## Adding Excel Export (Basic): TypeScript

I have found Serenity as a huge help and wanted to return the favor with the little snippets of re-usable code that doesn't always get mentioned but that are useful too keep in a "toolbox" of sorts.

Adding an excel export to your table is simpler than you think. ReportHelper and ExcelExportHelper do most of the work for you when setting things up. You just need to add code in 2 places: xyzGrid.ts and xyzEndpoint.cs

In xyzGrid.ts add

getButtons() {

var buttons = super.getButtons();

var MyRow = xyzRow.Fields;

buttons.push(myProject.Common.ExcelExportHelper.createToolButton({

grid: this,

onViewSubmit: () => this.onViewSubmit(),

service: myTableService.baseUrl + '/ListExcel',

separator: true,

hint: "This is the Hint",

title:"This is the Title"

}));

return buttons;

}

Explanation of fields:

* Grid: takes the current view of the grid (if you hid columns using column picker it ignores them)
* onViewSubmit: action being taken when event occurs
* service: Location of list repository (See xyzEndpoint.cs)
* hint: Caption that appears in the button itself.
* title: Caption that appears as label/name of button when you hover

On the xyzEndpoint.cs file, add the following code

using Serenity.Reporting;

using Serenity.Web;

using System;

//.....///

public FileContentResult ListExcel(IDbConnection connection, ListRequest request)

{

var data = List(connection, request).Entities;

var report = new DynamicDataReport(data, request.IncludeColumns, typeof(Columns.xyzColumns));

var bytes = new ReportRepository().Render(report);

var reportName = "xyzList\_";

return ExcelContentResult.Create(bytes, reportName + DateTime.Now.ToString("yyyyMMdd\_HHmmss") + ".xlsx");

}

You'll need to have Serenity.Reporting, Serenity.Web and System to be able to run the code properly. The two fields that you will be needing to update are: reportName and typeof(Columns.xyzColumns).

-reportName: Designates name of the output file. -typeof(Columns.xyzColumns): Defines the source columns.

With that all in place, you are good to go. The tables with this code will display the excel export button on the toolbar. Something worth remembering is that the export will pull only the columns currently active in the view thus if you have 5 columns but you are only displaying 3, you'll only export 3 columns.

The Excel Export is based on EPPLus ( <http://epplus.codeplex.com/>) which allows further editing of the output but I leave that for another time.

## Add or Remove a Button

Need to add or remove a Button? Easy.

**For Grid.ts:**

protected getButtons(): Serenity.ToolButton[] {

var buttons = super.getButtons();

//Remove a Button

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "<Class of Button to remove> Example: 'add-button'"), 1);

//Add a Button

buttons.push({

title: "<title of Button>",

hint: "<Hover hint of button>",

cssClass: "<Css defined class of button> Example: 'add-note-button' ",

icon: "<Icon to use with button> Example: 'fa-hand-lizard-o'",

onClick: () => { /\*Add Function Here \*/ }

})

return buttons;

}

**For Dialog.ts**

protected getToolbarButtons(): Serenity.ToolButton[] {

let buttons = super.getToolbarButtons();

//Remove Button

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "<Class of Button to remove> Example: 'save-and-close-button'"), 1);

//Add a Button

buttons.push({

title: "<title of Button>",

hint: "<Hover hint of button>",

cssClass: "<Css defined class of button> Example: 'add-note-button' ",

icon: "<Icon to use with button> Example: 'fa-hand-lizard-o'",

onClick: () => { /\*Add Function Here \*/ }

})

return buttons;

}

## Adding Active Delete (Delete/Undelete) to a Grid

Being able to add active delete/undelete is simple. Add an "IsActve" Int16 field to the table you wish to use active delete on and then do the following:

**on Row.Cs**

[ConnectionKey("Default"), TableName("[tbl].[xyz]"), DisplayName("Entries"), InstanceName("Entry"), TwoLevelCached]

[ReadPermission("Administration:General"), ModifyPermission("Administration:General")]

public sealed class xyzRow : Row, IIdRow, INameRow, IIsActiveRow, IIsActiveDeletedRow //<---

{

//...

[NotNull, Insertable(false), Updatable(true)]

public Int16? IsActive //<---

{

get { return Fields.IsActive[this]; }

set { Fields.IsActive[this] = value; }

}

Int16Field IIsActiveRow.IsActiveField //<---

{

get { return Fields.IsActive; }

}

//...

}

**On Repository.cs add**

public UndeleteResponse Undelete(IUnitOfWork uow, UndeleteRequest request)

{

return new MyUndeleteHandler().Process(uow, request);

}

private class MyUndeleteHandler : UndeleteRequestHandler<MyRow> { }

**On Endpoint.cs add**

[HttpPost, AuthorizeDelete(typeof(MyRow))]

public UndeleteResponse Undelete(IUnitOfWork uow, UndeleteRequest request)

{

return new MyRepository().Undelete(uow, request);

}

**On Grid.ts add**

namespace FMCorpIntranet.Default {

@Serenity.Decorators.registerClass()

export class EmployeesGrid extends Serenity.EntityGrid<EmployeesRow, any> {

protected getIsActiveProperty() { return EmployeesRow.isActiveProperty; } //<----

protected getColumnsKey() { return 'Default.Employees'; }

//....

}

}