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](#_Toc487709522)

[2. Implement Import/Export for a Web Screen 6](#_Toc487709523)

[2.1 Subclass BaseExportImportService in your service 6](#_Toc487709524)

[2.2 Subclass BaseExportImportRepository in your business repository 6](#_Toc487709525)

[2.2.1 Override GetExportImportBusinessEntityProperty() 6](#_Toc487709526)

[2.2.2 Optionally override GetImportTypes() 7](#_Toc487709527)

[2.2.3 Optionally override GetExportImportOptions() 7](#_Toc487709528)

[2.2.4 Optionally override SetExportImportAdditionalProperties() 8](#_Toc487709529)

[2.3 Register the ExportImportController in web bootstrapper 8](#_Toc487709530)

[2.4 Setup the import/export event handler 8](#_Toc487709531)

1. Overview

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

The current desktop implementation for import/export covers all sorts of situations, i.e., batch, header-detail (sequence-revisioned, ordered-revisioned), flat view, optional views, etc. And the engine will handle both the business logic and the presentation. When you want to add import/export support to a screen, all you need to do is add the menu item and event handler as can be 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

For the Sage 300 2018 Web Screens, we have implemented the import/export engine in a comparable fashion to our desktop implementation. As in the desktop, most of the logic will already be in the framework and so the application programmer 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 with the new framework.

1. Implement Import/Export for a Web Screen
   1. Subclass BaseExportImportService in your service

Most of the base service classes already subclass from BaseExportImportService. However, if your service does not, then you must either subclass BaseExportImportService or other types of service classes that already do, i.e., FlatService, BaseOrderedHeaderDetailService, etc.

* 1. Subclass BaseExportImportRepository in your business repository

You will need to override a few things.

* + 1. Override GetExportImportBusinessEntityProperty()

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 kind of composition in Accpac Business Views. These are the properties for a BusinessEntityProperty:

* ViewName – Accpac View Roto id
* KeyType – there are 2 types
  + SystemGenerated – any views that generate keys (i.e., entry number, uniquifier, detail line number, etc) internally
  + UserSpecified – header, flat views, or detail views with ordered revision that requires the user to specify the key
* ComposeOnly – for when we only need the views composed but do not need to import data into it
* IsSystemLink – for importing into system data tables
* NeedPostForImport – set this property to true when you need to call the viewPost function in order to commit the changes to the database (required for several Accpac views)
  + 1. Optionally override GetImportTypes()

By default, the import types for the base class is Insert/Update/InsertUpdate. You may want to override this in your subclass. For example, for batch type transactions, we would want to limit it 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 – import will skip the existing records if the entity KeyType is UserSpecified
* Update – normally for UserSpecified key type, just updates the existing record and updates detail only if detail exists
* 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. Optionally override GetExportImportOptions()

By default, the import/export options for the base class is an empty list. In some special 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. Optionally override SetExportImportAdditionalProperties()

By default, the import/export additional properties for the base class is null. In some special 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. Register the ExportImportController in web bootstrapper

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

* 1. Setup the import/export event handler

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 for triggering the import action
* Import name to match what we 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 batch after import