http://www.dotnetawesome.com/2016/12/crud-operation-using-datatables-aspnet-mvc.html

https://www.youtube.com/watch?v=3pEax-5wXG8&list=WL&index=28

Introduction

In the previous article of this series "[Implementing jQuery Datatables in ASP.NET MVC application](http://www.dotnetawesome.com/search/label/jQuery%20DataTable)", we have seen followings.

1. [Implement jQuery Datatable in ASP.NET MVC application](http://www.dotnetawesome.com/2015/11/implement-jquery-datatable-in-aspnet-mvc.html)
2. [jQuery Datatable server side pagination and sorting in ASP.NET MVC](http://www.dotnetawesome.com/2015/11/jquery-datatable-server-side-pagination-sorting.html)
3. [Implement custom multicolumn server-side filtering in jQuery dataTables](http://www.dotnetawesome.com/2015/12/implement-custom-server-side-filtering-jquery-datatables.html)
4. [Full CRUD operation using datatables in ASP.NET MVC](http://www.dotnetawesome.com/)

Where we have used [Datatables](https://datatables.net/) plugin to basically display data with various features.  
  
Because a lot of requests from readers to me to write a tutorial for implement CRUD operation in Datatables, today I am going to show you how we can implement CRUD operation in Datatables.

We know that Datatable is free but it's editor, which is used for CRUD operations are paid. Hence in this article, we will implement CRUD operation in Datatable using jquery modal popup.  
  
Ok, Let's start implementing CRUD operation in jQuery Datatable.

Follow the following steps in order to implement "CRUD operation using Datatables in ASP.NET MVC".

**Here In this article, I have used Visual Studio 2015**

Step - 1: Create New Project.

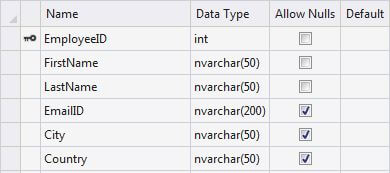
Go to File > New > Project > ASP.NET  Web Application (under web) > Enter enter application name > select your project location > and then click on add button > It will brings up a new dialog window for select template > here I will select Empty template > checked  MVC checkbox from Add folder and core referances for: > and then click on ok button.

Step-2: Add a Database.

Now I will create a database for our application. As this is a tutorial project, I will add a database in our applications here in the app\_data folder.  
  
Go to Solution Explorer > Right Click on App\_Data folder > Add > New item > Select SQL Server Database Under Data > Enter Database name > Add.

Step-3: Create a table in our database.

In this example, I have added the table for store Employee information for implementing CRUD operation in Datatable.

[](https://2.bp.blogspot.com/-MjUdUITj_Ak/WFJB-P48H-I/AAAAAAAACXk/cOtpHthyXnsXAqs0eC6iaxv8ChK0JdPvQCLcB/s1600/table.JPG)

double click on the database under **app\_data**folder  for open the database in server explorer > expand the database and Right click on Tables node > click on Add New Table >  here we will write schema of the table for the table we want to create > now click on **Update** button for create the table and then again click on **Update Database** button.

Step-4: Add Entity Data Model.

Go to Solution Explorer > Right Click on Project name form Solution Explorer > Add > New item > Select ADO.net Entity Data Model under data > Enter model name > Add.  
A popup window will come (Entity Data Model Wizard) > Select Generate from database > Next >  
Chose your data connection > select your database > next > Select tables > enter Model Namespace > Finish.  
  
As I have already shown you how we can implement [server-side paging, sorting](http://www.dotnetawesome.com/2015/11/jquery-datatable-server-side-pagination-sorting.html" \t "_blank), [filtering functionality in Datatables](http://www.dotnetawesome.com/2015/12/implement-custom-server-side-filtering-jquery-datatables.html" \t "_blank), here we will only focus on CRUD functionality. So here we will show data in Datatables without implementing server-side paging, sorting, filtering functionality for the sake of simplicity.

