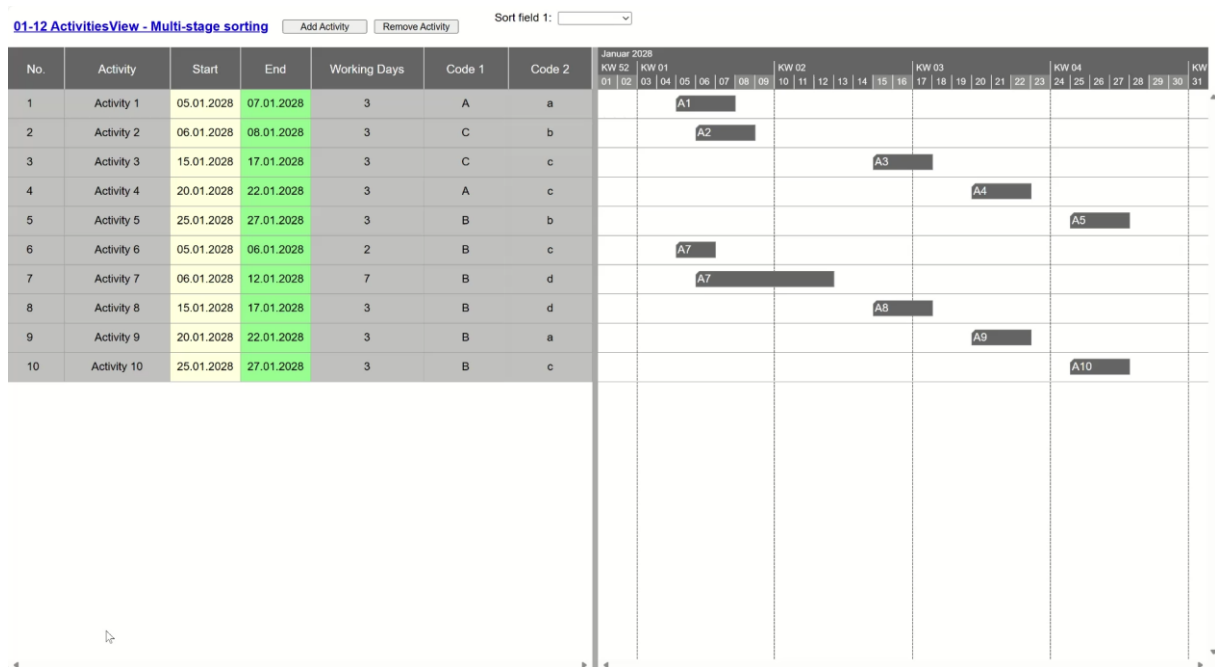


13-03 Activities view – Sorting – Custom multi-level sorting

This example shows how a customised multi-level sorting is implemented. Columns that are not displayed in the table can also be included in the sorting.



The core routine for sorting is the **updateRanks** function, which is supported by the **smartCompare** function. The **smartCompare** function ensures that type-specific and country-specific sorting can be carried out correctly.

The sorting result is stored in the property rank for each activity. The sorting in the Visual Scheduling widget is set to this data field and is not changed further down the line.

```
activityRowSortCodePropertyName: "rank",
activityRowSortMode: RowSortMode.Ascending,
```

To continue to enable sorting by clicking on columns, the

```
interactiveSwitchingOfSortOrderEnabled: true,
```

option must be activated. As a result, the callback function is called internally when the column is clicked, in which we still have to react accordingly. The default column sorting must be switched off by

```
args.cancel = true;
```

Then we need to add the arrows for sorting direction and the sort criteria sequence to the texts in the table header. When clicking on the columns, note that a sequence is run through: Ascending sort, descending sort and no sort.

