

23-02 Resources view – Sorting – Programmatic column sorting

This example demonstrates how table column sorting can be controlled programmatically. The main advantage is that it allows sorting based on column content that is not visible in the table.

The sort is a display sort and is only maintained as long as the sort function is active. During this time, however, the sort is dynamic — i.e., changes to the data in the relevant column automatically update the sort order. If there are identical values, the initial order is preserved.

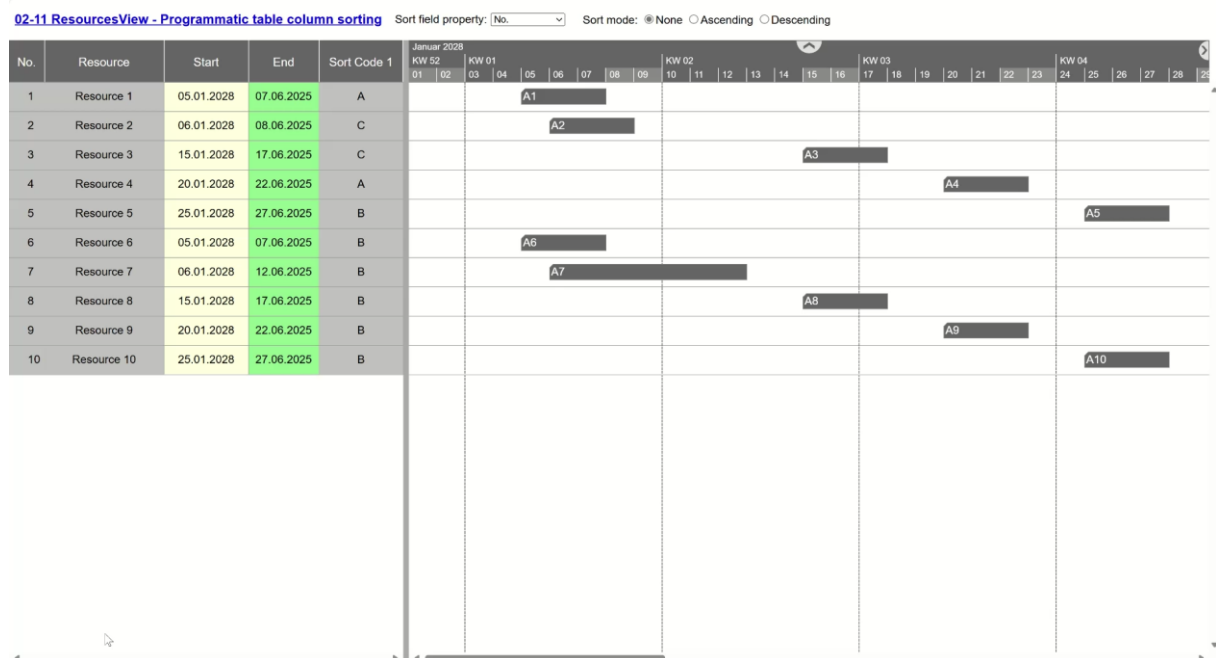
Only one column can be sorted at a time. When switching from one sorted column to another, the sort is applied to the new column. The starting point for sorting is the original order.

Two options must be set to use this functionality:

```
resourceRowSortCodePropertyName: "No.",  
resourceRowSortMode: RowSortMode.Descending,
```

You need to specify the name of the activity property to sort by. Then, choose whether the sorting should be ascending or descending. To disable sorting and restore the original order, set the value of **resourceRowSortMode** to **None**.