Step-5: Create a Controller.

Go to Solution Explorer > Right Click on Controllers folder form Solution Explorer > Add > Controller > Enter Controller name > Select Templete "empty MVC Controller"> Add.

Here I have created a controller "HomeController"

Now we will add a view for Index action (which is already added in the home controller) where we will show data in Datatables.

Step-6: Add view for Index action.

Right Click on Action Method (here right click on Index action) > Add View... > Enter View Name > Select "Empty" under Template dropdown  > > Add.  
In this page, first of all, we need to add required the jQuery & CSS libraries.  So we will add those required libraries in our application from NuGet package manager.  
For add libraries from NuGet package manager  
Go to solution explorer > Right Click on your project name > Manage NuGet Packages... > Search & select the resources you want to Add in your project > click Install button.  
We will add following libraries from NuGet Package Manager

1. jQuery Library (I have downloaded jQuery 3.1.1)
2. jQuery.UI.Combined 1.12.1
3. jQuery.Validation 1.16.0
4. Microsoft.jQuery.Unobtrusive.Validation 3.2.3

**HTML Code**

1. @{
2. Layout = null;
3. }
4. <!DOCTYPE html>
5. <html>
6. <head>
7. <meta name="viewport" content="width=device-width" />
8. <title>Index</title>
9. <link rel="stylesheet" href="<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css>" />
10. <link rel="stylesheet" href="<https://cdn.datatables.net/1.10.13/css/jquery.dataTables.min.css>" />
11. <link href="~/Content/themes/base/jquery-ui.min.css" rel="stylesheet" />
12. <style>
13. span.field-validation-error {
14. color: red;
15. }
16. </style>
17. </head>
18. <body>
19. <div style="width:90%; margin:0 auto" class="tablecontainer">
20. <a class="popup btn btn-primary" href="/home/save/0" style="margin-bottom:20px; margin-top:20px;">Add New Employee</a>
21. <table id="myDatatable">
22. <thead>
23. <tr>
24. <th>First Name</th>
25. <th>Last Name</th>
26. <th>Email ID</th>
27. <th>City</th>
28. <th>Country</th>
29. <th>Edit</th>
30. <th>Delete</th>
31. </tr>
32. </thead>
33. </table>
34. </div>
36. <script src="~/Scripts/jquery-3.1.1.min.js"></script>
37. <script src="~/Scripts/jquery.validate.min.js"></script>
38. <script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
39. <script src="<https://cdn.datatables.net/1.10.13/js/jquery.dataTables.min.js>"></script>
40. <script src="~/Scripts/jquery-ui-1.12.1.min.js"></script>
42. <script>
43. $(document).ready(function () {
44. var oTable = $('#myDatatable').DataTable({
45. "ajax": {
46. "url" : '/home/GetEmployees',
47. "type" : "get",
48. "datatype" : "json"
49. },
50. "columns": [
51. { "data": "FirstName", "autoWidth": true },
52. { "data" : "LastName", "autoWidth" : true},
53. { "data": "EmailID", "autoWidth": true },
54. { "data": "City", "autoWidth": true },
55. { "data": "Country", "autoWidth": true },
56. {
57. "data": "EmployeeID", "width": "50px", "render": function (data) {
58. return '<a class="popup" href="/home/save/'+data+'">Edit</a>';
59. }
60. },
61. {
62. "data": "EmployeeID", "width": "50px", "render": function (data) {
63. return '<a class="popup" href="/home/delete/' + data + '">Delete</a>';
64. }
65. }
66. ]
67. })
68. $('.tablecontainer').on('click', 'a.popup', function (e) {
69. e.preventDefault();
70. OpenPopup($(this).attr('href'));
71. })
72. function OpenPopup(pageUrl) {
73. var $pageContent = $('<div/>');
74. $pageContent.load(pageUrl, function () {
75. $('#popupForm', $pageContent).removeData('validator');
76. $('#popupForm', $pageContent).removeData('unobtrusiveValidation');
77. $.validator.unobtrusive.parse('form');
79. });
81. $dialog = $('<div class="popupWindow" style="overflow:auto"></div>')
82. .html($pageContent)
83. .dialog({
84. draggable : false,
85. autoOpen : false,
86. resizable : false,
87. model : true,
88. title:'Popup Dialog',
89. height : 550,
90. width : 600,
91. close: function () {
92. $dialog.dialog('destroy').remove();
93. }
94. })
96. $('.popupWindow').on('submit', '#popupForm', function (e) {
97. var url = $('#popupForm')[0].action;
98. $.ajax({
99. type : "POST",
100. url : url,
101. data: $('#popupForm').serialize(),
102. success: function (data) {
103. if (data.status) {
104. $dialog.dialog('close');
105. oTable.ajax.reload();
106. }
107. }
108. })
110. e.preventDefault();
111. })
112. $dialog.dialog('open');
113. }
114. })
115. </script>
116. </body>
117. </html>

