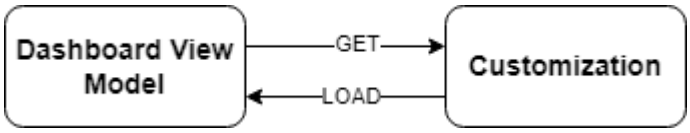
Version: 4.2.2

Customization

Article 08/28/2023

The dReveal dashboard component provides a feature to get or load a specific state of it



The dashboard state is a string object that could be stored in a binary file, database, or any storage mechanism.

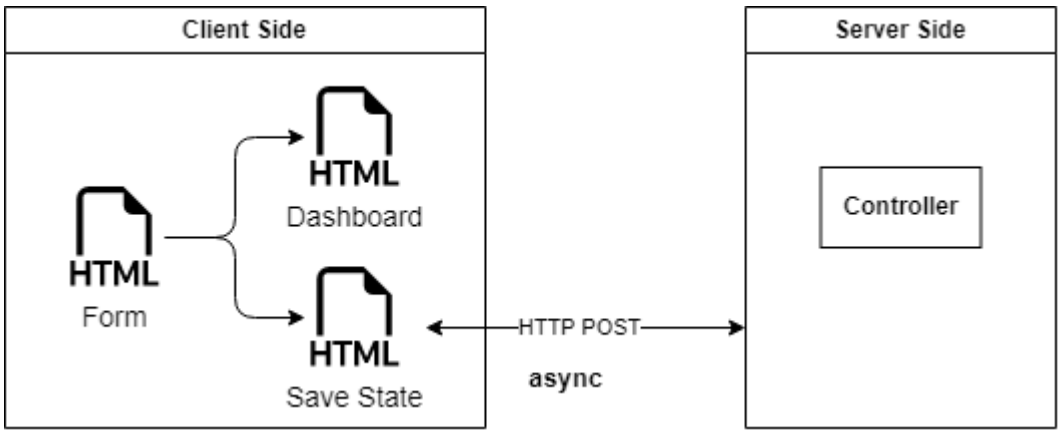
> The scope of the dashboard elements that can be stored in the state are filters applied and modifications with the edit mode feature.

Dashboard State Methods

dReveal dashboard component provides throughout the `DashboardViewModel` class, methods to get and load the dashboard state.

Get Customization

The `DashboardViewModel` class provides a method to get the dashboard customization (dashboard state), called `Serialize`. For further information about serialize method, see the Dashboard View Model documentation.



The following code demonstrate how implement this method:

```
[HttpPost]
public ActionResult
DashboardReport([ModelBinder(typeof(DRevealDataStructureBinder))]
DRevealDataStructure dashboardState)
{
    DashboardViewModel viewModel = new DashboardViewModel(
        ...
         // Set controller/action to use in javascript function
         saveDashboardControllerAction: RouteData.Values["Controller"]+"/"+
"SaveState",
        ...
    );

    viewModel.SetState(dashboardState);
    viewModel.LoadInitialConfiguration();
    return View("DashboardReport", viewModel);
}

public ActionResult
SaveState([ModelBinder(typeof(DRevealDataStructureBinder))]
DRevealDataStructure dashboardState)
{
    // create DashboardViewModel from DRevealDataStructure
    DashboardViewModel viewModel =
DashboardViewModel.GetCustomizationViewModel(dashboardState);
    // Serialize
    string state = viewModel.Serialize();
    // save using the storage mechanism choosen.
    ....
    return new EmptyResult();
}
```

In the client side is required to add the following method for state saving asynchronously:

```
saveStateButton.addEventListener("click", () => {
    dr.dashboardSave();
})
```

> **NOTE**

> This function performs an Ajax request to the controller/Action set in `DashboardViewModel.saveDashboardControllerAction`.

Load Customization

The `DashboardViewModel` class provides a method to load the dashboard customization (dashboard state), called `AssignUserCustomizationDefinitionModel()` and after applying the `LoadInitialConfiguration()`, the `UpdateCustomization()` method is required. For further information about these methods, see the Dashboard View Model documentation.

The load customization process can load a previous version of a dashboard state once the DashboardViewModel will be configured. The following code demonstrates how implement this method:

```
[HttpPost]
public ActionResult DashboardReport()
{
    DashboardViewModel viewModel = new DashboardViewModel(
        memoryStream: fileStream,
        connectionString: "Data Source=MyServer; Initial Catalog=myDatabase; User ID=myUser; Password=myPassword",
        dashboardId: "dasboard-id",
        htmlFormId: "dasboard-from",
        fullGridControllerActionPath: HttpContext.Request.Url.AbsolutePath,
        useNewExportToImagePro: true,
        filterComponentParentId: "filter-component",
        drillThroughControllerAction: string.Empty,
        // Set controller/action to use in javascript function
        saveDashboardControllerAction: RouteData.Values["Controller"]+"/"+"SaveState",
        excelXlsxExportControllerAction: string.Empty
    );
    // set customization
    viewModel.AssignUserCustomizationDefinitionModel(dashboardState);
    viewModel.LoadInitialConfiguration();
    // apply customization
    viewModel.UpdateCustomization();
    return View("DashboardReport", viewModel);
}
```