Setting the option **sortingIndicatorVisible** to **true** displays a sorting arrow if the sort field is visible in the table.



```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <title>02-11 NETRONIC VSW SE</title>  
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>  
  <link href="../../VSWidget/nwaf-apptools.min.css" rel="stylesheet"/>  
  <link href="../../VSWidget/nwaf-table.min.css" rel="stylesheet"/>  
  <link href="../../VSWidget/nwaf-gantt.min.css" rel="stylesheet"/>  
  <link href="../../VSWidget/nwaf-rab.min.css" rel="stylesheet"/>  
  <script src="https://cdn.jsdelivr.net/npm/hammerjs@2.0.8/hammer.min.js"></script>
```



```

        en: {
            year: "numeric",
            month: "2-digit",
            day: "2-digit"
        },
    },
    resourceRowSortCodePropertyName: "No",
    resourceRowSortMode: RowSortMode.None,
    sortingIndicatorVisible: true,
    fixedTableColumnWidth: 0,
};

const vsWidget = Miscellaneous.instantiateVSWidget(document.querySelector("#VSWidget"), options);

const tableRowDefinitions = createTableRowDefinitions();
const allocations = createAllocations();
const resources = createResources();
vsWidget.addTableRowDefinitions(tableRowDefinitions);
vsWidget.addResources(resources);
vsWidget.addAllocations(allocations);
vsWidget.render();

function createTableRowDefinitions() {
    const tableRowDefinitions = [
        {
            ID: "CustomRowDef",
            CellDefinitions: [
                {
                    TitleText: "No.",
                    TextSource: "my_No",
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 50,
                },
                {
                    TitleText: "Resource",
                    TextSource: "my_Name",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 170,
                },
                {
                    TitleText: "Start",
                    TextFormat: "{my_Start:date:en}",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    BackgroundColor: "lightyellow",
                    Width: 120,
                },
                {
                    TitleText: "End",
                    TextSource: "End",
                    TextFormat: "{End:date:en}",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    BackgroundColor: "lightgreen",
                    Width: 100,
                },
                {
                    TitleText: "Working Days",
                    TextSource: "my_WorkingDays",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 120,
                },
                {
                    TitleText: "Sort Code 1",
                    TextSource: "my_SortCode1",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 120,
                },
            ],
        },
    ],
};

```

```

        return tableRowDefinitions;
    }

    function createResources() {
        const millisecondsPerDay = 24 * 60 * 60 * 1000;
        const sortCode1 = ["A", "C", "C", "A", "B", "B", "B", "B", "B", "B"];
        const sortCode2 = ["a", "b", "c", "c", "b", "c", "d", "d", "a", "c"];
        const resources = [];

        for (let i = 1; i <= 10; i++) {
            const start = allocations[i].Entries[0].Start;
            const end = allocations[i].Entries[0].End;
            const resource = {
                ID: `Res${i}`,
                my_Name: `Resource ${i}`,
                my_No: i,
                my_SortCode1: sortCode1[i - 1],
                my_SortCode2: sortCode2[i - 1],

                my_Start: start,
                my_End: end,
                my_WorkingDays: allocations[i].my_WorkingDays,
                my_RunningDays: end.getDate() - start.getDate() + 1,
                my_EndDay: new Date(end - 1 * millisecondsPerDay), // End day is the day before the end date
            };
            resources.push(resource);
        }

        return resources;
    }

    function createAllocations() {
        const allocations = [];
        const startDays = [5, 6, 15, 20, 25, 5, 6, 15, 20, 25];
        const workingDays = [3, 3, 3, 3, 3, 3, 7, 3, 3, 3];

        for (let i = 0; i < startDays.length; i++) {
            const allocation = {
                ID: `Alloc${i + 1}`,
                ResourceID: `Res${i + 1}`,
                BarText: `A${i + 1}`,
                my_WorkingDays: workingDays[i],
                Entries: [
                    {
                        Start: new Date(2028, 0, startDays[i]),
                        End: new Date(2028, 0, startDays[i] + workingDays[i]),
                    },
                ],
            };
            allocations.push(allocation);
        }

        return allocations;
    }

    function changeSortField() {
        const selectElement = document.getElementById("dataFields");
        vsWidget.option("resourceRowSortCodePropertyName", selectElement.value);
    }

    function changeSortMode(event) {
        const radio = event.target;

        const rowSortMode = [
            RowSortMode.None,
            RowSortMode.Ascending,
            RowSortMode.Descending,
        ];

        vsWidget.option("resourceRowSortMode", rowSortMode[radio.value]);
    }
});
</script>

```

```
</body>  
</html>
```