***[Development Passion](https://developmentpassion.blogspot.com/)***

Things i have experienced (C#, asp.net, linq,mvc 4, jquery and much more)

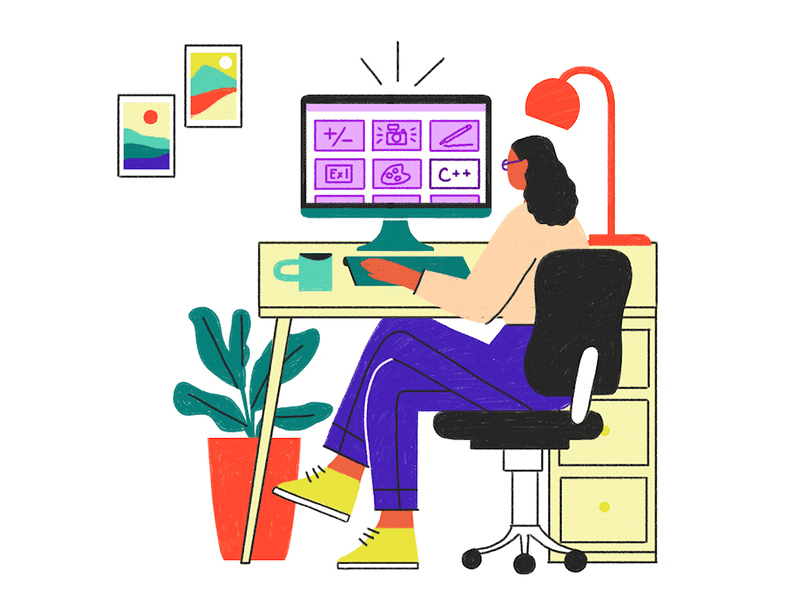
* [**HOME**](http://developmentpassion.blogspot.com/)
* [**C# .NET**](http://developmentpassion.blogspot.com/p/c-net.html)
* [**CODE SNIPPETS**](https://developmentpassion.blogspot.com/2017/06/ajax-crud-in-grid-using-jquery.html)
* [**JAVASCRIPT & JQUERY**](http://developmentpassion.blogspot.com/p/jquery.html)
* [**ASP.NET MVC**](http://developmentpassion.blogspot.com/p/aspnet-mvc.html)
* [**OOP (OBJECT ORIENTED PROGRAMMING)**](http://developmentpassion.blogspot.com/p/oop-object-oriented-programming.html)

Top of Form



Bottom of Form

[Development Passion](https://developmentpassion.blogspot.com/) [ajax](https://developmentpassion.blogspot.com/search/label/ajax?&max-results=6)  [asp.net mvc](https://developmentpassion.blogspot.com/search/label/asp.net%20mvc?&max-results=6)  [CRUD](https://developmentpassion.blogspot.com/search/label/CRUD?&max-results=6)  [grid](https://developmentpassion.blogspot.com/search/label/grid?&max-results=6)  [jquery datatables](https://developmentpassion.blogspot.com/search/label/jquery%20datatables?&max-results=6)  AJAX CRUD in Grid using JQuery DataTables in ASP.NET MVC 5

[](http://click.linksynergy.com/fs-bin/click?id=Twqblj5P7jA&offerid=323058.247&subid=0&type=4)

***AJAX CRUD in Grid using JQuery DataTables in ASP.NET MVC 5***

 Sunday, June 11, 2017  by [Ehsan Sajjad](https://www.blogger.com/profile/17686377274911040657) 0 Comments [A+](javascript:addsize()) [a-](javascript:deletesize())

Introduction

I frequently come across questions on online forums like **StackOverflow**in which the questioners are able to create CRUD operations for their entity which is a normal form post, but they struggle implementing the CRUD via ajax so that page does not reloads completely for better User Experience.

In this post, I will be addressing this problem and we will learn that how can implement CRUD using **JQuery DataTables** in one page without any full page reloads.