In the **onRowSortingChangeRequest** callback, first the sorting and then the activities must be updated.

```
updateRanks(activities, sortDef.FieldNames, sortDef.SortOrders, false);
vsWidget.updateActivities(activities);
vsWidget.render();
```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <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>
  <script src="https://cdn.jsdelivr.net/npm/d3@7.9.0/dist/d3.min.js"></script>
  <script src="https://cdn.jsdelivr.net/npm/tinycolor2@1.6.0/cjs/tinycolor.min.js"></script>
  <script src="../../VSWidget/nwaf-apptools.min.js"></script>
  <script src="../../VSWidget/nwaf-table.min.js"></script>
  <script src="../../VSWidget/nwaf-gantt.min.js"></script>
  <script src="../../VSWidget/nwaf-rab.min.js"></script>
  <script src="../../LicenseKey.js"></script>
  <script src="Miscellaneous.js"></script>
</head>
<body>
  <div id="toolbar">
    <h3>
      <a href="docs/13-03_ActivitiesView-Sorting-Custom_multi-level_sorting.pdf"
        target="_blank">13-03_Activities view - Sorting - Custom multi-level sorting</a>
    </h3>
    <input id="addActivity" type="button" value="Add Activity"
      style="margin-left: 15px; height: 20px; width: 120px"/>
    <input id="removeActivity" type="button" value="Remove Activity"
      style="margin-left: 5px; height: 20px; width: 120px"/>
    <table style="margin-left: 30px">
      <tr>
        <td id="field1">
          <label for="dataFields1" style="margin-left: 20px">Sort field 1:</label>
          <select id="dataFields1" name="fields1">
            <option value="" selected></option>
            <option value="my_No">No.</option>
            <option value="my_Name">Activity</option>
            <option value="Start">Start</option>
            <option value="my_EndDay">End</option>
            <option value="my_WorkingDays">Working Days</option>
            <option value="my_SortCode1">Code 1</option>
            <option value="my_SortCode2">Code 2</option>
          </select>
        </td>
        <td id="sortOrder1">
          <label style="margin-left: 20px">Sort mode 1:</label>
          <label>
            <input id="none1" type="radio" name="sortMode1" value="0" checked/>
            None
          </label>
          <label>
            <input id="ascending1" type="radio" name="sortMode1" value="1"/>
            Ascending
          </label>
          <label>
            <input id="descending1" type="radio" name="sortMode1" value="-1"/>
            Descending
          </label>
        </td>
      </tr>
      <tr id="row2">
        <td id="field2">
          <label for="dataFields2" style="margin-left: 20px">Sort field 2:</label>
          <select id="dataFields2" name="fields2">
            <option value="" selected></option>
            <option value="No">No.</option>
            <option value="my_Name">Activity</option>
            <option value="Start">Start</option>
            <option value="my_EndDay">End</option>
            <option value="my_WorkingDays">Working Days</option>
            <option value="my_SortCode1">Code 1</option>
            <option value="my_SortCode2">Code 2</option>
          </select>
        </td>
        <td id="sortOrder2">
          <label style="margin-left: 20px">Sort mode 2:</label>

```

```

        <label>
            <input id="none2" type="radio" name="sortMode2" value="0" checked/>
            None
        </label>
        <label>
            <input id="ascending2" type="radio" name="sortMode2" value="1"/>
            Ascending
        </label>
        <label>
            <input id="descending2" type="radio" name="sortMode2" value="-1"/>
            Descending
        </label>
    </td>
</tr>
</table>
</div>
<div id="VSWidget"></div>
<script>
    Miscellaneous.ready(() => {
        const { BarDragModes, BarSortMode, HorizontalAlignment,
            RowDesigns, RowSortMode } = netronic.nVSW;

        document.getElementById("addActivity").addEventListener("click", addActivity);
        document.getElementById("removeActivity").addEventListener("click", removeActivity);
        document.getElementById("none1").addEventListener("click", () => { changeSortMode(0); });
        document.getElementById("ascending1").addEventListener("click", () => { changeSortMode(0); });
        document.getElementById("descending1").addEventListener("click", () => { changeSortMode(0); });
        document.getElementById("none2").addEventListener("click", () => { changeSortMode(1); });
        document.getElementById("ascending2").addEventListener("click", () => { changeSortMode(1); });
        document.getElementById("descending2").addEventListener("click", () => { changeSortMode(1); });
        document.getElementById("dataFields1").addEventListener("change", () => { changeSortField(0); });
        document.getElementById("dataFields2").addEventListener("change", () => { changeSortField(1); });

        const timeAreaStart = new Date(2028, 0, 1);
        const timeAreaEnd = new Date(2028, 1, 1);

        const sortDef = {
            FieldNames: [ "", "" ],
            SortOrders: [ -1, 0 ],
        };

        const options = {
            licenseKey: window.VSW_LICENSE_KEY,

            start: timeAreaStart,
            end: timeAreaEnd,

            defaultActivityExpandedRowDesign: RowDesigns.Bars,
            defaultActivityCollapsedRowDesign: RowDesigns.Bars |
                RowDesigns.BarsInHiddenDescendantRows |
                RowDesigns.BarsStacked,
            activityBarSortModeForStackedRowDesign: BarSortMode.StartAndEnd,
            defaultActivityAllowedBarDragModes: BarDragModes.None,
            defaultActivityBarSelectable: false,
            defaultActivityRowSelectable: false,
            multipleSelectionEnabled: false,
            tableRowDefinitionIDForTitleInActivitiesView: "CustomRowDef",
            defaultActivityTableRowDefinitionID: "CustomRowDef",
            intlDateTimeFormatOptionsMap: {
                dateFmt: {
                    year: "numeric",
                    month: "2-digit",
                    day: "2-digit"
                },
            },
        },
        interactiveSwitchingOfSortOrderEnabled: true,
        activityRowSortCodePropertyName: "rank",
        activityRowSortMode: RowSortMode.Ascending,
        animationDuration: 0,

        onRowSortingChangeRequested: function (args) {
            sortDef.FieldNames[1] = "";
            sortDef.SortOrders[1] = 0;
        }
    });

```

