

Server Side Pagination With Datatables

03 September 2019

Goals:

In "ASP.NET CORE Web Application",

- Create list of objects to be used as test data.
- Display records in datatables
- Use page ids of datatables to fetch data from data source.

Keys:

Paging, Pagination

Datatables

Create sample objects in startup.

Static keyword usage.

Assumptions:

Your project is already open in Visual Studio.

Only the Details page will be ready at the end of the example. Create, Update and Delete pages have been ignored to make video shorter.

Scenario:

Prepare sample data to be displayed in tables.

Make necessary changes to enable pagination via datatables and ajax calls.

Fetch and Display example students with paging feature.

Listen RUD buttons events and determine id of clicked button of datatable records.

Result:

Now the result of the example will be displayed.

Summary of Steps

| | |
|--|----|
| Part 1: Prepare Sample Data | 3 |
| 1. Create Student class as an example object class | 3 |
| 2. Create Data folder in project. | 3 |
| 3. Create StudentContext class in Data folder as if a data source to keep Student objects. | 3 |
| 4. Update Startup.cs to initialize data at application start | 3 |
| Part 2: Preparing Controller and View | 4 |
| 1. Create StudentsController | 4 |
| 2. Create Index View of Students | 4 |
| 3. Install latest dataTables files as client-side library | 4 |
| 4. Add dataTables CSS into Students/Index view | 4 |
| 5. Update view to display table using dataTables. | 4 |
| 6. Add dataTables.js into Scripts section of the view file | 4 |
| 7. Add js code to initialize datatable into section Scripts | 5 |
| 8. Add link to Home page to be able to display Students/Index page | 5 |
| 9. Update Details Action and Create Details View of Students | 5 |
| 10. Run Project | 5 |
| Part 3: Use Server Side Paging Feature In DataTable | 5 |
| 1. Stop the application if its still running. | 5 |
| 2. Create action in StudentsController to get filtered records related to paging requests | 5 |
| 3. Update Students/Index to display student list in Home page | 7 |
| 4. Optional: Update site.css to enable loading spinner during data fetching calls. | 10 |
| 5. Run Project | 11 |

Steps

Part 1: Prepare Sample Data

We will create Student class and StudentContext to keep sample Student objects in StudentList. Also we will update startup.cs to create initial students at application startup. The students will be used in datatable in Part 2 and Part 3.

1. Create Student class as an example object class

- A. Right click on Models folder + mouseover to Add + Select Class
- B. Enter the name of the class as Student
- C. In Models/Student.cs that we have just created, enter the properties of Student class as below

```
public class Student
{
    public int Id { get; set; }
    public string Firstname { get; set; }
    public string Lastname { get; set; }
    public DateTime CreatedDate { get; set; }
}
```

2. Create Data folder in project.

- A. Right click on Project folder + mouseover to Add + Select New Folder
- B. Enter the name of the folder as Data

3. Create StudentContext class in Data folder as if a data source to keep Student objects.

- A. Right click on Data folder + mouseover to Add + Select class
- B. Enter the name of the class as StudentContext.cs
- C. Replace code of StudentContext class with the code below

```
public static class StudentContext
{
    //StudentList is static to be able to reach in application scope
    public static List<Student> StudentList { get; set; }

    //Creates "studentCount" students and adds into student list
    public static void InitStudentList(int studentCount)
    {
        StudentList = new List<Student>();
        for (int i = 1; i < studentCount + 1; i++)
        {
            StudentList.Add(
                new Student()
                {
                    Id = i,
                    Firstname = "Firstname" + i,
                    Lastname = "Lastname" + i,
                    CreatedDate = DateTime.Now
                }
            );
        }
    }
}
```

- D. Add using statement for the Student class.

4. Update Startup.cs to initialize data at application start

- A. Open /Startup.cs file
- B. Add the code line below to create 1000 students

```
public Startup(IConfiguration configuration)
{
    Configuration = configuration;
}
```

```
Data.StudentContext.InitStudentList(1000); //Add this line to create 1000
students
}
```

Part 2: Preparing Controller and View

1. Create StudentsController

- Right click on Controllers folder
- Select Add / Controller
- Select MvcController with read/write actions + Click Add
- Enter name of the class as StudentsController
- Update the Index action in StudentsController.cs as below

```
public ActionResult Index()
{
    return View(Data.StudentContext.StudentList); // Send students to view
}
```

2. Create Index View of Students

- Right click on Index action in StudentsController.cs
- Select add / view
- Select template as List
- Select model class as Student + Click Add

3. Install latest dataTables files as client-side library

- right click on wwwroot/lib
- select add clientside library
- enter datatables@ into Library field.
- select latest datatables version (1.10.19 current version in this example)
- click Install.