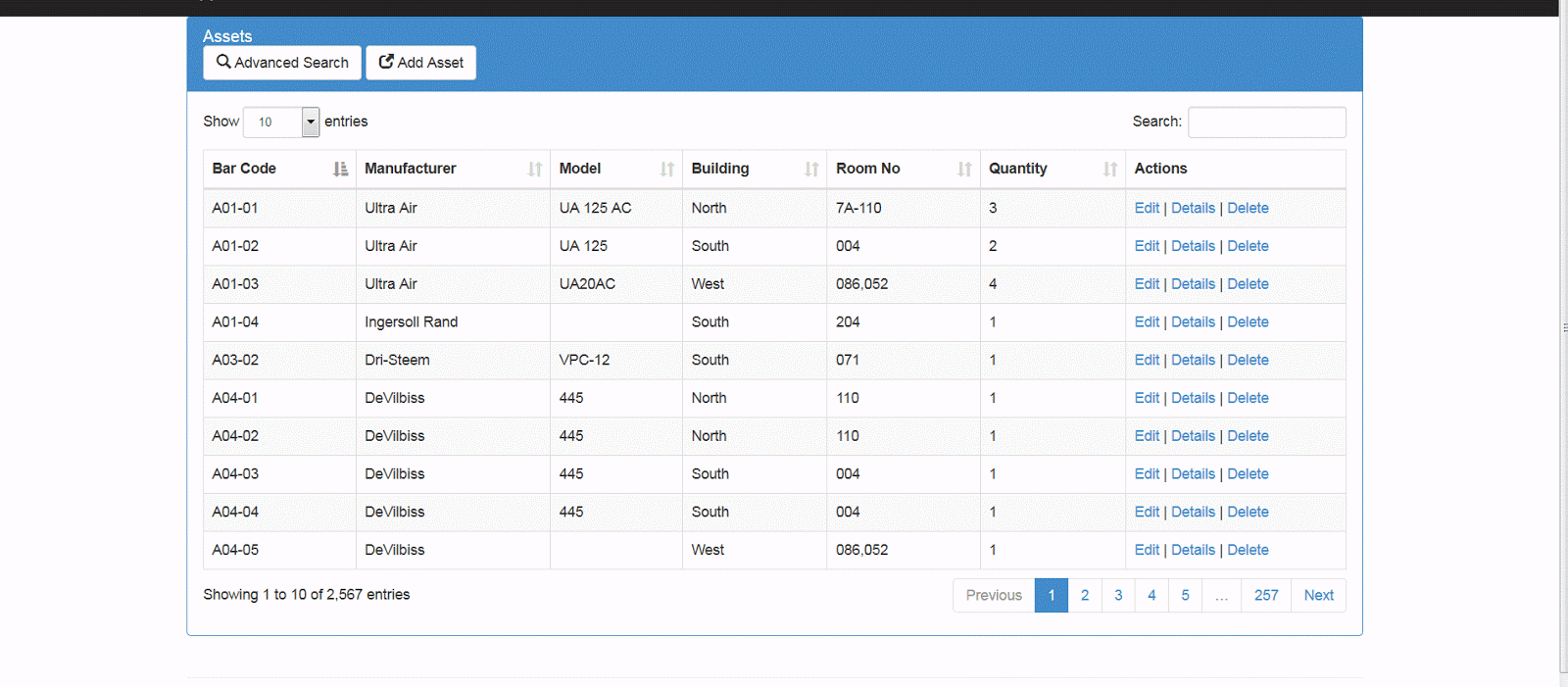
Background

I had been writing in last few posts about the usage of ***JQuery DataTables*** and it can be leveraged to build a ***GridView***like functionality in asp.net mvc, we saw how we can install it using ***NuGet Package Manager*** and implement a simple gird using the plugin which provides us the essential features like Sorting, Searching and Pagination, then we saw how we can implement the paging, filtering and Ordering, then we proceeded to see how advanced search can be added for more better user experience.

If someone wants to have refresher or interested to read the previous posts related, they can be found here:

* [Beginners Guide for Creating GridView in ASP.NET MVC 5](http://developmentpassion.blogspot.nl/2016/07/beginners-guide-for-creating-gridview.html)
* [Grid View with Server Side Filtering, Sorting and Paging in ASP.NET MVC 5](http://developmentpassion.blogspot.nl/2016/08/grid-view-with-server-side-filtering.html)
* [Grid View With Server Side Advanced Search Using jQuery DataTables In ASP.NET MVC 5](http://developmentpassion.blogspot.com/2017/02/gridview-with-server-side-advacned.html)

In this post we will learn how we can add create, update and delete operations support in the grid that we already implemented using JQuery DataTables. The end result will look something like:

[](https://4.bp.blogspot.com/-HuQLYObeYQE/WTySGruuqcI/AAAAAAAAEdg/8WwcwzXONSE-rIwpQr27J_l_GisUnGZ5QCEw/s1600/GridAjaxCRUDDemo.gif)

 Database Creation using Script

First of all we will run the script for database and tables creation that would be essential for this post. Following is the script for that:

CREATE DATABASE [AdvancedSearchGridExampleMVC]

GO

CREATE TABLE [dbo].[FacilitySites] ([FacilitySiteID] UNIQUEIDENTIFIER NOT NULL,

[FacilityName] NVARCHAR (MAX) NULL,

[IsActive] BIT NOT NULL,

[CreatedBy] UNIQUEIDENTIFIER NOT NULL,

[CreatedAt] DATETIME NOT NULL,

[ModifiedBy] UNIQUEIDENTIFIER NULL,

[ModifiedAt] DATETIME NULL,

[IsDeleted] BIT NOT NULL

);