```

    if (args.sortCodeSource !== sortDef.FieldNames[0]) {
        sortDef.FieldNames[0] = args.sortCodeSource;
        sortDef.SortOrders[0] = 1;
    } else {
        if (sortDef.SortOrders[0] === 0)
            sortDef.SortOrders[0] = 1;
        else if (sortDef.SortOrders[0] === 1)
            sortDef.SortOrders[0] = -1;
        else if (sortDef.SortOrders[0] === -1)
            sortDef.SortOrders[0] = 0;
    }
    updateControls(sortDef);
    updateRanks(activities, sortDef.FieldNames, sortDef.SortOrders, false);
    vsWidget.updateActivities(activities);
    vsWidget.render();
    args.cancel = true;
},
});

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

const tableRowDefinitions = createTableRowDefinitions();
const activities = createActivities();
updateRanks(activities, "my_No", 1, false);
vsWidget.addTableRowDefinitions(tableRowDefinitions);
vsWidget.addActivities(activities);

updateControls(sortDef);

vsWidget.render();
vsWidget.fitTimeAreaIntoView(timeAreaStart, timeAreaEnd);

function createTableRowDefinitions() {
    const tableRowDefinitions = [
        {
            ID: "CustomRowDef",
            CellDefinitions: [
                {
                    Title: "No.",
                    TextSource: "my_No",
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 80,
                },
                {
                    Title: "Activity",
                    TextSource: "my_Name",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 150,
                },
                {
                    Title: "Start",
                    TextSource: "Start",
                    TextFormat: "{Start:date:dateFmt}",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    BackgroundColor: "lightyellow",
                    Width: 100,
                },
                {
                    Title: "End",
                    TextSource: "my_EndDay",
                    TextFormat: "{my_EndDay:date:dateFmt}",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    BackgroundColor: "lightgreen",
                    Width: 100,
                },
                {
                    Title: "Working Days",
                    TextSource: "my_WorkingDays",
                    HorizontalTitleAlignment: HorizontalAlignment.Center,
                    HorizontalAlignment: HorizontalAlignment.Center,
                    Width: 160,
                }
            ]
        }
    ];
}

```

```

    },
    {
      Title: "Code 1",
      TextSource: "my_SortCode1",
      HorizontalTitleAlignment: HorizontalAlignment.Center,
      HorizontalAlignment: HorizontalAlignment.Center,
      Width: 120,
    },
    {
      Title: "Code 2",
      TextSource: "my_SortCode2",
      HorizontalTitleAlignment: HorizontalAlignment.Center,
      HorizontalAlignment: HorizontalAlignment.Center,
      Width: 120,
    },
  ],
},
];

return tableRowDefinitions;
}

function updateControls() {
  if (sortDef.FieldNames[0] === "") {
    sortDef.FieldNames[1] = "";
    sortDef.SortOrders[0] = 0;
    sortDef.SortOrders[1] = 0;
    document.getElementById("dataFields1").value = sortDef.FieldNames[0];
    document.getElementById("dataFields2").value = sortDef.FieldNames[1];
    document.querySelector('input[name="sortMode1"]:checked').value = sortDef.SortOrders[0];
    document.querySelector('input[name="sortMode2"]:checked').value = sortDef.SortOrders[1];
    document.getElementById("field1").style.visibility = "visible";
    document.getElementById("field1").style.display = "block";
    document.getElementById("sortOrder1").style.visibility = "hidden";
    document.getElementById("sortOrder2").style.visibility = "hidden";
    document.getElementById("field2").style.visibility = "hidden";
  } else {
    if (sortDef.FieldNames[1] === "")
      sortDef.SortOrders[1] = 0;

    if (sortDef.SortOrders[0] === 0)
      document.querySelector('input[name="sortMode1"][value="0"]').checked = true;
    else if (sortDef.SortOrders[0] === 1)
      document.querySelector('input[name="sortMode1"][value="1"]').checked = true;
    else if (sortDef.SortOrders[0] === -1)
      document.querySelector('input[name="sortMode1"][value="-1"]').checked = true;

