Grouping rows in Gridview

Integrating collapsible groups with triangles in a Yii2 `GridView` requires a creative approach because `GridView` is primarily designed for flat data presentation. However, you can achieve this by manipulating the `GridView` and incorporating custom HTML and JavaScript for the collapsible functionality. Here’s a conceptual outline of how you might approach this:

Step 1: Prepare Your Data

Organize your data in a way that it can be grouped logically. Your data query should include the field(s) you intend to group by.

Step 2: Customize Gridview Rows

You'll use the `beforeRow` and `afterRow` properties of the `GridView` to inject custom HTML for group headers and footers. The group header will include the triangle and group title, and the group footer will serve as a marker for the end of a group.

PHP

```
echo GridView::widget([
    'dataProvider' => $dataProvider,
    'filterModel' => $searchModel,
    'beforeRow' => function ($model, $key, $index, $grid) {
        static $lastGroup = null;
        $currentGroup = $model->groupAttribute; // Replace 'groupAttribute' with your actual attribute used for
grouping

        if ($lastGroup !== $currentGroup) {
            $lastGroup = $currentGroup;
            return "<tr class='group-header'><td colspan='5'>
                        <div class='group-title'><span class='triangle'>&#9662;</span> {$currentGroup}
                        </div>
                    </td></tr>";
        }
        return null;
    },
    // Define your columns, and other GridView configurations here...
]);
```

Step 3: CSS for Styling

Add CSS for the group headers and the triangle icon. You can style the triangle to indicate collapsibility and use CSS transformations to change its orientation based on the group's collapsed state.

CSS

```
.group-header .triangle {
    cursor: pointer;
    display: inline-block;
    transition: transform 0.3s ease-in-out;
}

.group-header.collapsed .triangle {
    transform: rotate(-90deg); /* Pointing right when collapsed */
}

.group-content {
    display: none; /* Initially hide the content */
}
```

Step 4: JavaScript for Collapse/Expand Functionality

Use JavaScript to handle the click event on the group header, toggling the visibility of the group content and the orientation of the triangle icon.

JavaScript

```
$(document).on('click', '.group-header', function() {
    var header = $(this);
    header.toggleClass('collapsed');
    header.nextUntil('.group-header').toggle(); // This will show/hide the rows until the next group header
});
```

Step 5: Integration with Gridview

- **Group Content**: The actual rows of the `GridView` will serve as the group content. They will be shown/hidden based on the header click.
- **Group Headers and Footers**: These are custom rows injected into the `GridView` using `beforeRow`. There's no built-in `afterRow`, but you can simulate this by checking if the next row belongs to a different group or if it's the last row of the current group.

Caveats and Considerations

- **Performance**: With many rows and groups, the performance might be impacted, especially client-side due to the large DOM manipulation.
- **Complexity**: This solution adds a layer of complexity to your `GridView` rendering logic, so it's crucial to maintain clean and well-documented code.
- **Styling and Behavior**: You might need to refine the CSS and JavaScript to fit the visual and functional requirements of your application.

This approach is a bit of a hack around the default `GridView` behavior, so it requires careful implementation and testing to ensure it works smoothly across different datasets and user interactions.

Revision #4
Created 2 February 2024 19:14:33 by Max
Updated 4 February 2024 19:09:38 by Max