GO

CREATE TABLE [dbo].[Assets] (

[AssetID] UNIQUEIDENTIFIER NOT NULL,

[Barcode] NVARCHAR (MAX) NULL,

[SerialNumber] NVARCHAR (MAX) NULL,

[PMGuide] NVARCHAR (MAX) NULL,

[AstID] NVARCHAR (MAX) NOT NULL,

[ChildAsset] NVARCHAR (MAX) NULL,

[GeneralAssetDescription] NVARCHAR (MAX) NULL,

[SecondaryAssetDescription] NVARCHAR (MAX) NULL,

[Quantity] INT NOT NULL,

[Manufacturer] NVARCHAR (MAX) NULL,

[ModelNumber] NVARCHAR (MAX) NULL,

[Building] NVARCHAR (MAX) NULL,

[Floor] NVARCHAR (MAX) NULL,

[Corridor] NVARCHAR (MAX) NULL,

[RoomNo] NVARCHAR (MAX) NULL,

[MERNo] NVARCHAR (MAX) NULL,

[EquipSystem] NVARCHAR (MAX) NULL,

[Comments] NVARCHAR (MAX) NULL,

[Issued] BIT NOT NULL,

[FacilitySiteID] UNIQUEIDENTIFIER NOT NULL

);

GO

CREATE NONCLUSTERED INDEX [IX\_FacilitySiteID]

ON [dbo].[Assets]([FacilitySiteID] ASC);

GO

ALTER TABLE [dbo].[Assets]

ADD CONSTRAINT [PK\_dbo.Assets] PRIMARY KEY CLUSTERED ([AssetID] ASC);

GO

ALTER TABLE [dbo].[Assets]

ADD CONSTRAINT [FK\_dbo.Assets\_dbo.FacilitySites\_FacilitySiteID] FOREIGN KEY ([FacilitySiteID]) \_

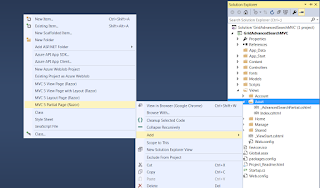
REFERENCES [dbo].[FacilitySites] ([FacilitySiteID]) ON DELETE CASCADE;

GO

You can find the script in attached source code in file name dbScript.sql which also contains sample data to get started quickly, running it will create the DB and will add some sample data in the tables as well.

Create/Insert Operation

For the insertion part, first of all we need to create a partial view in ***Views >> Asset*** by right clicking the ***Asset***folder and navigate to ***Add >> MVC 5 Partial Page (Razor)*** like the below screen shot:

[](https://1.bp.blogspot.com/-8i9oWVaCcps/WTyUs3EhCtI/AAAAAAAAEdo/FZk6WtX6D1gUiKDrIMfBx4nZWBbXZ0r4gCLcB/s1600/Screen1.png)

Open the container view in which asset rows are being rendered which ***\_AssetsPartial.cshtml*** located in ***Views >> Asset*** directory and add the html for the Add Asset button which will open up a popup for inserting a new asset row in database:

<button type="button" class="btn btn-default btn-md" data-toggle="modal" data-url="@Url.Action("Create","Asset")" id="btnCreateAsset">

<span class="glyphicon glyphicon-new-window" aria-hidden="true"></span> Add Asset

</button>

Now, open the Index View of Asset which is  ***Views >> Asset >> index.csthtml*** and in scripts section add the following jQuery code for the button click event which we just created above, it will call the create get action and will display the partial view in bootstrap modal to user, the code for which is:

$("#btnCreateAsset").on("click", function () {

var url = $(this).data("url");

$.get(url, function (data) {

$('#createAssetContainer').html(data);

$('#createAssetModal').modal('show');

});

});

We also need to add bootstrap modal with container div in which we will be loading create partial view of Asset, so we will add the following html at the end of ***Index.cshtml*** view:

<div class="modal fade" id="createAssetModal" tabindex="-1" role="dialog" aria-labelledby="CreateAssetModal" aria-hidden="true" data-backdrop="static">

<div id="createAssetContainer">

</div>

</div>

Implementing Add GET Action

Add action method for Asset Creation get request in the Asset controller file located at ***Controllers >> AssetController.cs*** which would be called by the above jQuery code, inside the action method we are populating the ViewModel and passing the instance back to PartialView as normally we do in mvc based application:

public ActionResult Create()

{

var model = new AssetViewModel();

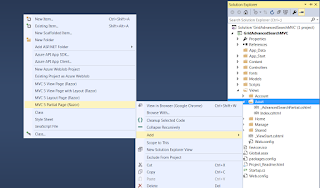
model.FacilitySitesSelectList = GetFacilitiySitesSelectList();

return View("\_CreatePartial", model);

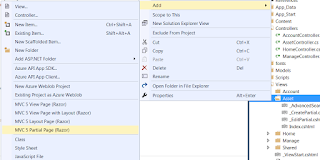
}

Add Partial View Creation:

Add a new Partial view in project in Asset Views for Create form, for that right click the ***Asset***folder under ***Views***in the ***Solution Explorer***:

[](https://3.bp.blogspot.com/-8i9oWVaCcps/WTyUs3EhCtI/AAAAAAAAEdw/bDViK2b_h8oXYZVblJqbC1Im2ChUABf4QCEw/s1600/Screen1.png)

Enter the name of partial view to be ***\_CreatePartial*** in textbox or whatever name you think would be better in your case.

