Sage 300 Web Screens SDK

Import/Export Support

July 2017

The MIT License (MIT)

Copyright © 2017 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](#_Toc488315118)

[2. Implementation 6](#_Toc488315119)

[2.1 Service Inheritance 6](#_Toc488315120)

[2.2 Business Repository Inheritance 6](#_Toc488315121)

[2.3 Overrides 6](#_Toc488315122)

[2.3.1 GetExportImportBusinessEntityProperty 6](#_Toc488315123)

[2.3.2 GetImportTypes (Optional) 7](#_Toc488315124)

[2.3.3 GetExportImportOptions (Optional) 8](#_Toc488315125)

[2.3.4 SetExportImportAdditionalProperties (Optional) 8](#_Toc488315126)

[2.4 WebBootstrapper Registration 9](#_Toc488315127)

[2.5 Setup - Event Handler 9](#_Toc488315128)

1. Overview

Prior to Sage 300 2018, every web screen implemented its own logic to support import/export functionality. This was inefficient and error prone due to copious amounts of code duplication.

In contrast, the current desktop implementation for import/export covers all sorts of scenarios, i.e., batch, header-detail (sequence-revision, ordered-revision), flat view, optional views, etc. And the implementation handles both the business logic and the presentation. When import/export functionality is added to a screen, only the menu item and event handler are required to be coded as seen below:

Dim ie As ImportExport

Set ie = New ImportExport

With ie

' Open the import/export engine.

.Open mDBLinkCmpRW

' Set the views into which data will be imported

.SetView APIBC.ROTOID, "", VIEW\_BATCH, Null

.SetView APIBH.ROTOID, APIBC.ROTOID, VIEW\_SEQUENCED\_HEADER, Null

.SetView APIBD.ROTOID, APIBH.ROTOID, VIEW\_DETAIL\_SEQUENCED, Null

.SetView APIBS.ROTOID, APIBH.ROTOID, VIEW\_DETAIL\_SEQUENCED, Null

If mbHaveOptFldLicense Then

.SetViewEx APIBHO.ROTOID, APIBH.ROTOID, VIEW\_DETAIL\_ORDERED, Null, OPTIONAL\_VIEW

.SetViewEx APIBDO.ROTOID, APIBD.ROTOID, VIEW\_DETAIL\_ORDERED, Null, OPTIONAL\_VIEW

End If

.SetBatchNumber adsAPIBC.Fields.FieldByID(APIBC.IDX\_CNTBTCH).Value

.ImportAction = IMPORT\_INSERT

.VerifyOnPut = True

' Show the "Import" dialog box, then

' close the import/export engine.

.Import

.Close

'import can change the database and so IBC needs a read

adsAPIBC.Read

End With ' ie

Therefore, for the Sage 300 2018 Web Screens, the import/export functionality has been re-factored into an engine comparable to the desktop implementation. As in the desktop, most of the logic will already be in the framework and thus the application developer only needs to supply the appropriate configurations to make use of the engine.

This document will guide the reader on how to add import/export functionality to a web screen utilizing the new framework.

1. Implementation
   1. Service Inheritance

Most of the base service classes already inherit from **BaseExportImportService**.

However, if your service does not, then you must either inherit BaseExportImportService or inherit from other types of service classes that already do (i.e. FlatService, BaseOrderedHeaderDetailService, etc.).

* 1. Business Repository Inheritance

All business repositories already inherit from **BaseExportImportRepository**.

See the below sections for sample overrides where you will need to provide the business logic specific to your screen.