4. Add dataTables CSS into Students/Index view

- Open Views/Students/Index.cshtml
- Drag and Drop "**jquery.dataTables.css**" file from wwwroot/lib after the table definition.

5. Update view to display table using dataTables.

- Open view file /Views/Students/Index.cshtml
- Find table definition.
- Update header before the table definition as `<h1>Students</h1>`
- Update id of the table as `<table id="studentTable" class="table table-striped">`
- Add code below to the bottom of the view file.

```
@section Scripts{
}
```

6. Add dataTables.js into Scripts section of the view file

- Drag and Drop "**jquery.dataTables.js**" file from wwwroot/lib into the section Scripts.
Section Scripts will be like below.

```
@section Scripts{
<script src="~/lib/datatables/js/jquery.dataTables.js"></script>
}
```

7. Add js code to initialize datatable into section Scripts

```
@section Scripts{
//script for datatables js.
<script src="~/lib/datatables/js/jquery.dataTables.js"></script>
//script to initialize datatable.
<script>
    $(document).ready(function () {
        $('#studentTable').DataTable();//studentTable is the id of the
        table to be displayed as dataTable
    });
</script>
}
```

8. Add link to Home page to be able to display Students/Index page

- Open Views/Home/Index.cshtml
- Add code below into the page.

```
<h1>Datatables With Pagination At Server Side</h1>

<a asp-controller="Students" asp-action="Index">Open Students Page</a>
```

9. Update Details Action and Create Details View of Students

- Find the Details action in StudentsController.cs
- Replace the Details action in StudentsController.cs as below to display Student data.

```
public ActionResult Details(int id)
{
    Student s = Data.StudentContext.StudentList.FirstOrDefault(a => a.Id == id);
    return View(s);
}
```

- Right click on Details action in StudentsController
- Select add / view
- Select template as Details
- Select model class as Student + Click Add

10. Run Project.

Part 3: Use Server Side Paging Feature In Datatable

- Stop the application if its still running.
- Create action in StudentsController to get filtered records related to paging requests.
 - Open Controllers/StudentsControllers.cs
 - Find Index action and delete the parameter that sends StudentList to view.
 - Add new action to be used by datatable ajax requests.

```
//Action to be called by js in details page when search, sort or page numbers
clicked
// Search is applied only to Firstname and Lastname properties
public JsonResult GetFilteredItems()
{
    System.Threading.Thread.Sleep(2000);//Used to display loading
    spinner in demonstration, remove this line in production
    int draw = Convert.ToInt32(Request.Query["draw"]);

    // Data to be skipped ,
```

```

// if 0 first "length" records will be fetched
// if 1 second "length" of records will be fetched ...
int start = Convert.ToInt32(Request.Query["start"]);

// Records count to be fetched after skip
int length = Convert.ToInt32(Request.Query["length"]);

// Getting Sort Column Name
int sortColumnIdx =
Convert.ToInt32(Request.Query["order[0][column]"]);
string sortColumnName = Request.Query["columns[" + sortColumnIdx +
"name]"];

// Sort Column Direction
string sortColumnDirection = Request.Query["order[0][dir]"];

// Search Value
string searchValue =
Request.Query["search[value]"].FirstOrDefault()?.Trim();

// Records Count matching search criteria
int recordsFilteredCount =
    Data.StudentContext.StudentList
        .Where(a => a.Lastname.Contains(searchValue) ||
a.Firstname.Contains(searchValue))
        .Count();

// Total Records Count
int recordsTotalCount = Data.StudentContext.StudentList.Count();

// Filtered & Sorted & Paged data to be sent from server to view
List<Student> filteredData = null;
if (sortColumnDirection == "asc")
{
    filteredData =
        Data.StudentContext.StudentList
            .Where(a => a.Lastname.Contains(searchValue) ||
a.Firstname.Contains(searchValue))
            .OrderBy(x =>
x.GetType().GetProperty(sortColumnName).GetValue(x))//Sort by sortColumn
            .Skip(start)
            .Take(length)
            .ToList<Student>();
}
else
{
    filteredData =
        Data.StudentContext.StudentList
            .Where(a => a.Lastname.Contains(searchValue) ||
a.Firstname.Contains(searchValue))
            .OrderByDescending(x =>
x.GetType().GetProperty(sortColumnName).GetValue(x))
            .Skip(start)
            .Take(length)
            .ToList<Student>();
}
// Send data
return Json(
    new {
        data = filteredData,
        draw = Request.Query["draw"],
        recordsFiltered = recordsFilteredCount,
        recordsTotal = recordsTotalCount
    }
);
}