[](https://4.bp.blogspot.com/-1y9cdkKrb1U/WTyXmHPNDtI/AAAAAAAAEd0/ZcWTC6tQaWwI6JZlT8oOCRMRam3hfOR_QCLcB/s1600/Screen2.png)

[](https://1.bp.blogspot.com/-wkLAZEWJznQ/WTyXogkS5WI/AAAAAAAAEd4/fl3p00axYxI9tx578iV0INFpFVV9wljAwCLcB/s1600/Screen3.png)

Implementing Add Partial View

Now we will write the partial view that we created in previous step ***\_CreatePatial***, add the following code in that View:

@model GridAjaxCRUDMVC.Models.AssetViewModel

@{

Layout = null;

}

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title">Add Asset</h4>

</div>

@using (Ajax.BeginForm("Create", "Asset", null, new AjaxOptions { HttpMethod = "Post", OnSuccess = "CreateAssetSuccess" }, new { @class = "form-horizontal", role = "form" }))

{

<div class="modal-body">

<div class="form-horizontal">

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.AssetID, new { Value = Guid.NewGuid() })

<div class="form-group">

@Html.LabelFor(model => model.Barcode, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Barcode, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Barcode, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.SerialNumber, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.SerialNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.SerialNumber, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.FacilitySiteID, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.FacilitySiteID, Model.FacilitySitesSelectList, "Select One", new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FacilitySiteID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.PMGuide, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.PMGuide, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.PMGuide, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.AstID, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.AstID, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.AstID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ChildAsset, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ChildAsset, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ChildAsset, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.GeneralAssetDescription, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.GeneralAssetDescription, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.GeneralAssetDescription, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.SecondaryAssetDescription, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.SecondaryAssetDescription, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.SecondaryAssetDescription, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Quantity, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Manufacturer, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Manufacturer, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Manufacturer, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ModelNumber, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ModelNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ModelNumber, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Building, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Building, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Building, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Floor, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Floor, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Floor, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Corridor, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Corridor, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Corridor, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.RoomNo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.RoomNo, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.RoomNo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.MERNo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.MERNo, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.MERNo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.EquipSystem, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.EquipSystem, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.EquipSystem, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Comments, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Comments, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Comments, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Issued, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

<div class="checkbox">

@Html.EditorFor(model => model.Issued)

@Html.ValidationMessageFor(model => model.Issued, "", new { @class = "text-danger" })

</div>

</div>

</div>

</div>

</div>

<div class="modal-footer">

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

<input type="submit" class="btn btn-primary" value="Save" />

</div>

</div>

</div>

}

</div>

</div>

Implementing Add/Create Post Action

The important thing to note in the view code is the ***Ajax.BeginForm*** helper portion as the class ***Ajax*** should be enough to understand that it will post the model back to controller action via Ajax, which means that whole will not refresh i.e. no full postback will happen in terms on asp.net web forms.

The post action is simply mapping ViewModel object to the Model and then saving it in the repository:

[HttpPost]

public async Task<ActionResult> Create(AssetViewModel assetVM)

{

if (!ModelState.IsValid)

return View("\_CreatePartial", assetVM);

Asset asset = MaptoModel(assetVM);

DbContext.Assets.Add(asset);

var task = DbContext.SaveChangesAsync();

await task;

if (task.Exception != null)

{

ModelState.AddModelError("", "Unable to add the Asset");

return View("\_CreatePartial", assetVM);

}

return Content("success");

}

We are trying to save the newly entered asset in the database and tracking if it is saved successfully or not and if it saves successfully we are returning a string message **“success”** back to client side and in the success call back of Ajax Form we will be checking if operation was successful or not to do UI changes according to that.

Implementing Ajax Callback Function

We can see that in ***BeginForm***helper method parameters we are specifying JavaScript function to be called when the Ajax Form successfully returns back from server i.e. ***OnSuccess = "CreateAssetSuccess"***  
  
Now let’s go to the ***index.cshtml*** view of Asset and define the success callback implementation:

function CreateAssetSuccess(data) {

if (data != "success") {

$('#createAssetContainer').html(data);

return;

}

$('#createAssetModal').modal('hide');

$('#createAssetContainer').html("");

assetListVM.refresh();

}

What we are doing here is if the operation is not successful we are updating the client side html to notify the user about the failure of creation and if the insert goes successful we are closing the modal popup and refreshing the Grid to display the up to date information.

