Spreadsheet Deep Customizations

Definition

This tutorial cater for use cases that requires a highly customized spreadsheet.

The Spreadsheet form element uses the Handsontable library, specifically version 6.2.2 for Joget Workflow V6.

There are a wealth of plugins and APIs in the library documentation to change the cell appearance, cell selection, dynamic data, dynamic validation, and much more.

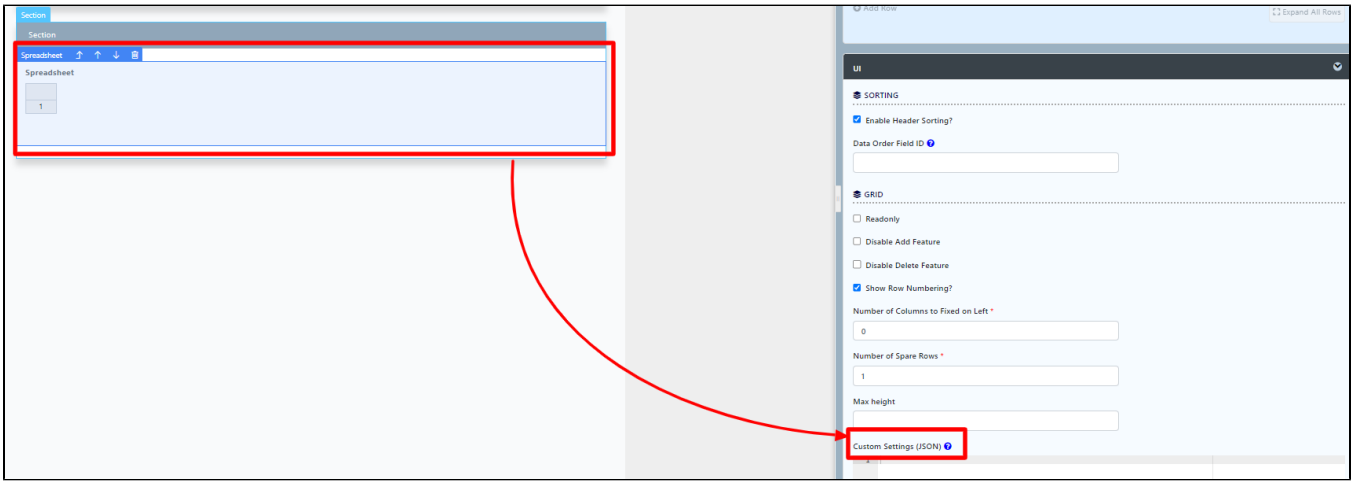

Figure 1 : Spreadsheet Form Element Properties - UI - Custom Settings

In **Figure 1**, we can add in more spreadsheet properties available in the library documentation.

The configuration here is formatted as JSON.

Usually for simple configurations, a single object and value in Custom Settings would suffice. But for more complex ones, this configuration can be used in combination with a Custom HTML form element to complete a functionality.

Do scroll down below for a few examples referenced from the documentation.

-

Example 1 - Limit Cell Selection

https://handsontable.com/docs/6.2.2/Options.html#selectionMode

Use case: Limit the cell(s) that can be selected to only single cell.

Copy & paste this code snippet into Custom Settings.

```
{
    selectionMode: 'single'
}
```

Example 2 - Custom invalid cell style

https://handsontable.com/docs/6.2.2/Options.html#invalidCellClassName

Use case: Change the appearance of a cell if values does not match regex validation (regex configurable in spreadsheet properties).

First, use a Custom HTML form element to write a simple class style. Do use the **!important** notation only if the style is being overridden.

```
<style>
        .invalidCellCustom {
                background:pink !important;
        }
</style>
```

Then, copy & paste this code snippet into Custom Settings.

```
{
        invalidCellClassName: 'invalidCellCustom'
}
```

Example 3 - Get spreadsheet handsontable instance by form element ID

https://handsontable.com/docs/6.2.2/Core.html

Use case: Get the spreadsheet handsontable instance to use core functions.

First, copy & paste this code snippet into Custom Settings.

```
{
    "afterInit" : function() {
        var hot = this;
        $(hot.rootElement).data("hot", hot);
    }
}
```

Then, use a Custom HTML form element to get the 'hot' instance. After that, you are able to perform core functions on your specified spreadsheet element.

```
<script>
        $(function(){
            var hot = FormUtil.getField("_yourSpreadsheetFormElementIdHere_").data("hot");
                //console.log(hot.getSettings());
                //hot.setDataAtRowProp(0, '_yourcellColumnIdHere_', '_myNewValue_');
        });
</script>
```

Example 4 - Add New Row using Javascript

After performing example 3 above, we can use the script below to programmatically add new rows into the speadsheet.

```
<script>
    var col = hot.countRows();
    hot.alter('insert_row', col, 1);
    hot.setDataAtCell(col, 0, '-Name-');
    hot.setDataAtCell(col, 1, '-Surname-');
    hot.setDataAtCell(col, 2, '-Age-')
</script>
```

Reference: https://jsfiddle.net/ck4859zm/