```

3. Update Students/Index to display student list in Home page

- Open view file Views/Students/Index.cshtml
- Find table definition.
- Remove table body including `<tbody></tbody>` tags. (Table body will be provided by ajax request given in next step)
- Enter "Actions" text between last(empty) `th` tags to be used for crud actions `<th></th>`
- Delete script to initialize datatable. We will add different code in next step. Be careful, do not delete the script link to datatables js library.

```
@section Scripts{
    //script for datatables js.
    <script src="~/lib/datatables/js/jquery.dataTables.js"></script>
    //script to initialize datatable.
    <script>
    $(document).ready(function () {
        $('#studentTable').DataTable();//studentTable is the id of the table to
        be displayed as dataTable
    });
    </script>
}
```

- Add scripts to initialize and manage ajax requests as below.

This script has been copied from https://datatables.net/examples/server_side/pipeline.html Although it seems complex, very few lines have been changed. Updated parts have been highlighted with yellow with comments.

```
<script>
    //
    // Pipelining function for DataTables. To be used to the `ajax` option of
    DataTables
    // Copied from https://datatables.net/examples/server_side/pipeline.html
    //
    $.fn.dataTable.pipeline = function (opts) {
        // Configuration options
        var conf = $.extend({
            pages: 5, // number of pages to cache. That means action(url) will be
            called in 1st, 6th, 11th ... pages
            url: 'Students/GetFilteredItems', // url to controller action
            data: null, // function or object with parameters to send to the server
            method: 'GET' // Ajax HTTP method
        }, opts);

        // Private variables for storing the cache
        var cacheLower = -1;
        var cacheUpper = null;
        var cacheLastRequest = null;
        var cacheLastJson = null;

        return function (request, drawCallback, settings) {
            var ajax = false;
            var requestStart = request.start;
            var drawStart = request.start;
            var requestLength = request.length;
            var requestEnd = requestStart + requestLength;

            if (settings.clearCache) {
                // API requested that the cache be cleared
                ajax = true;
                settings.clearCache = false;
            }
            else if (cacheLower < 0 || requestStart < cacheLower || requestEnd >
            cacheUpper) {
```

```

        // outside cached data - need to make a request
        ajax = true;
    }
    else if (JSON.stringify(request.order) !==
JSON.stringify(cacheLastRequest.order) ||
        JSON.stringify(request.columns) !==
JSON.stringify(cacheLastRequest.columns) ||
        JSON.stringify(request.search) !==
JSON.stringify(cacheLastRequest.search)
    ) {
        // properties changed (ordering, columns, searching)
        ajax = true;
    }

    // Store the request for checking next time around
    cacheLastRequest = $.extend(true, {}, request);

    if (ajax) {
        // Need data from the server
        if (requestStart < cacheLower) {
            requestStart = requestStart - (requestLength * (conf.pages - 1));

            if (requestStart < 0) {
                requestStart = 0;
            }
        }

        cacheLower = requestStart;
        cacheUpper = requestStart + (requestLength * conf.pages);

        request.start = requestStart;
        request.length = requestLength * conf.pages;

        // Provide the same `data` options as DataTables.
        if (typeof conf.data === 'function') {
            // As a function it is executed with the data object as an arg
            // for manipulation. If an object is returned, it is used as the
            // data object to submit
            var d = conf.data(request);
            if (d) {
                $.extend(request, d);
            }
        }
        else if ($.isPlainObject(conf.data)) {
            // As an object, the data given extends the default
            $.extend(request, conf.data);
        }

        settings.jqXHR = $.ajax({
            "type": conf.method,
            "url": conf.url,
            "data": request,
            "dataType": "json",
            "cache": false,
            "success": function (json) {
                cacheLastJson = $.extend(true, {}, json);

                if (cacheLower !== drawStart) {
                    json.data.splice(0, drawStart - cacheLower);
                }
                if (requestLength >= -1) {
                    json.data.splice(requestLength, json.data.length);
                }

                drawCallback(json);
            }
        });
    }
}