Edit/Update Operation:

Until now we should be able to run the application and successfully add new assets in the database via Ajax using the bootstrap modal which we created above, now let’s move to the update part of the Asset.  
We will be adding another column in the datatable columns collection which will contain hyper link that will navigate to Edit View, but as we are using Ajax and bootstrap modal, we will be doing updates as well same way, so no redirect will be involved. Let’s get started.  
Open the ***Index.csthml*** file and add a new column in the columns collection in jQuery datatable initialization, after updating the columns array our code would look like:

"columns": [

{ "title": "Bar Code", "data": "BarCode", "searchable": true },

{ "title": "Manufacturer", "data": "Manufacturer", "searchable": true },

{ "title": "Model", "data": "ModelNumber", "searchable": true },

{ "title": "Building", "data": "Building", "searchable": true },

{ "title": "Room No", "data": "RoomNo" },

{ "title": "Quantity", "data": "Quantity" },

{

"title": "Actions",

"data": "AssetID",

"searchable": false,

"sortable": false,

"render": function (data, type, full, meta) {

return '<a href="@Url.Action("Edit","Asset")?id=' + data + '" class="editAsset">Edit</a>';

}

}

]

We are setting the header of new column to display Actions as title and we would need to disable the searching and sorting for this column, as it is for edit operation and it does not makes sense to enable sorting and searching on this column. Next we are defining the render method of the column and we are generating anchor link html which could call the Edit action of Asset controller and will pass the current asset id to pull the information of it and display in the Edit View.

Defining Edit/Update GET Action:

After doing the datatable js changes, now we need to create a get action method which will pull the asset record from the database and will display it for editing to the user in a popup.  
Let’s implement the Edit get action of it:

public ActionResult Edit(Guid id)

{

var asset = DbContext.Assets.FirstOrDefault(x => x.AssetID == id);

AssetViewModel assetViewModel = MapToViewModel(asset);

if (Request.IsAjaxRequest())

return PartialView("\_EditPartial",assetViewModel);

return View(assetViewModel);

}

The action simply retrieves the row from the database and after converting it to ViewModel passes it to back to partial view to rendered or returned back to the client side for processing, as no post backs would happen it will generate the html and will send the html back in response of ajax call which client side will handle and decide where to put that html.

Handling Action Links Events on Client Side using JQuery:

From the column render function you can see that there is class applied on anchor link called ***EditAsset***, it is defined because jQuery event handler will be applied to the anchor link and Ajax call will be sent to server, let’s define the event handler for that in Index View:

$('#assets-data-table').on("click", ".editAsset", function (event) {

event.preventDefault();

var url = $(this).attr("href");

$.get(url, function (data) {

$('#editAssetContainer').html(data);

$('#editAssetModal').modal('show');

});

});

Addition of Bootstrap Modal

Now add the bootstrap modal html in the Index View which will be placeholder for loading the Edit View, define it just after create  bootstrap modal html:

<div class="modal fade" id="editAssetModal" tabindex="-1" role="dialog" aria-labelledby="EditAssetModal" aria-hidden="true" data-backdrop="static">

<div id="editAssetContainer">

</div>

</div>

Edit/Update Partial View Creation:

Create another new partial view following the same steps which we did for adding ***\_CreatePartial.cshtml*** which was for Create Asset, So add a new Partial View in the Asset folder in ***Views***with name ***\_EditPartial.cshtml*** and add the following code in it:

@model TA\_UM.ViewModels.AssetViewModel

@{

Layout = null;

}

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title">Edit Asset</h4>

</div>

@using (Ajax.BeginForm("Edit", "Asset", null, new AjaxOptions { HttpMethod="Post", OnSuccess = "UpdateAssetSuccess" }, new { @class = "form-horizontal", role = "form" }))

{

<div class="modal-body">

<div class="form-horizontal">

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

@Html.HiddenFor(model => model.AssetID)

<div class="form-group">

@Html.LabelFor(model => model.Barcode, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Barcode, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Barcode, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.SerialNumber, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.SerialNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.SerialNumber, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.FacilitySiteID, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.DropDownListFor(model => model.FacilitySiteID,Model.FacilitySitesSelectList, "Select One", new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.FacilitySiteID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.PMGuide, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.PMGuide, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.PMGuide, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.AstID, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.AstID, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.AstID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ChildAsset, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ChildAsset, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ChildAsset, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.GeneralAssetDescription, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.GeneralAssetDescription, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.GeneralAssetDescription, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.SecondaryAssetDescription, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.SecondaryAssetDescription, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.SecondaryAssetDescription, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Quantity, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Manufacturer, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Manufacturer, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Manufacturer, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.ModelNumber, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.ModelNumber, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.ModelNumber, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Building, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Building, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Building, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Floor, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Floor, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Floor, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Corridor, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Corridor, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Corridor, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.RoomNo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.RoomNo, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.RoomNo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.MERNo, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.MERNo, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.MERNo, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.EquipSystem, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.EquipSystem, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.EquipSystem, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Comments, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

@Html.EditorFor(model => model.Comments, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Comments, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.Issued, htmlAttributes: new { @class = "control-label col-md-2" })

<div class="col-md-10">

<div class="checkbox">

@Html.EditorFor(model => model.Issued)

@Html.ValidationMessageFor(model => model.Issued, "", new { @class = "text-danger" })

</div>

</div>

</div>

</div>

</div>

<div class="modal-footer">

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<button type="button" class="btn btn-default" data-dismiss="modal">Close</button>

<input type="submit" class="btn btn-primary" value="Save changes"/>

</div>

</div>

</div>

}

</div>

</div>

The Edit View is also pretty same as we had for Insert except it would be posting to different action which would be responsible for handling the updates of a particular Asset row.

