DataGrid2: Auto-Select First Row

Overview

A common request in Mendix development is to automatically select the first row in a DataGrid2 after data changes. While the predecessor of DataGrid2 had this functionality built-in, it's notably absent in the current version.

This guide provides a JavaScript-based solution to automatically select a row in DataGrid2.

The Challenge

The main technical challenge is that the `clickable` class on DataGrid2 rows is not immediately available when the page loads. The class is added asynchronously after the `mx.addOnLoad` function executes, requiring a delayed execution approach.

Solution Architecture

The solution consists of four components:

1. **JavaScript Action** - Core logic to select a row
2. **Nanoflow** - Wrapper to execute the JavaScript Action
3. **Event Handler**- Trigger mechanism (page load)
4. **Widgets On Event Handlers** - Trigger mechanism (selection change, etc.)

JavaScript Implementation

Code Structure

```
// This file was generated by Mendix Studio Pro.
//
// WARNING: Only the following code will be retained when actions are regenerated:
// - the import list
// - the code between BEGIN USER CODE and END USER CODE
// - the code between BEGIN EXTRA CODE and END EXTRA CODE
// Other code you write will be lost the next time you deploy the project.
import "mx-global";
import { Big } from "big.js";

// BEGIN EXTRA CODE
function sleep(ms) {
 return new Promise(resolve => setTimeout(resolve, ms));
}
// END EXTRA CODE

/**
 * DataGrid 2 doesn't have a function to select the first row.
 * This JavaScript action allows you to select a row in a DataGrid 2 when opening a page, and/or when a
selection is changed.
 *
 * Parameters:
```

```
 * - Datagrid2Name: Mendix name of the DataGrid 2 (without the class prefix "mx-name-").
 * - Datagrid2Row: Number of the row to be selected. (1 for the first row.)
 *
 * Note: To my surprise, the "clickable" class names aren't yet present when mx.addOnLoad is executed.
 * Therefore, a small delay of 100 milliseconds has been added and the test if "clickable" is present is
repeadedly tested. When present, the clickElement function is invoked.
 * @param {string} datagrid2Name - Name of the Datagrid2.
(The part "mx-name-"is added in the coding.)
 * @param {string} datagrid2Row - Rownumber to select.
 * @returns {Promise.<void>}
 */
export async function JS_DG2_SelectRow(datagrid2Name, datagrid2Row) {
 // BEGIN USER CODE
 mx.addOnLoad(function () {
  const gridname = ".mx-name-" + datagrid2Name;
  var tries = 0;
  var content = null;
  var clickelement = null;

  // Wait until the gridtable is available
  async function waitForContent() {
   tries = 0;
   do {
    tries += 1;
    if (tries > 50) {
     console.log("JS_DG2_SelectRow: Giving up after 5 seconds waiting for datagrid content.");
     return;
    } // endif
    await sleep(100);
    const gridtable = document.querySelector(gridname);
    if (gridtable) {
     content = gridtable.querySelector(".widget-datagrid-grid-body");
    } // endif
   } // enddo
   while (content == null);
  } // end function waitForContent

  // Wait until the clickable element is available
  async function waitForClickable() {
   tries = 0;
   do {
    tries += 1;
    if (tries > 50) {
     console.log("JS_DG2_SelectRow: Giving up after 5 seconds waiting for clickable element.");
     return;
    } // endif
    await sleep(100);
    const tablerow = content.querySelector(".tr:nth-child(" + datagrid2Row + ")");
    if (tablerow) {
     clickelement = tablerow.querySelector(".clickable");
    } // endif
```

```
   } // enddo
   while (clickelement == null || clickelement == 'undefined');
  } // end function waitForClickable

  // First wait for content, then for clickable, then click it.
  waitForContent().then(() => {
   if (content != null) {
    waitForClickable().then(() => {
     if (clickelement != null) {
      clickelement.click();
     } // endif
    }); // end then
   } // endif
  }); // end then
 }); // end mx.addOnLoad
 // END USER CODE
}
```

How It Works

1. **Polling Mechanism**: Checks every 100ms until the gridtable is found
2. **Polling Mechanism**: Checks every 100ms until the clickable class is found
3. **Row Selection**: Simulate a click on the specified row
4. **CSS Selectors**: Uses `.mx-name-{datagrid2Name}` to locate the grid, `.widget-datagrid-grid-body` to locate the grid body, `.tr:nth-child({datagrid2Row})` to locate the row and `.clackable` to locate the first clickable element in the row.

Configuration in Mendix Studio Pro

JavaScript Action Parameters

Parameter	Type	Description
`datagrid2Name`	String	Name of the DataGrid2 (without "mx-name-" prefix)
`datagrid2Row`	String	Row number to select (typically "1" for first row)

Nanoflow Setup

Create a nanoflow named `NF_DG2_SelectRow` that:

1. Accepts the same parameters as the JavaScript Action
2. Calls the `JS_DG2_SelectRow` JavaScript Action
3. Passes through the parameters

Use Cases

1. Page Load Event

Use a **Page Event** widget (from Marketplace) to trigger selection when a page opens.

Configuration:

- Event: On Load
- Action: Call nanoflow `NF_DG2_SelectRow`
- Parameters:
  - `datagrid2Name`: Enter grid name (e.g., "myDataGrid")
  - `datagrid2Row`: "1" (or desired row number)

2. Master-Detail Grids

When a DataGrid2 listens to another grid (master-detail pattern).

Configuration:

- Widget: Master DataGrid2
- Event: **On Selection Change**
- Action: Call nanoflow `NF_DG2_SelectRow`
- Target: Detail grid name

3. Dropdown/Combobox Filtering

When a DataGrid2 is filtered by a dropdown selection.

Configuration:

- Widget: Dropdown/Combobox
- Event: **On Change**
- Action: Call nanoflow `NF_DG2_SelectRow`
- Target: Filtered grid name

Best Practices

1. **Naming Convention**: Use clear, consistent names for DataGrid2 widgets
2. **Row Number**: While typically "1", you can select any row number for specific scenarios
3. **Performance**: The 100ms polling interval balances responsiveness and performance
4. **Error Handling**: Consider adding null checks if grids might be empty
5. **Testing**: Test with different data volumes to ensure reliable row selection

Troubleshooting

Issue	Cause	Solution
Row not selected	Grid name incorrect	Verify exact DataGrid2 name (case-sensitive)
Wrong row selected	Incorrect row parameter	Check row number parameter value
No response	JavaScript not executing	Verify nanoflow is triggered correctly

Dependencies

- **Mendix Marketplace**: Events widget (for page load triggers)
- **Browser Support**: Modern browsers with Promise support
- **Mendix Version**: Compatible with Mendix versions supporting DataGrid2

Conclusion

This solution effectively restores the auto-select functionality missing from DataGrid2. By using a polling mechanism to wait for DOM readiness, it reliably selects rows across different scenarios and timing conditions.

The modular approach with separate JavaScript Action and nanoflow makes it reusable across your application and easy to maintain.