```



```

else {
    json = $.extend(true, {}, cacheLastJson);
    json.draw = request.draw; // Update the echo for each response
    json.data.splice(0, requestStart - cacheLower);
    json.data.splice(requestLength, json.data.length);

    drawCallback(json);
}
}
};

// Register an API method that will empty the pipelined data, forcing an Ajax
// fetch on the next draw (i.e. `table.clearPipeline().draw()`)
// Copied from https://datatables.net/examples/server_side/pipeline.html
$.fn.dataTable.Api.register('clearPipeline()', function () {
    return this.iterator('table', function (settings) {
        settings.clearCache = true;
    });
});

//
// DataTables initialization
// Copied from https://datatables.net/examples/server_side/pipeline.html
// Updated according to our data
//
$(document).ready(function () {
    $('#studentTable').DataTable({
        "processing": true,
        "serverSide": true,
        "searching": true,
        "paging": true,
        "ajax": $.fn.dataTable.pipeline({
            url: 'Students/GetFilteredItems',
            pages: 5 //number of pages to cache
        }),
        //Column definitions are sent to action to be used in sorted column definition
        //name parts are assigned as the exact property name to determine sort columns
        // render definition has been given to display format of CreatedDate property
        "columns": [
            // For Student.Id
            { "data": "id", "name": "Id" },
            // For Student.Firstname
            { "data": "firstname", "name": "Firstname" },
            // For Student.Lastname
            { "data": "lastname", "name": "Lastname" },
            // For Student.CreatedDate
            { "data": "createdDate", "name": "CreatedDate",
                "render": function (data) {
                    var date = new Date(data);
                    return date.toLocaleString();
                }
            },
            // Optional: Buttons For Action Listeners
            {
                'data': null,
                'render': function (data, type, row) {
                    return '<button id="' + row.id + '"
onclick="detailsClick(this)">Details</button>'
                    + '<button id="' + row.id + '"
onclick="editClick(this)">Edit</button>'
                    + '<button id="' + row.id + '"
onclick="deleteClick(this)">Delete</button>'
                }
            }
        ],
        language: {
            processing: '<div class="spinner"></div>', // Optional to use loading
            spinner. Instead of it you can define a simple string.
            zeroRecords: "No matching records found"
        }
    });
});

```

```

    });
  });

  //Optional: Details button listener
  function detailsClick(obj) {
    //var rowID = $(obj).attr('id');
    var studentId = $(obj).closest('tr').find('td:first').html();
    alert("Id = " + studentId + " for details");
  }
  // Optional: Edit button listener
  function editClick(obj) {
    //var rowID = $(obj).attr('id');
    var studentId = $(obj).closest('tr').find('td:first').html();
    alert("Id = " + studentId + " for edit");
  }
  // Optional: Delete button listener
  function deleteClick(obj) {
    //var rowID = $(obj).attr('id');
    var studentId = $(obj).closest('tr').find('td:first').html();
    alert("Id = " + studentId + " for delete");
  }
}
</script>

```

4. Optional: Update site.css to enable loading spinner during data fetching calls.

A. Open /wwwroot/site.css

```

/*
  Begin - Style of Centered spinner
  Code from https://www.w3schools.com/howto/howto_css_loader.asp
*/
.spinner {
  position: absolute;
  left: 50%;
  top: 50%;
  z-index: 1;
  width: 150px;
  height: 150px;
  margin: -75px 0 0 -75px;
  border: 16px solid #f3f3f3;
  border-radius: 50%;
  border-top: 16px solid #3498db;
  width: 120px;
  height: 120px;
  -webkit-animation: spin 2s linear infinite;
  animation: spin 2s linear infinite;
}

@-webkit-keyframes spin {
  0% {
    -webkit-transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(360deg);
  }
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }
}