Implementing Edit/Update Post Action:

 Now let’s implement the post action of Edit:

[HttpPost]

public async Task<ActionResult> Edit(AssetViewModel assetVM)

{

assetVM.FacilitySitesSelectList = GetFacilitiySitesSelectList(assetVM.FacilitySiteID);

if (!ModelState.IsValid)

{

Response.StatusCode = (int)HttpStatusCode.BadRequest;

return View(Request.IsAjaxRequest() ? "\_EditPartial" : "Edit", assetVM);

}

Asset asset = MaptoModel(assetVM);

DbContext.Assets.Attach(asset);

DbContext.Entry(asset).State = EntityState.Modified;

var task = DbContext.SaveChangesAsync();

await task;

if (task.Exception != null)

{

ModelState.AddModelError("", "Unable to update the Asset");

Response.StatusCode = (int)HttpStatusCode.BadRequest;

return View(Request.IsAjaxRequest() ? "\_EditPartial" : "Edit", assetVM);

}

if(Request.IsAjaxRequest())

{

return Content("success");

}

return RedirectToAction("Index");

}

Handling Update Ajax Success CallBack

In ***Index.cshtml*** implement the ***OnSuccess***callback function which will be called when the asset gets updated successfully, in the call back function modal would be closed and the form will be cleared so that if user opens for editing again the fresh html would be fetched and updated information will be displayed in View and of course datatable will also be refreshed to display the latest updates:

/\*\*\*\* Edit Asset Ajax Form CallBack \*\*\*\*\*\*\*\*/

function UpdateAssetSuccess(data) {

if (data != "success") {

$('#editAssetContainer').html(data);

return;

}

$('#editAssetModal').modal('hide');

$('#editAssetContainer').html("");

assetListVM.refresh();

}

The same approach will be followed for the details and delete action, let’s see the delete portion, first of all  open the Index view and let’s add the details and delete action hyperlinks in the render function where we defined the edit link above:

"render": function (data, type, full, meta) {

return '<a href="@Url.Action("Edit","Asset")?id=' + data + '" class="editAsset">Edit</a> | <a href="@Url.Action("Details","Asset")?id=' + data + '">Details</a> | <a href="@Url.Action("Delete","Asset")?id=' + data + '">Delete</a>';

}

Now the action column will contain three hyperlinks for Edit, Details and Delete of Asset.

Retrieve and Delete Operation:

At this stage we should be able to see the insert and update functionality working correctly, now we will move to the deletion part to see how we implement the Deletion part for Assets. For doing that, we will need to add class to the hyperlinks which we are generated in the render function for the column in which links will be appearing, let’s do that first, we need to define the **render**property for the last column in **columns**array of the DataTables initialization code, and we will define the how the column value should be rendered:

"render": function (data, type, full, meta) {

return '<a href="@Url.Action("Edit","Asset")?id=' + data + '" class="editAsset">Edit</a> | <a href="@Url.Action("Details","Asset")?id=' + data + '" class="detailsAsset">Details</a> | <a href="@Url.Action("Delete","Asset")?id=' + data + '" class="deleteAsset">Delete</a>';

}

We have added the **detailsAsset**and **deleteAsset**classes to the respective anchor tags so that we can bind the events using jQuery and do some logic to display details or delete particular asset.  
  
After doing the above step, now we will be writing the events to handle the click of these two hyperlinks. We will have to write the following code to achieve it :

$('#assets-data-table').on("click", ".detailsAsset", function (event) {

event.preventDefault();

var url = $(this).attr("href");

$.get(url, function (data) {

$('#detailsAssetContainer').html(data);

$('#detailsAssetModal').modal('show');

});

Now as the handler for details tag is placed, let’s move to the view part and write the needed razor code in the respective view. So, Create a new partial view named **\_detailsPartial** which will be responsible for displaying the details of the particular asset selected. For that again Right click the Asset folder in Views directory in Solution Explorer and do the same steps as previously and create the partial with the name mentioned above and add the following code in it.

