Steps of creating new Entity UI (MVC using the template)

This example based on the code here

<http://positiondescriptions.lsvle.uat.det.nsw.edu.au/Directorate>

# Step 1 basics:

## Classes and code.

1. Generate the data driven model from template if it applicable like (Status, Type, category,…)
2. Generate the template code.
3. Update the Entity data and light classes with display field names.
4. Update the Light class with the foreign entity display names like

*public System.String DirectorateName { get; set; } // foreign entity display name*

*public System.String BUnitDescription { get; set; }// foreign entity display name*

*public System.String StatusName { get; set; }// foreign entity display name*

*public System.String ExecutiveTitle { get; set; }// foreign entity display name*

1. Regenerate the template.
2. Exclude the NA entity value from the List() method
3. Ensure and reviews the Includes statements are there ( you may create different Lists() for different includes for purpose of performance)
4. Add GetEntityById in the repository.
5. Update FilterMethod in the repository
6. For automatic generated keys use this.GetDbNewId("Entityname");
7. Create a unit tests for the repository

## Controller

1. If you are planning to regenerate the controller few times, you may create all new methods in controller\_partial.cs files so it will not be overridden
2. Update all GetByEntityKey to the valid GetByEntityId (or code) points to List() with all includes
3. Add trailing codes in Edit & Create with post methods

*var ajaxResult = new Result();*

*var trailingList = directorate.UpdateSignature(this.Repository.RepositoryFactory);*

*this.ModelState.RemoveList(trailingList);*

*ClearModelState();*

1. Create the Create lookup and filter status fields based on the use.
2. Create() create a model and initialise the model, before you pass it to the View like

var model = new BusinessUnit

{

BUnitId = Repository.GetMaxKeyValue() + 10

};

if (Request.IsAjaxRequest()) {

return View("Create-modal", model);

} else {

return View(model);

}

## Views

1. Make the Entity/index page Ajax
2. Choose the columns that you display in \_partial/\_list
3. Update the columns foreign key attributes to match the sort ( like Enity.Parent.Title)
4. Delete extra fields in \_partial/\_details
5. Delete extra fields in \_partial/\_updates (especially trailing fields)
6. Update any text area you believe it should be simple input.
7. Update any button name or heading names looks odd.

# Step 2 Security:

## Classes and code.

1. Add at least 3 methods grouped in IUserTask as.

Enums.Privilege GetEntityPrivilege();

Enums.Privilege GetEntityPrivilege (Entity bUnit);

Enums.Privilege GetEntityPrivilege (int id);

1. Create a unit tests for these methods.
2. Add privilege class to the light

## Controller

1. Add Security attributes in all actions and Controller level
2. Populate privilege class to the ListJson method()

## Views

1. Index add security condition for Create new Entity
2. Service JavaScript add conditions to the Add/Edit and delete buttons

# Step 3 Advanced

1. Add UI custom lookup and filters to the ajax list page.
2. Add more lookups to Create and Edit popup forms.
3. Add more custom validation.
4. Update the UI for accessibility.

Develop the specific features for each entity

# Step 4 basic accessibility

## Detail screens

1. <label tabindex=0 aria-label=”value”>value</label>
   1. Tab index for keyboard tabbing
   2. Aria-label for the screen reader the value in most of browsers (chrome need this attribute)
2. To implement the basic accessibility
   1. updateAccessibility() in interactive.js will be applied for all these elemnts
      1. Forms contains
      2. label
      3. p has form-control-static as class

# Step 5 Creating Filtered lists with advanced filter and in Edit / Create screens

## Controller code

1. In public ActionResult ListJson([FromUri] DirectorateArgument arg)  
   : Strong type the argument with all filtered parameters In CreateLookups() : Add all filter lists In case of Edit or Create you may filter some lists as initial value based on the parent lists
2. In Create method add parameters to set the initial values of the form like public ActionResult Create(string divisionCode, int? directorateId)

## 

## Index page

1. Add the filter code in expandable use filterColumnnameId to avoid any collision with a form column name

## Java script

in the service script add these code

1. initialise: function () {

loadFilterLookups();

loadUpdateFilterLookups();

},

1. For the datatable add 'fnServerParams' option

'fnServerParams': function (aoData) {

aoData.push({

name: "divisionCode", value: $('#filterDivisionCode').val()

}, {

name: "directorateId", value: $('#filterDirectorateId').val()

}

);

1. Add these similar code

function hookPopup() {

if ($('.pop-up').length > 0) {

interactive.hookPopupButton(

{} // custom validation settings

,function(){ // modal load callback function

interactive.basicInitialisation('#tempModal');

loadUpdateFilterLookups();

}

,true // use Ajax Submit Handler

,function(data){ // success callback function

$('.error-popup').hide();

$('#formTab').hide();

$('#lbl-success')

.text('Data has been successfully updated');

$('.success-popup').show(500);

$('.table').dataTable().fnDraw();

$('.table').dataTable().fnDraw();

}

,function(data){ // error callback function

}

,'.pop-up' // element selector

,true // auto hide

);

}

}

1. Update the create link with the new query string like this

So when you create a new object it will take the global filter

function loadFilterLookups() {

var selectDir = $('#filterDirectorateId');

$('#filterDivisionCode').on('change', function () {

helper.setSelectEmpty(selectDir);

setDropdwnOptions(selectDir, cnt.GetDirectorates, { divisionCode: $('#filterDivisionCode').val(), displayNumbers: true });

$('.table').dataTable().fnDraw();

var link = $('#create-function-area');

link.attr('href', appUrl + 'FunctionalArea/Create?divisionCode=' + $('#filterDivisionCode').val());

});

1. Sample of Filter methods

function loadUpdateFilterLookups() {

var selectDir = $('#DirectorateId');

var divisionSel = $('#DivisionCode');

divisionSel.on('change', function () {

helper.setSelectEmpty(selectDir);

setDropdwnOptions(selectDir, cnt.GetDirectorates, { divisionCode: divisionSel.val(), displayNumbers: true });

});

}

function setDropdwnOptions(elem, action, data) {

api.ajaxGet(action, data,

function (items) {

$.each(items, function (id) {

elem.append($('<option/>', {

value: items[id].Value,

text: items[id].Text

}));

});

});

elem.select2("val", "0");

}