Step-7: Add a new method in HomeController.

Here I have added a method **GetEmployees**for fetch employees data from the database.

1. public ActionResult GetEmployees()
2. {
3. using (MyDatabaseEntities dc = new MyDatabaseEntities())
4. {
5. var employees = dc.Employees.OrderBy(a => a.FirstName).ToList();
6. return Json(new { data = employees }, JsonRequestBehavior.AllowGet);
7. }
8. }

Now for doing CRUD operation, we will add some MVC action in our Home Controller.

Step-8: Add a new MVC Action Save (GET) in HomeController.

Here we will add a new MVC Action (GET Method) for getting the form for add/update employee information.

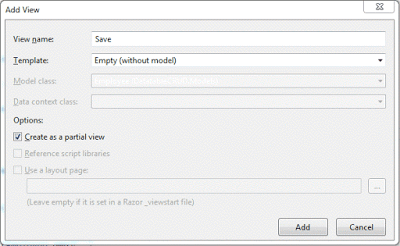
1. [HttpGet]
2. public ActionResult Save(int id)
3. {
4. using (MyDatabaseEntities dc = new MyDatabaseEntities())
5. {
6. var v = dc.Employees.Where(a => a.EmployeeID == id).FirstOrDefault();
7. return View(v);
8. }
9. }

Step-9: Add a partial view for that  Save MVC action.

Here we will add a partial view for the save action as we will open the form in jQuery popup dialog window.

1. @model DatatableCRUD.Models.Employee
2. <h2>Save</h2>
3. @using (Html.BeginForm("save","home", FormMethod.Post, new { id= "popupForm" }))
4. {
5. if (Model != null && Model.EmployeeID > 0)
6. {
7. @Html.HiddenFor(a=>a.EmployeeID)
8. }
10. <div class="form-group">
11. <label>First Name</label>
12. @Html.TextBoxFor(a=>a.FirstName,new { @class = "form-control" })
13. @Html.ValidationMessageFor(a=>a.FirstName)
14. </div>
15. <div class="form-group">
16. <label>Last Name</label>
17. @Html.TextBoxFor(a => a.LastName, new { @class = "form-control" })
18. @Html.ValidationMessageFor(a => a.LastName)
19. </div>
20. <div class="form-group">
21. <label>Email</label>
22. @Html.TextBoxFor(a => a.EmailID, new { @class = "form-control" })
23. @Html.ValidationMessageFor(a => a.EmailID)
24. </div>
25. <div class="form-group">
26. <label>City</label>
27. @Html.TextBoxFor(a => a.City, new { @class = "form-control" })
28. @Html.ValidationMessageFor(a => a.City)
29. </div>
30. <div class="form-group">
31. <label>Country</label>
32. @Html.TextBoxFor(a => a.Country, new { @class = "form-control" })
33. @Html.ValidationMessageFor(a => a.Country)
34. </div>
36. <div>
37. <input type="submit" value="Save" />
38. </div>
39. }

Right click inside the save action >  Add View.. > Enter view name > Select Empty (without model) from template dropdown > Check the Checkbox "Create as a partial view".