@model GridAdvancedSearchMVC.Models.AssetViewModel

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title">Asset Details</h4>

<hr/>

</div>

<div class="modal-body">

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Barcode)

</dt>

<dd>

@Html.DisplayFor(model => model.Barcode)

</dd>

<dt>

@Html.DisplayNameFor(model => model.SerialNumber)

</dt>

<dd>

@Html.DisplayFor(model => model.SerialNumber)

</dd>

<dt>

@Html.DisplayNameFor(model => model.FacilitySite)

</dt>

<dd>

@Html.DisplayFor(model => model.FacilitySite)

</dd>

<dt>

@Html.DisplayNameFor(model => model.PMGuide)

</dt>

<dd>

@Html.DisplayFor(model => model.PMGuide)

</dd>

<dt>

@Html.DisplayNameFor(model => model.AstID)

</dt>

<dd>

@Html.DisplayFor(model => model.AstID)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ChildAsset)

</dt>

<dd>

@Html.DisplayFor(model => model.ChildAsset)

</dd>

<dt>

@Html.DisplayNameFor(model => model.GeneralAssetDescription)

</dt>

<dd>

@Html.DisplayFor(model => model.GeneralAssetDescription)

</dd>

<dt>

@Html.DisplayNameFor(model => model.SecondaryAssetDescription)

</dt>

<dd>

@Html.DisplayFor(model => model.SecondaryAssetDescription)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Quantity)

</dt>

<dd>

@Html.DisplayFor(model => model.Quantity)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Manufacturer)

</dt>

<dd>

@Html.DisplayFor(model => model.Manufacturer)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ModelNumber)

</dt>

<dd>

@Html.DisplayFor(model => model.ModelNumber)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Building)

</dt>

<dd>

@Html.DisplayFor(model => model.Building)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Floor)

</dt>

<dd>

@Html.DisplayFor(model => model.Floor)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Corridor)

</dt>

<dd>

@Html.DisplayFor(model => model.Corridor)

</dd>

<dt>

@Html.DisplayNameFor(model => model.RoomNo)

</dt>

<dd>

@Html.DisplayFor(model => model.RoomNo)

</dd>

<dt>

@Html.DisplayNameFor(model => model.MERNo)

</dt>

<dd>

@Html.DisplayFor(model => model.MERNo)

</dd>

<dt>

@Html.DisplayNameFor(model => model.EquipSystem)

</dt>

<dd>

@Html.DisplayFor(model => model.EquipSystem)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Comments)

</dt>

<dd>

@Html.DisplayFor(model => model.Comments)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Issued)

</dt>

<dd>

@Html.DisplayFor(model => model.Issued)

</dd>

</dl>

</div>

</div>

</div>

Details Get Action Implementation:

Now it’s the time to define the controller action code which will retrieve the asset information from the data source and will pass it back to view to be displayed to the user. Here is the code for controller action:

public async Task<ActionResult> Details(Guid id)

{

var asset = await DbContext.Assets.FirstOrDefaultAsync(x => x.AssetID == id);

var assetVM = MapToViewModel(asset);

if(Request.IsAjaxRequest())

return PartialView("\_detailsPartial", assetVM);

return View(assetVM);

}

Implementing Delete Operation:

We will add the GET action method for delete which will get the particular asset from the repository and will display the details in popup, where user would be able to delete it or cancel it. Here is the code for the action method:

public ActionResult Delete(Guid id)

{

var asset = DbContext.Assets.FirstOrDefault(x => x.AssetID == id);

AssetViewModel assetViewModel = MapToViewModel(asset);

if (Request.IsAjaxRequest())

return PartialView("\_DeletePartial", assetViewModel);

return View(assetViewModel);

}

Delete Partial View Addition

We will now add another partial view in the solution for delete part, so navigate to the ***Views >> Asset*** folder in the ***Solution Explorer*** and from the context menu which appears by right clicking the ***Asset***folder, add a new View using the Option ***Add >> MVC 5 Partial Page (Razor)*** and name the partial view to ***\_DeletePartial*** and add the following code to it:

@model GridAdvancedSearchMVC.Models.AssetViewModel

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">×</button>

<h4 class="modal-title">Delete Asset</h4>

<h3>Are you sure you want to delete this?</h3>

</div>

<div class="modal-body">

<dl class="dl-horizontal">

<dt>

@Html.DisplayNameFor(model => model.Barcode)

</dt>

<dd>

@Html.DisplayFor(model => model.Barcode)

</dd>

<dt>

@Html.DisplayNameFor(model => model.SerialNumber)

</dt>

<dd>

@Html.DisplayFor(model => model.SerialNumber)

</dd>

<dt>

@Html.DisplayNameFor(model => model.FacilitySite)

</dt>

<dd>

@Html.DisplayFor(model => model.FacilitySite)

</dd>

<dt>

@Html.DisplayNameFor(model => model.PMGuide)

</dt>

<dd>

@Html.DisplayFor(model => model.PMGuide)

</dd>

<dt>

@Html.DisplayNameFor(model => model.AstID)

</dt>

<dd>

@Html.DisplayFor(model => model.AstID)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ChildAsset)

</dt>

<dd>

@Html.DisplayFor(model => model.ChildAsset)

</dd>

<dt>

@Html.DisplayNameFor(model => model.GeneralAssetDescription)

