Beehive NetUI Data Grids

Table of contents

1. Overview

The NetUI data grid tag library is a high-level abstraction for rendering grids of data in HTML. This tag library supports rendering uses from simple tables to complex sortable, filterable, pageable, and updatable grids of data. The <netui-data:dataGrid> is part of NetUI's <netui-data:xxx> tag library.

2. A Simple Data Grid

The simplest data grid will render a List of JavaBeans into a basic HTML table. Such a grid might look like:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="petGrid">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

This data grid binds to a List of PetBean objects stored in the PageContext's attribute map and renders an HTML table that is four cells wide (i.e., four columns) containing the petId, name, description, and price fields. The rendered table would appear as:

Simple data grid

The values for the cells are specified with a JSP 2.0 expression and can reference any implicit object. In this case, an expression like `${container.item.petId}` references an implicit object called `container` that the `<netui-data:rows>` tag provides. This implicit object has an `item` property that references the current item from data set bound to the `dataSource` attribute of the `<netui-data:dataGrid>` tag.

3. Data Grid Cells

The data grid tags use cell based rendering. This means that their structure is not defined in terms of columns but rather in terms of cells in the table. This allows for greater flexibility to have cells span columns and rows as needed. Cells can be placed in two regions of the data grid -- the header and the body. The header of the grid is the region that renders at the top of the rows containing data and often has column titles, sorting / filtering links, and paging UI. Cells in the body of the data grid are used to render the grid's data; the `spanCell` tags in the simple example above are an example of this.

3.1. Body Cells

Body cells are used to render rows that display data. For example:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="petGrid">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
        <netui-data:anchorCell value="Details" action="details">
            <netui:parameter name="id" value="${container.item.petId}"/>
        </netui-data:anchorCell>
    </netui-data:rows>
</netui-data:dataGrid>
```

This data grid displays three columns of text with a fourth column containing HTML anchors. Though it's not visible in the image, each anchor contains an HTTP request parameter with name `id` and value `${container.item.petId}`.

Data Grid with Anchor

3.2. Header Cells

Header cells are used to render HTML table cells at the top of a data grid. For example:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="petGrid">
    <netui-data:header>
        <netui-data:headerCell headerText="Pet ID"/>
        <netui-data:headerCell headerText="Name"/>
        <netui-data:headerCell headerText="Description"/>
        <netui-data:headerCell headerText="Price"/>
    </netui-data:header>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

The data grid rendered here will have a single HTML table row that contains header cells with titles for the data columns. This table might appear as:

Data Grid with Header

3.3. Formatting Cells

The NetUI formatter tags can also be applied to data grid cells. For example, a cell

containing a Date object could be formatted using:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:header>
        <netui-data:headerCell headerText="Name"/>
        <netui-data:headerCell headerText="Invoice Date"/>
    </netui-data:header>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.invoiceDate}">
            <netui:formatDate pattern="M/dd/yy"/>
        </netui-data:spanCell>
    </netui-data:rows>
</netui-data:dataGrid>
```

which would render a data grid where the formatted column has dates in M/dd/yy format:

Data Grid with Formatter

4. Adding Styles to the Data Grid

The data grid has three ways to apply styles to the rendered HTML elements including table rows, cells, captions, and the HTML table itself. By default, the data grid renders a set of HTML style class names which can be used in conjunction with a user-provided CSS file to automatically apply styles to the grid. The default CSS names for various data grid regions are listed in a table below.
In addition, the style information can be customized by the page author in two ways via the `style` and `styleClass` attributes. Many of the data grid tags provide these attributes and set the HTML `style` and `class` attributes on rendered HTML elements.

4.1. Style Attributes

The `style` attribute can be used to apply a specific style to a specifc data grid region. For example, the font in a column can be changed to red and bold by using the style `color:red;font-weight:bold;` style on the `spanCell` as:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"
style="color:red;font-weight:bold;"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

renders a data grid that looks like:

Data Grid with Style

4.2. CSS Attributes

CSS files can also be used to apply styles to the data grid by matching names style class attribute names rendered on data grid HTML markup with names in a correctly linked `.css` file. For example, alternating row styles can be applied to a data grid with a `datagrid.css` file containing:

```
.datagrid-even {
background-color: #f0f0f0;
}

.datagrid-odd {
background-color: #ffffc0;
}
```

and a data grid:

```
<link rel="stylesheet" href="datagrid.css">
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

renders a data grid that looks like:

Data Grid with CSS

Notice that the `<netui-data:dataGrid>` tag does not have any special attributes that enable style class rendering; the default style names are always rendered. The default style prefix of `datagrid` can be changed by setting the `dataGrid`'s `styleClassPrefix` attribute. For example, if the above CSS is changed to:

```
.foo-even {
background-color: #f0f0f0;
}

.foo-odd {
background-color: #ffffc0;
}
```

and the data grid changed to:

```
<link rel="stylesheet" href="datagrid.css">
<netui-data:dataGrid dataSource="pageScope.pets" name="pets"
```

```
styleClassPrefix="foo">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

then the data grid will render with the same alternating row colors as above. Rendering of style class names can be completely disabled by setting the `<netui-data:dataGrid>`'s `styleClassPolicy` attribute to the value `none`. For example:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets"
styleClassPolicy="empty">
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

will render a plain HTML table with no HTML `class` or `style` attributes.

