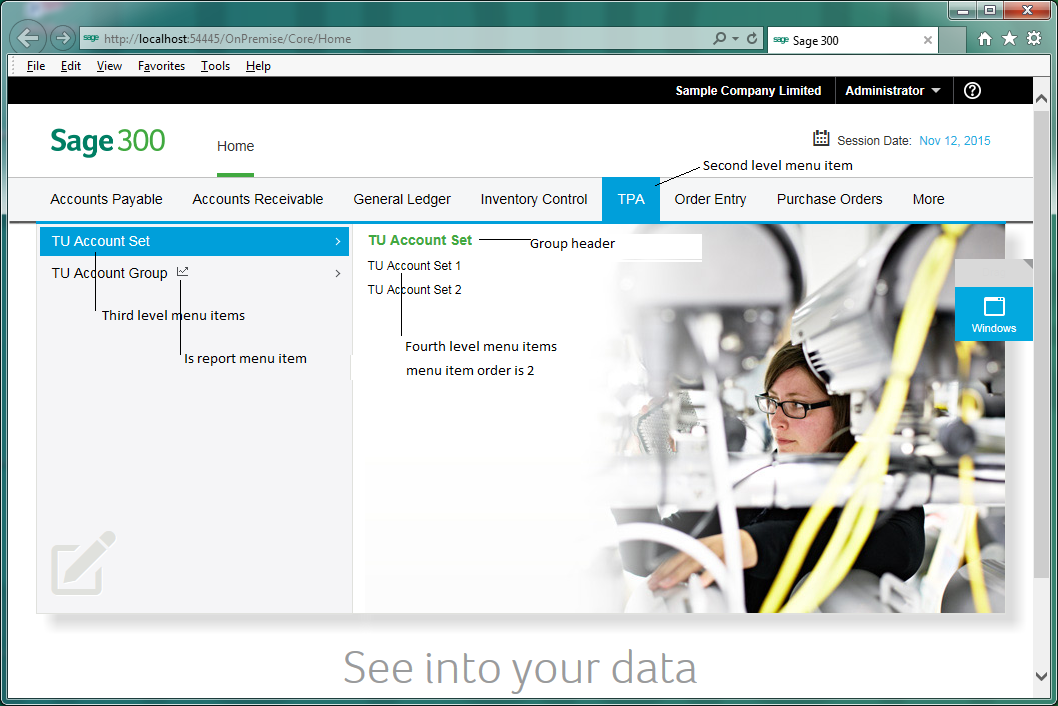
<ModuleName>TU</ModuleName> //module name

</item>

Multiple Security Resource Keys may be specified where a hyphen (-) is “or” and a comma (,) is “and”. For example:

<SecurityResourceKey>ICINQRY-ICSTOCKI,ICCOST-ICTRANSI</SecurityResourceKey>

Which reads: ICINQRY || ICSTOCK && ICCOST || ICTRANSI



For menu configuration details, see *Sage 300 Web Screens SDK – Navigation Menu*, or review TUMenuDetails.xml in the demo project.

1. JavaScript and View files

For screens to appear in Sage 300 web screens, you will need to copy these files and directories to the Sage 300 web area directory.

Copy all files and directories from these locations:

Web\Areas\{area}\Scripts

Web\Areas\{area}\Views

to

Sage 300 online web\Areas\{area}\Scripts

Sage 300 online web\Areas\{area}\Views

For example, in the demo project, copy all files and directories from these locations:

{Web project directory}\Areas\TU\Scripts\\*.\*

{Web project directory}\Areas\TU\Views\\*.\*

to

{Sage 300ERP Online Web}\Areas\TU\Scripts\\*.\*

{Sage 300ERP Online Web}\Areas\TU\Views\\*.\*

1. Compiling and testing the developed application

For the developed application to function in Sage 300 web screens, you will need to copy the Bootstrap, menu configuration, area JavaScript, compiled view files, app\_web\_\*.dlls, and assemblies to the Sage 300 web directory.

Note: To improve application performance, compile the Razor views and copy all compiled files to the Sage 300 bin directory.

* 1. Compile views

To compile the views, use the aspnet\_compiler.exe tool.

* 1. Copy files to the Sage 300 web directory

To copy files to the web directory:

1. Copy the module Bootstrapper.xml file to the Sage 300 web directory.
2. Copy the {Web project directory}\MenuDetails.xml file to the Sage 300 web App\_Data\MenuDetail and subdirectories directory.
3. Copy the {Web project directory}\Areas\{area}\Javascript js files to the Sage 300 web directory in \Areas\{area}\Scripts.
4. Copy the Razor view compiled view files to the Sage 300 web directory in web\Areas\{area}\Views.
5. Copy the Razor view compiled files(\*.compiled, App\_Web\_\*.dll) to the Sage 300 web bin directory.
6. Copy compiled assembly files to the Sage 300 web bin directory.
   1. Compile and test the solution

In the wizard-generated web project, the utility MergeISVProject.exe is added to the web project as a post-build event.

When you compile the solution projects in release mode, all views are compiled automatically and files are copied to Sage 300 directories to hook up the developed application.

To compile and test the solution:

1. Check that that {ModuleId}MenuDetails.xml is configured correctly for the solution.

The menu item SecurityResourceKey should be consistent with the registered resource ID in the desktop module application.

1. Change the application solution to release mode and build the solution.

This compiles views and copies all the necessary files to Sage 300 web screens directories for integration.

1. Open the Sage 300 desktop application, and in Administrative Services, use the Data Activation screen to activate your application.
2. Open a browser and sign in to Sage 300 web screens.

The module menu for your application now appears on the navigation menu.