</dt>

<dd>

@Html.DisplayFor(model => model.GeneralAssetDescription)

</dd>

<dt>

@Html.DisplayNameFor(model => model.SecondaryAssetDescription)

</dt>

<dd>

@Html.DisplayFor(model => model.SecondaryAssetDescription)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Quantity)

</dt>

<dd>

@Html.DisplayFor(model => model.Quantity)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Manufacturer)

</dt>

<dd>

@Html.DisplayFor(model => model.Manufacturer)

</dd>

<dt>

@Html.DisplayNameFor(model => model.ModelNumber)

</dt>

<dd>

@Html.DisplayFor(model => model.ModelNumber)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Building)

</dt>

<dd>

@Html.DisplayFor(model => model.Building)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Floor)

</dt>

<dd>

@Html.DisplayFor(model => model.Floor)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Corridor)

</dt>

<dd>

@Html.DisplayFor(model => model.Corridor)

</dd>

<dt>

@Html.DisplayNameFor(model => model.RoomNo)

</dt>

<dd>

@Html.DisplayFor(model => model.RoomNo)

</dd>

<dt>

@Html.DisplayNameFor(model => model.MERNo)

</dt>

<dd>

@Html.DisplayFor(model => model.MERNo)

</dd>

<dt>

@Html.DisplayNameFor(model => model.EquipSystem)

</dt>

<dd>

@Html.DisplayFor(model => model.EquipSystem)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Comments)

</dt>

<dd>

@Html.DisplayFor(model => model.Comments)

</dd>

<dt>

@Html.DisplayNameFor(model => model.Issued)

</dt>

<dd>

@Html.DisplayFor(model => model.Issued)

</dd>

</dl>

@using (Ajax.BeginForm("Delete", "Asset", null, new AjaxOptions { HttpMethod = "Post", OnSuccess = "DeleteAssetSuccess" }, new { @class = "form-horizontal", role = "form" }))

{

<div class="form-actions no-color">

@Html.HiddenFor(x => x.AssetID)

<input type="submit" value="Delete" class="btn btn-default" /> |

@Html.ActionLink("Back to List", "Index",null,new { data\_dismiss = "modal" })

</div>

}

</div>

</div>

</div>

Now we need to again a container div for delete popup as well which will be holding the html returned by the partial view ***\_DeletePartial*** as we did for other three operations, so add the following html in the ***Index.cshtml*** view of ***Asset***:

<div class="modal fade" id="deleteAssetModal" tabindex="-1" role="dialog" aria-labelledby="DeleteAssetModal" aria-hidden="true" data-backdrop="static">

<div id="deleteAssetContainer">

</div>

</div>

Handling Delete Link Event with JQuery:

We also need to implement the delete button click event which will be responsible for calling the action method asynchronously and will add the response html to the popup container and the popup container will be displayed to user, the code for which would be:

$('#assets-data-table').on("click", ".deleteAsset", function (event) {

event.preventDefault();

var url = $(this).attr("href");

$.get(url, function (data) {

$('#deleteAssetContainer').html(data);

$('#deleteAssetModal').modal('show');

});

});

Handling DELETE Ajax POST Success Callback:

If you notice here as well we are using ***Ajax.BeginForm*** helper method to post the **GUID**of Asset row that needs to be deleted, and we have specified a JavaScript success callback, but we haven’t yet defined the function in the ***Index***view, so let’s do that as well, here is the function definition:

function DeleteAssetSuccess(data) {

if (data != "success") {

$('#deleteAssetContainer').html(data);

return;

}

$('#deleteAssetModal').modal('hide');

$('#deleteAssetContainer').html("");

assetListVM.refresh();

}

This is also quite same as we did for other actions, we are making sure that if deletion was successful or not, we update the UI accordingly, that’s what the above code is doing, it closes the popup, clears the html of the container div and refreshes the datatable ViewModel to reflect the latest changes in the grid.

Implementing Delete POST Action:

Lastly, we need to define the POST action for delete which will be responsible for deleting the row from the database table and will return the status of the action back to view in either case i.e. success or failure, let’s do that, here is the code for **Delete**post action:

[HttpPost, ActionName("Delete")]

public async Task<ActionResult> DeleteAsset(Guid AssetID)

{

var asset = new Asset { AssetID = AssetID };

DbContext.Assets.Attach(asset);

DbContext.Assets.Remove(asset);

var task = DbContext.SaveChangesAsync();

await task;

if (task.Exception != null)

{

ModelState.AddModelError("", "Unable to Delete the Asset");

Response.StatusCode = (int)HttpStatusCode.BadRequest;

AssetViewModel assetVM = MapToViewModel(asset);

return View(Request.IsAjaxRequest() ? "\_DeletePartial" : "Delete", assetVM);

}

if (Request.IsAjaxRequest())

{

return Content("success");

}

return RedirectToAction("Index");

}

Now run the application and you should be able to add update and delete rows via Ajax without navigating between the pages.