[](https://1.bp.blogspot.com/-hObKxXxy7DY/WFJQYpfSC-I/AAAAAAAACX0/fzWaEOaB0cU6XfDK31xZkuolL4Ut4h7BwCLcB/s1600/View.PNG)

Step-10: Add an another MVC Action Save (POST) in HomeController.

Here we will add a another MVC Action (POST Method) for saving data to the database.

1. [HttpPost]
2. public ActionResult Save(Employee emp)
3. {
4. bool status = false;
5. if (ModelState.IsValid)
6. {
7. using (MyDatabaseEntities dc = new MyDatabaseEntities())
8. {
9. if (emp.EmployeeID > 0)
10. {
11. //Edit
12. var v = dc.Employees.Where(a => a.EmployeeID == emp.EmployeeID).FirstOrDefault();
13. if (v != null)
14. {
15. v.FirstName = emp.FirstName;
16. v.LastName = emp.LastName;
17. v.EmailID = emp.EmailID;
18. v.City = emp.City;
19. v.Country = emp.Country;
20. }
21. }
22. else
23. {
24. //Save
25. dc.Employees.Add(emp);
26. }
27. dc.SaveChanges();
28. status = true;
29. }
30. }
31. return new JsonResult { Data = new { status = status} };
32. }

Step-11: Add an another MVC Action Delete (GET) in HomeController.

Here we will add an another MVC Action (GET Method) for getting the confirm dialog window before deleting the selected employee.

1. [HttpGet]
2. public ActionResult Delete(int id)
3. {
4. using (MyDatabaseEntities dc = new MyDatabaseEntities())
5. {
6. var v = dc.Employees.Where(a => a.EmployeeID == id).FirstOrDefault();
7. if (v != null)
8. {
9. return View(v);
10. }
11. else
12. {
13. return HttpNotFound();
14. }
15. }
16. }

Step-12: Add a partial view for that  Delete MVC action.

Right click inside the save action >  Add View.. > Enter view name > Select Empty (without model) from template dropdown > Check the Checkbox "Create as a partial view".

1. @model DatatableCRUD.Models.Employee
3. <h2>Delete Employee</h2>
4. @using (Html.BeginForm("delete","home", FormMethod.Post,new { id= "popupForm" }))
5. {
6. @Html.HiddenFor(a=>a.EmployeeID)
7. <div class="form-group">
8. <label>First Name</label>
9. <p>@Model.FirstName</p>
10. </div>
12. <div class="form-group">
13. <label>Last Name</label>
14. <p>@Model.LastName</p>
15. </div>
17. <div class="form-group">
18. <label>Email</label>
19. <p>@Model.EmailID</p>
20. </div>
22. <div class="form-group">
23. <label>City</label>
24. <p>@Model.City</p>
25. </div>
27. <div class="form-group">
28. <label>Country</label>
29. <p>@Model.Country</p>
30. </div>
32. <div>
33. <input type="submit" value="Delete" />
34. </div>
35. }

Step-13: Add an another MVC Action Delete (POST) in HomeController.

Here in this MVC Action, we will write code for delete selected employee data from our database.

1. [HttpPost]
2. [ActionName("Delete")]
3. public ActionResult DeleteEmployee(int id)
4. {
5. bool status = false;
6. using (MyDatabaseEntities dc = new MyDatabaseEntities())
7. {
8. var v = dc.Employees.Where(a => a.EmployeeID == id).FirstOrDefault();
9. if (v != null)
10. {
11. dc.Employees.Remove(v);
12. dc.SaveChanges();
13. status = true;
14. }
15. }
16. return new JsonResult { Data = new { status = status} };
17. }

Step-14: Run Application.

We have done all the steps. Now it's time to run the application.