* 1. Overrides
     1. GetExportImportBusinessEntityProperty

Example:

public virtual BusinessEntityProperty GetExportImportBusinessEntityProperty(string option = null, bool isExport = false)

{

var batch = new BusinessEntityProperty(PaymentAdjustmentBatch.EntityName, ViewKeyType.SystemGenerated);

var header = new BusinessEntityProperty(PaymentAdjustment.EntityName, ViewKeyType.SystemGenerated);

var detail = new BusinessEntityProperty(AppliedPayment.EntityName, ViewKeyType.SystemGenerated);

var detailAdjustment = new BusinessEntityProperty(AdjustmentGLDistribution.EntityName, ViewKeyType.SystemGenerated);

var optionalFields = new BusinessEntityProperty(PaymentAdjustmentOptionalField.EntityName, ViewKeyType.UserSpecified);

batch.AddDetail(header);

header.AddDetail(detail);

header.AddDetail(optionalFields);

detail.AddDetail(detailAdjustment);

return batch;

}

You must supply a BusinessEntityProperty or a hierarchy of BusinessEntityProperty which represents the header/detail composition in the Accpac Business Views.

The following are the properties for a BusinessEntityProperty:

* ViewName
  + Located in the {model}.EntityName property
* KeyType
  + Accpac ViewKeyType for which there are 2 types:
    - SystemGenerated – Any view that generate keys internally (i.e. entry number, uniquifier, detail line number, etc.)
    - UserSpecified – Header, flat views, or detail views with ordered revision that requires the user to specify the key
* ComposeOnly
  + Compose views only and will not need import
* IsSystemLink
  + For importing into system tables
* NeedPostForImport
  + Set this property to **true** when you need to call the viewPost function to commit the changes to the database (required for several Accpac views)
    1. GetImportTypes (Optional)

By default, the import types for the base class is Insert/Update/InsertUpdate. This may be overridden in your class.

The following example would be limited to insert only:

public override IEnumerable<ImportType> GetImportTypes(string option = null)

{

return new List<ImportType>{ImportType.Insert};

}

The following is the list of available import types:

* Insert
  + The import will skip the existing records if the entity key type is UserSpecified
* Update
  + Normally for the UserSpecified key type, the import will update the existing record and/or details only if details exist
* InsertUpdate
  + Combination of Insert and Update
* Replace
  + Normally for sequenced details, same as update for the header view, deletes all detail records
* InsertReplace
  + Combination of Insert and Replace
    1. GetExportImportOptions (Optional)

By default, the import/export options for the base class is an empty list. In some exceptional cases, you may want to override it.

For example, in A/P Vendors, you have the choice to import into either Vendors, Vendor Statistics or Vendor Comments:

public override IEnumerable<CustomSelectList> GetExportImportOptions(bool isExport = false)

{

return EnumUtility.GetItemsList<VendorExportOptions>();

}

* + 1. SetExportImportAdditionalProperties (Optional)

By default, the import/export additional properties for the base class is null. In some exceptional cases, we need to do some additional handling. We can use this function to pass in additional parameters to the import/export engine:

public override void SetExportImportAdditionalProperties(dynamic additionalProperties, string option = null, bool isExport = false)

{

//Set outer transactions in case some Accpac views rely on external transaction capability to commit changes to database.

additionalProperties.RequireTransactionForImport = true;

//Set verify on put to true in case some Accpac views need to report validation error upon put.

additionalProperties.VerifyOnPut = true;

}

* 1. WebBootstrapper Registration

See the below example for the **ExportImportController**

Example:

BaseExportImportControllerInternal<EmailMessage, Interfaces.Services.IEmailMessageService<EmailMessage>>>(container, Constants.Constants.EmailMessagesExportImport, new InjectionConstructor(typeof(Context)));

* 1. Setup - Event Handler

See the below example for the **setImportEvent**

Example:

getKey: function () {

var key = [];

key[0] = "AD";

key[1] = $("#Data\_BatchNumber").val();

return key;

},

sg.importHelper.setImportEvent("btnOptionImport", “apadjustmententry”, false, getKey);

sg.importHelper.setExportEvent("btnOptionExport", “apadjustmententry”, false, getKey);

Parameters for setImportEvent():

* Button id triggering the import action
* Import name to match what was registered in the ExportImportController
* Has import/export options
* The keys to locate the record (only for batch type)
* Callback function – i.e., you may want to refresh the UI after import