Grid View - Max MVC

Below is sample documentation that explains how to configure each column in your grid. You can include this in your project’s documentation (for example, as a README or developer guide).

Grid Column Data Structure

Each column in the grid is defined as an associative array. The following keys are available:

Required Keys

- **`name`**
  Type: *string*
  Description:
  The header text shown at the top of the column.
  Example:

```
'name' => 'Order ID'
```

- **`data`**
  Type: *string*
  Description:
  Specifies how to obtain the value for each cell in this column.
  - If the value is a plain key (e.g., `"order_id"`), the cell displays the corresponding value from each record in your data array.
  - If the value contains tokens enclosed in curly braces (e.g. `"{price} * {quantity}"`), it is treated as a formula. Each token is replaced by the value from the corresponding key in the data record, and the resulting expression is evaluated.
    Examples:

```
// Plain data field (must exist in each $data record)
'data' => 'order_id'
```

```
// Calculated field using tokens
'data' => '{price} * {quantity}'
```

Optional Keys

- **`width`**
  Type: *string*
  Description:
  The width of the column as a CSS value (e.g., `"60px"` or `"10%"`).
  Example:

```
'width' => '80px'
```

- **`align`**
  Type: *string*
  Description:
  Sets the text alignment for the column. Valid values are:
  - `"left"` (default)
  - `"right"`
  - `"center"` **Example:**

```
'align' => 'right'
```

- **`formatter`**
  Type: *string*
  Description:
  A PHP expression used to format the cell's value. The expression is evaluated using `eval()`. You can use the variable `$item` (or `$value` for computed columns) to reference the current cell data.
  Example:

```
'formatter' => 'number_format($item["price"], 2, ".", ",")'
```

  When using formulas for the `data` key, the computed value can be formatted by referencing `$value`:

```
'formatter' => 'number_format($value, 2, ".", ",")'
```

- **`aggregate`**
  Type: *string*
  Description:
  Specifies how to compute an aggregate value for this column (displayed in the footer). Accepted values:
  - `"sum"` – calculates the total.
  - `"average"` – calculates the average.
  - A formula string (e.g., `"({YTD_PL} / {VALUE_EUR}) * 100"`) that uses tokens to reference computed aggregate values from other columns. **Example:**

```
// Standard aggregate (sum)
'aggregate' => 'sum',

// Aggregate based on a formula
'aggregate' => '({YTD_PL} / {VALUE_EUR}) * 100'
```

- **`aggregateToken`**
  Type: *string*
  Description:
  Used when the computed aggregate value of this column needs to be referenced in another aggregate formula. When provided, the calculated value is stored in a global object (or passed to JavaScript) under this token name.
  Example:

```
'aggregateToken' => 'YTD_PL'
```

- **`sortable`**
  Type: *boolean* (or *integer* with values 0/1)
  Description:
  Indicates whether the column is sortable by clicking on the header.
  Example:

```
'sortable' => true
```

- **`filter`**
  Type: *string*
  Description:
  Specifies the type of filtering available for the column. Valid values are:
  - `"none"` – no filter input.
  - `"select"` – a dropdown list of unique values.

- `"text"` – a text input for filtering. **Example:**

```
'filter' => 'text'
```

- **`hide`**
  Type: *boolean*
  Description:
  Determines whether the column is visible in the rendered grid. If set to `true`, the column is hidden from the user view via CSS (`display: none`), but it remains in the DOM so that its data is still available for calculations (such as aggregates or formulas).
  Example:

```
'hide' => true
```

Example of a Complete Column Configuration

Below is a sample array of columns that demonstrates how to configure different aspects:

```
$columns = [
  [
    'name'       => 'Order ID',
    'data'       => 'order_id',
    'width'      => '80px',
    'align'      => 'left',
    'sortable'   => true,
    'filter'     => 'text',
  ],
  [
    'name'       => 'Customer',
    'data'       => 'customer_name',
    'width'      => '150px',
    'align'      => 'left',
    'sortable'   => true,
    'filter'     => 'select',
  ],
  [
```

```
    'name'       => 'Price',
    'data'       => 'price',
    'width'      => '80px',
    'align'      => 'right',
    'formatter'  => 'number_format($item["price"], 2, ".", ",")',
    'aggregate'  => 'sum',
    'sortable'   => true,
    'filter'     => 'none',
  ],
  [
    'name'       => 'Quantity',
    'data'       => 'quantity',
    'width'      => '60px',
    'align'      => 'right',
    'aggregate'  => 'sum',
    'sortable'   => true,
    'filter'     => 'none',
  ],
  [
    'name'       => 'Total',
    'data'       => '{price} * {quantity}', // Calculated column using tokens
    'width'      => '100px',
    'align'      => 'right',
    'formatter'  => 'number_format($value, 2, ".", ",")', // $value refers to the computed total
    'aggregate'  => 'sum',
    'sortable'   => false,
    'filter'     => 'none',
  ],
  [
    'name'       => 'Secret Code',
    'data'       => 'secret_code',
    'hide'       => true, // Hidden from the view but used for calculations or references
    'sortable'   => false,
    'filter'     => 'none',
  ],
];
```

Notes

- **Formulas in `data` or `aggregate`:**
  When using a formula, ensure that tokens (e.g., `{price}` or `{quantity}`) exactly match the keys in your data records. If a token is missing, you may get unexpected results or errors.
- **DOM and Calculations:**
  Even if a column is hidden (`hide` is set to `true`), it is still rendered in the DOM (using a CSS class such as `hidden-col` with `display: none`). This is important so that JavaScript functions that perform calculations (such as aggregate totals) can still access the values.
- **Formatter Security:**
  Since `formatter` is evaluated using `eval()`, make sure that any code provided is trusted and controlled. Do not use untrusted input in these expressions.

This documentation provides an overview of how to set up and customize the grid’s column definitions. Adjust the examples to match your application’s needs, and feel free to extend the configuration with additional keys or logic as required.

Revision #2
Created 16 February 2025 18:16:10 by Max
Updated 16 February 2025 18:21:40 by Max