Version: 4.2.2

Filter Component

Article 08/28/2023

The dReveal filter component provides features to filter the data displayed based on different criteria in the grid component.

Controller

This model binder structure is mapped via the `DRevealDataStructure` class, that returns the current state of the filter component. This state contains the filter values applied. The state received after mapping processing must be assigned to the Grid View Model method called `SetState`.

The following code demonstrates how to invoke the method mentioned before:

```
[Route("GridReport")]
public ActionResult GridReport(
    [ModelBinder(typeof(DRevealDataStructureBinder))] DRevealDataStructure gridState
)
{
    GridViewModel viewModel = new GridViewModel(
        memoryStream: fileStream,
        connectionString: "Data Source=MyServer; Initial Catalog=myDatabase; User ID=myUser; Password=myPassword",
        gridId: "grid_id",
        htmlFormId: "main_form",
        rowsPerPage: 15,
        enableColumnFinderFeature: true,
        isGridPagerSimpleMode: true,
        fullGridControllerActionPath: HttpContext.Request.Url.AbsolutePath,
        gridActionName: string.Empty,
        drillThroughControllerAction: string.Empty,
        saveGridControllerAction: string.Empty,
        exportToExcelGridControllerAction: string.Empty
    );
```

```
20
21      // Set dReveal Data Strucuture
22      viewModel.SetState(gridState);
23      viewModel.LoadInitialConfiguration();
24
25      return View("GridReport", viewModel);
26  }
```

View

The dReveal filter component provides a method to be able of visualizing it throughout a razor page. It is possible to invoke this method following manner:

```
@Html.InfoArch().BasicFilter(Model.FiltersSettings).BindBasicFilterExtension().R
```

The `FiltersSettings` is the parameter required by the method mentioned above, and prepared in the controller to load it in the view.The following code shows how to use the method in a view:

```
1  <div id="filter-component">
2      @{
3
   Html.InfoArch().BasicFilter(Model.FiltersSettings).BindBasicFilterExtension()
4      }
5      <button id="clear-button" type="button">Clear</button>
6      <button id="apply-button" type="button">Apply</button>
7  </div>
```

> **NOTE**
>
> These HTML buttons must be placed within the form tag.

It is important to mention that it is needed to implement the click events for the **Clear** and **Apply** HTML buttons. Additionally, the dReveal filter component provides a couple of methods on the client side. These methods are used to apply the filter values selected, and the other one to clear them.

Apply Filters

The method to apply filters is used to send the filter values from the client side to the controller action method in the server side described previously.

The apply filter method received two parameters, one of them is the HTML button that initiate the filter apply's event, and the other one is a validation function to check if some errors are throwed once the internal validation process has ended.

The following code demonstrates how to implement this method:

```
dRFilterComponent.send($(button), function (errors) {
    if (errors.length > 0){
        alert(`There are errors in the filters:${errors.toString()}`);
    }
});
```

Clean Filters

The method to clean filters is used to clear the filter values selected in the different filters displayed. The clean filters method received as a unique parameter, the HTML button that initiate the event for cleaning.

The following code demonstrates how to implement this method:

```
dRFilterComponent.clear($(button));
```

Example

The following code is a full sample of dReveal filter component integration:

```
@using InfoArch.Web.Mvc.UI
@using InfoArch.Web.Mvc.Grid
@using InfoArch.Web.Mvc.Enums

@model GridViewModel
@Html.InfoArch().RegisterRequiredThirdPartyLibraries(
    RegisterThirdPartyType.Grid,
    RegisterThirdPartyElements.JQuery
)
@Html.InfoArch().GetStyleSheets(
    new StyleSheet { ExtensionType = ExtensionType.Grid }
```

```
)
@Html.InfoArch().GetScripts(
    new Script { ExtensionSuite = ExtensionSuite.Grid }
)

<style>
    #grid-container {
        height: 100vh;
        width: 80%;
    }

    #filter-container {
        width: 20%;
        height: 100vh;
    }

    #reporting-grid {
        display: flex;
        flex-direction: row;
    }
</style>

<div >
    @using (@Html.BeginForm("GridReport", "Reporting", FormMethod.Post, new {
"grid-form" }))
    {
        @Html.AntiForgeryToken()
        <div id="reporting-grid">
            <section id="filter-container">
                <div id="filter-component">
                    @{

Html.InfoArch().BasicFilter(Model.FiltersSettings).BindBasicFilterExtension()
                    }
                    <button id="clear-button" type="button">Clear</button>
                    <button id="apply-button" type="button">Apply</button>
                </div>
            </section>
            <section id="grid-container">
                @Html.InfoArch().Grid(@Model.GridSettings).GetHtml()
            </section>
        </div>
    }
</div>
<script>
    document.addEventListener("DOMContentLoaded", function () {
        let applyButton = document.getElementById("apply-button");
```

```
        let clearButton = document.getElementById("clear-button");

        applyButton.addEventListener("click", (event) => {
            dRFilterComponent.send($(event.target), function (errors) {
                if (errors.length > 0){
                    alert(`There are errors in the filters:${errors.toString(
                }
            });
        })

        clearButton.addEventListener("click", () => {
            dRFilterComponent.clear($(clearButton));
        })

    });
</script>
```

> It is important to take in account that the filter component is into a form tag **<form>**.

TIP

Security Tip: The @Html.AntiForgeryToken() method could be added to prevent CSRF attacks. This method is provided by .Net Framework. For further information, visit the official Microsoft documentation.

> **TIP**