The data grid renders several default style names onto various HTML table regions. These values include:

Data grid region	Style name
Table	`datagrid`
Table Caption	`datagrid-caption`
Header Rows	`datagrid-header`
Header Table Cells (`ths`)	`datagrid`
Even Data Rows	`datagrid-even`
Odd Data Rows	`datagrid-odd`
Data Table Cells (`tds`)	`datagrid`
Footer Row	`datagrid-footer`
Header Row Group	`datagrid`
Data Row Group	`datagrid`
Footer Row Group	`datagrid`
Sortable Header Cell Class	`datagrid-sortable`
Sorted Header Cell Class	`datagrid-sorted`

Sorted Data Cell Class	`datagrid-sorted`
Notes: • Row numbering is zero based, so the first data row is even, the second odd, and so on. • When using a `styleClassPrefix` on the `dataGrid` tag, the `datagrid` part of the style name can be replaced with the value of the `styleClassPrefix`.	

5. Customizing the Data Grid's Pager

In order to effectively display a large data set, the data grid provides a paging mechansim that lets the grid display a subset of the data set on each "page". The default pager shows links for navigating to the previous and next pagers. For example, if the `pageScope.pets` data set contains more than ten items, the following data grid:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:configurePager pageSize="2"/>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

generates a pager that contains the current and total page count along with links that navigate to the previous and next pages:

Data Grid with Default Pager

If the data set is smaller than ten items, the default pager will simply show a pager with the message `Page 1 of 1`. To change the default page size, use the `<netui-data:configurePager>` tag to set the page size explicitly. For example:

```
<netui-data:configurePager pageSize="2"/>
```

The appearance of a data grid's can also be configured through the `pageFormat` attribute. Included with the data grid are pagers of two formats:

Pager Name	Pager Format
`prevNext`	previous / next
`firstPrevNextLast`	first / previous next / last

The `firstPrevNextLast` pager can be configured by using the `configurePager` tag as:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:configurePager pagerFormat="firstPrevNextLast"/>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

which will render a pager as:

Data Grid with FPNL Pager

5.1. Defining Custom Pagers

In addition to the pre-defined pager formats, the data grid exposes a set of implicit JavaBean objects into the JSP's PageContext that can be used when rendering a custom pager. These implicit objects provide access to the number of pages, the size of the page, the total size of the data set, and other properties that can be displayed or used to otherwise build a pager. For example, to define a custom label for a pager, the following example uses the `dataGrid` implicit object that is available in the PageContext. This JavaBean exposes a set of state that can be referenced via the JSP 2.0 EL to access additional information about the data grid itself. The `dataGrid` object provides access to the grid's pager state and could be used as:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:configurePager pageSize="2" pagerFormat="firstPrevNextLast"
disableDefaultPager="true"/>
    <netui-data:caption>
        <b>Displaying item ${dataGrid.state.pagerModel.row+1} to
${dataGrid.state.pagerModel.lastRowForPage+1}
            of ${dataGrid.state.pagerModel.dataSetSize} matching items.</b>
        <br/>
        <netui-data:renderPager/>
        <br/>
    </netui-data:caption>
    <netui-data:header>
        <netui-data:headerCell value="Pet Id"/>
        <netui-data:headerCell value="Name"/>
        <netui-data:headerCell value="Description"/>
        <netui-data:headerCell value="Price"/>
    </netui-data:header>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
</netui-data:dataGrid>
```

to render a data grid:

Data Grid with Implicit Object Pager

It can also be useful to disable the data grid's automatic rendering of the pager by setting the `disableDefaultPager` as:

```
<netui-data:configurePager disableDefaultPager="true"/>
```

With this attribute set, the data grid will not render any pager UI. The page author can then explicitly place the pager inside of the grid using the `<netui-data:renderPager>` tag. For example, the pager can be placed in the footer of the data grid with:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:configurePager pageSize="2" disableDefaultPager="true"/>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.petId}"/>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
    <netui-data:footer>
        <tr><td colspan="4"><netui-data:renderPager/></td></tr>
    </netui-data:footer>
</netui-data:dataGrid>
```

With the pager disabled, custom pager UI can be built anywhere inside of the data grid tags using the grid's implicit objects. The following example builds a custom pager that provides "jump to page" functionality via an HTML select box, for example:

```
<netui-data:dataGrid dataSource="pageScope.pets" name="pets">
    <netui-data:configurePager defaultPageSize="2"
pagerFormat="firstPrevNextLast" disableDefaultPager="true"/>
    <netui-data:header>
        <netui-data:headerCell headerText="Name"/>
        <netui-data:headerCell headerText="Description"/>
        <netui-data:headerCell headerText="Price"/>
    </netui-data:header>
    <netui-data:rows>
        <netui-data:spanCell value="${container.item.name}"/>
        <netui-data:spanCell value="${container.item.description}"/>
        <netui-data:spanCell value="${container.item.price}"/>
    </netui-data:rows>
    <netui-data:footer>
        <td colspan="2" align="left">
          <netui-data:renderPager/>
        </td>
        <td colspan="1" align="right">
            <c:if test="${dataGrid.dataSet.size > 0}">
                <form name="pageForm" action="simple-pager-jumptopage.jsp">
                Jump to Page:
                <script type="text/javascript">
                    function doPagerSubmit(comp)
```

```
                    {
                      var form = document.forms["pageForm"];
                      form.method="GET";
                      form.submit();
                    }
                </script>
                <select name="${dataGrid.urlBuilder.pagerRowQueryParamKey}"
onchange="doPagerSubmit(this); return true;">
                    <netui-data:repeater
dataSource="dataGrid.urlBuilder.pagerParamValues">
                        <c:choose>
                            <c:when test="${container.index ==
dataGrid.state.pagerModel.page}">
                                <option value="${container.item}"
selected="true">${container.index+1}</option>
                            </c:when>
                            <c:otherwise>
                                <option
value="${container.item}">${container.index+1}</option>
                            </c:otherwise>
                        </c:choose>
                    </netui-data:repeater>
                </select>
            </c:if>
        </td>
    </netui-data:footer>
</netui-data:dataGrid>
```