    if (sortDef.SortOrders[1] === 0)
      document.querySelector('input[name="sortMode2"][value="0"]').checked = true;
    else if (sortDef.SortOrders[1] === 1)
      document.querySelector('input[name="sortMode2"][value="1"]').checked = true;
    else if (sortDef.SortOrders[1] === -1)
      document.querySelector('input[name="sortMode2"][value="-1"]').checked = true;

    document.getElementById("dataFields1").value = sortDef.FieldNames[0];
    document.getElementById("sortOrder1").style.visibility = "visible";
    document.getElementById("field2").style.visibility = "visible";
    document.getElementById("field2").style.display = "block";
    document.getElementById("sortOrder2").style.visibility = sortDef.FieldNames[1].length > 0 ?
      "visible" : "hidden";
  }

  let prefix1 = ["↓", "", "↑"][sortDef.SortOrders[0] + 1];
  let prefix2 = ["↓", "", "↑"][sortDef.SortOrders[1] + 1];

  const tableRowDefinition = tableRowDefinition[0];
  tableRowDefinition.CellDefinitions.forEach((tableCellDefinition) => {
    tableCellDefinition.TitleText = tableCellDefinition.Title;

    if (tableCellDefinition.TextSource === sortDef.FieldNames[0])
      tableCellDefinition.TitleText = `${prefix1}${tableCellDefinition.Title} (1)`;

    if (tableCellDefinition.TextSource === sortDef.FieldNames[1])
      tableCellDefinition.TitleText = `${prefix2}${tableCellDefinition.Title} (2)`;
  });
}

```

```

});

vsWidget.updateTableRowDefinitions(tableRowDefinitions);
}

function smartCompare(a, b, direction = 1, locale = "en-US") {
  if (a === null && b === null)
    return 0;
  if (a === null)
    return 1;
  if (b === null)
    return -1;

  // Date comparison
  if (a instanceof Date && b instanceof Date)
    return direction * (a.getTime() - b.getTime());

  // Date-Strings
  const aDate = typeof a === "string" ? new Date(a) : null;
  const bDate = typeof b === "string" ? new Date(b) : null;
  const aValidDate = aDate instanceof Date && !isNaN(aDate);
  const bValidDate = bDate instanceof Date && !isNaN(bDate);
  if (aValidDate && bValidDate)
    return direction * (aDate.getTime() - bDate.getTime());

  // Numbers comparison
  if (typeof a === "number" && typeof b === "number")
    return direction * (a - b);

  return direction * a.toString().localeCompare(b.toString(), locale, {
    numeric: true,
    sensitivity: "base",
    ignorePunctuation: true,
  });
}

// Updates or adds to the ranking (`rank`) in an array of objects.
// list - Array of objects
// property - The property to be sorted by or null
//           If null, the existing rank is used for sorting (if mutateOrder = true)
// direction - Sorting direction
//             -1 = descending sort
//             1 = ascending sort
//             0 = No new rank assignment, but sorting of array according
//               to existing rank if mutateOrder == true
// mutateOrder - Whether the array should be returned sorted.

function updateRanks(list, properties, directions, mutateOrder) {
  // Special case: direction === 0 → sort by rank
  if (Array.isArray(directions) === false && directions === 0) {
    if (mutateOrder)
      list.sort((a, b) => smartCompare(a.rank, b.rank, 1));
    return;
  }
  if (!Array.isArray(properties))
    properties = [properties];
  if (!Array.isArray(directions))
    directions = [directions];

  const sortedIndices = list
    .map((_, index) => index)
    .sort((aIdx, bIdx) => {
      for (let i = 0; i < properties.length; i++) {
        const prop = properties[i];
        const result = smartCompare(list[aIdx][prop], list[bIdx][prop], directions[i] ?? 1);
        if (result !== 0)
          return result;
      }
      return 0;
    });

  for (let i = 0; i < sortedIndices.length; i++)
    list[sortedIndices[i]].rank = i + 1;
}

```