```

B. Add code below to the bottom of the site.css file for loading

```
100% {
    transform: rotate(360deg);
}

/* Add animation to "page content" */
.animate-bottom {
    position: relative;
    -webkit-animation-name: animatebottom;
    -webkit-animation-duration: 1s;
    animation-name: animatebottom;
    animation-duration: 1s
}

@-webkit-keyframes animatebottom {
    from {
        bottom: -100px;
        opacity: 0
    }

    to {
        bottom: 0px;
        opacity: 1
    }
}

@keyframes animatebottom {
    from {
        bottom: -100px;
        opacity: 0
    }

    to {
        bottom: 0;
        opacity: 1
    }
}

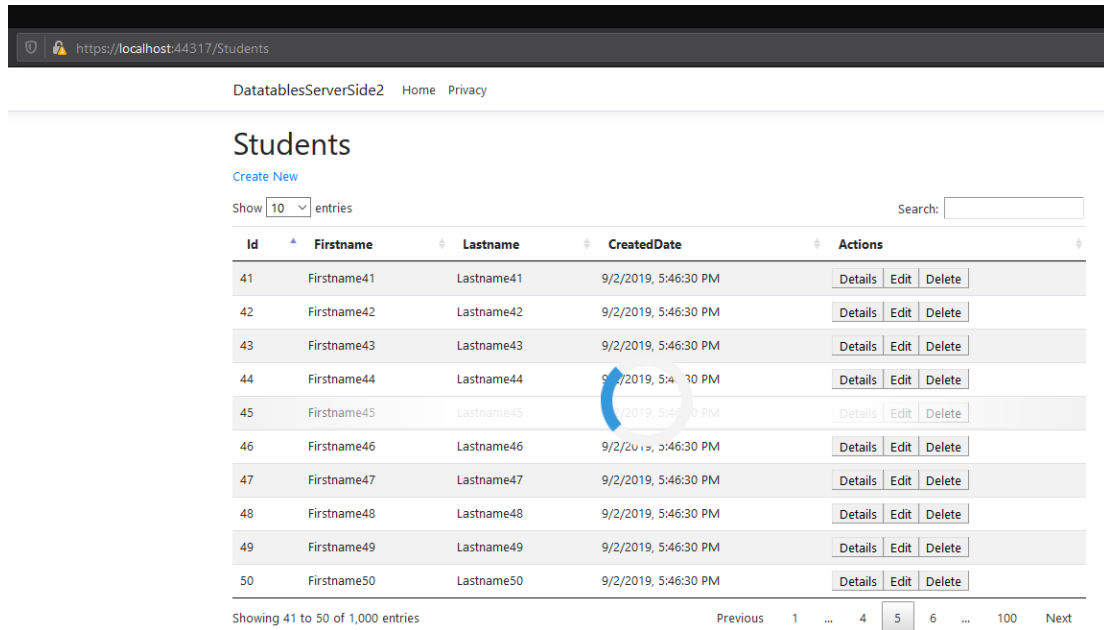
/* End - Style of Centered spinner */
spinner.
```

5. Run Project.

Run the Project by clicking IIS Express (or by pressing key F5)

See the result as below:

Figure 1: Result screen during the data load after click on Page 6



https://localhost:44317/Students

DatatablesServerSide2 Home Privacy

Students

[Create New](#)

Show entries Search:

| Id | Firstname | Lastname | CreatedDate | Actions |
|----|-------------|------------|----------------------|---|
| 41 | Firstname41 | Lastname41 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 42 | Firstname42 | Lastname42 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 43 | Firstname43 | Lastname43 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 44 | Firstname44 | Lastname44 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 45 | Firstname45 | Lastname45 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 46 | Firstname46 | Lastname46 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 47 | Firstname47 | Lastname47 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 48 | Firstname48 | Lastname48 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 49 | Firstname49 | Lastname49 | 9/2/2019, 5:46:30 PM | Details Edit Delete |
| 50 | Firstname50 | Lastname50 | 9/2/2019, 5:46:30 PM | Details Edit Delete |

Showing 41 to 50 of 1,000 entries Previous 1 ... 4 **5** 6 ... 100 Next

- Update detailsClick js function at Views/Students/Index.cshmtl to display Details page when button clicked.

```
//var rowID = $(obj).attr('id');
// Get Id of clicked student
var studentId = $(obj).closest('tr').find('td:first').html();
//Go to details page of student
window.location.replace("Students/Details/" + studentId);
```