```

    if (mutateOrder) {
      const sorted = sortedIndices.map((i) => list[i]);
      for (let i = 0; i < list.length; i++)
        list[i] = sorted[i];
    }
  }

function createActivities() {
  const activities = [
    {
      ID: "Act1",
      my_Name: "Activity 1",
      BarText: "A1",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 3,
      my_SortCode1: "A",
      my_SortCode2: "a",
      my_No: 1,
    },
    {
      ID: "Act2",
      my_Name: "Activity 2",
      BarText: "A2",
      Start: new Date(2028, 0, 6),
      my_WorkingDays: 3,
      my_SortCode1: "C",
      my_SortCode2: "b",
      my_No: 2,
    },
    {
      ID: "Act3",
      my_Name: "Activity 3",
      BarText: "A3",
      Start: new Date(2028, 0, 15),
      my_WorkingDays: 3,
      my_SortCode1: "C",
      my_SortCode2: "c",
      my_No: 3,
    },
    {
      ID: "Act4",
      my_Name: "Activity 4",
      BarText: "A4",
      Start: new Date(2028, 0, 20),
      my_WorkingDays: 3,
      my_SortCode1: "A",
      my_SortCode2: "c",
      my_No: 4,
    },
    {
      ID: "Act5",
      my_Name: "Activity 5",
      BarText: "A5",
      Start: new Date(2028, 0, 25),
      my_WorkingDays: 3,
      my_SortCode1: "B",
      my_SortCode2: "b",
      my_No: 5,
    },
    {
      ID: "Act6",
      my_Name: "Activity 6",
      BarText: "A7",
      Start: new Date(2028, 0, 5),
      my_WorkingDays: 2,
      my_SortCode1: "B",
      my_SortCode2: "c",
      my_No: 6,
    },
    {
      ID: "Act7",
      my_Name: "Activity 7",
      BarText: "A7",
      Start: new Date(2028, 0, 6),

```

```

        my_WorkingDays: 7,
        my_SortCode1: "B",
        my_SortCode2: "d",
        my_No: 7,
    },
    {
        ID: "Act8",
        my_Name: "Activity 8",
        BarText: "A8",
        Start: new Date(2028, 0, 15),
        my_WorkingDays: 3,
        my_SortCode1: "B",
        my_SortCode2: "d",
        my_No: 8,
    },
    {
        ID: "Act9",
        my_Name: "Activity 9",
        BarText: "A9",
        Start: new Date(2028, 0, 20),
        my_WorkingDays: 3,
        my_SortCode1: "B",
        my_SortCode2: "a",
        my_No: 9,
    },
    {
        ID: "Act10",
        my_Name: "Activity 10",
        BarText: "A10",
        Start: new Date(2028, 0, 25),
        my_WorkingDays: 3,
        my_SortCode1: "B",
        my_SortCode2: "c",
        my_No: 10,
    },
    ],
    ];

    const millisecondsPerDay = 24 * 60 * 60 * 1000;
    for (const activity of activities) {
        activity.End = new Date(activity.Start.getTime() +
            (activity.my_WorkingDays ?? 1) * millisecondsPerDay);
        activity.my_EndDay = new Date(activity.Start.getTime() +
            ((activity.my_WorkingDays ?? 1) - 1) * millisecondsPerDay);
    }

    return activities;
}

function addActivity() {
    const activity = {
        ID: `Act${activities.length + 1}`,
        my_Name: `Activity ${activities.length + 1}`,
        BarText: `A${activities.length + 1}`,
        Start: new Date(2028, 0, 5),
        End: new Date(2028, 0, 20),
        my_WorkingDays: 3,
        my_SortCode1: "B",
        my_SortCode2: "x",
        my_No: activities.length + 1,
    };

    const millisecondsPerDay = 24 * 60 * 60 * 1000;
    activity.End = new Date(activity.Start.getTime() +
        activity.my_WorkingDays * millisecondsPerDay);
    activity.my_EndDay = new Date(activity.Start.getTime() +
        (activity.my_WorkingDays - 1) * millisecondsPerDay);

    activities.push(activity);
    vsWidget.addActivities([activity]);
    updateRanks(activities, sortDef.FieldNames, sortDef.SortOrders, false);
    vsWidget.updateActivities(activities);
    vsWidget.render();
}

```



```

function removeActivity() {
  const activity = activities.pop();
  if (activity) {
    vsWidget.removeActivities([activity]);
    vsWidget.render();
  }
}

function changeSortField(index) {
  sortDef.FieldNames[index] = index === 0 ? document.getElementById("dataFields1").value :
                                          document.getElementById("dataFields2").value;

  updateControls();
  updateRanks(activities, sortDef.FieldNames, sortDef.SortOrders, false);
  vsWidget.updateActivities(activities);
  vsWidget.render();
}

function changeSortMode(index) {
  sortDef.SortOrders[index] = index === 0 ?
    document.querySelector('input[name="sortMode1"]:checked').value :
    document.querySelector('input[name="sortMode2"]:checked').value;

  updateControls();
  updateRanks(activities, sortDef.FieldNames, sortDef.SortOrders, false);
  vsWidget.updateActivities(activities);
  vsWidget.render();
}
});
</script>
</body>
</html>

```