**Reusable PrimeNG Data Table Component**

**Overview**

This document provides a comprehensive explanation of a reusable PrimeNG data table component built using Angular. The table is designed to handle dynamic datasets and column configurations, enabling the display of data in a flexible and aesthetically styled tabular format. The reusable component (GenericPtableComponent) supports functionalities such as pagination, sorting, search filtering, and responsive layout. Two parent components (Parent1Component and Parent2Component) demonstrate how the table can be reused with different data models and configurations.

**1. GenericPtableComponent**

**Purpose**

**GenericPtableComponent** serves as a reusable component to display data in a PrimeNG p-table. It takes dynamic data and column configurations via @Input bindings.

**Inputs**

* @Input() tableData: any[]: Array of data to be displayed in the table.
* @Input() columns: any[]: Column configuration including field and header values.

**Template Highlights**

<div class="custom-table-container">

<input

type="text"

pInputText

placeholder="Search..."

[(ngModel)]="searchText"

class="search-input"

/>

<p-table

[value]="filteredData"

[paginator]="true"

[rows]="10"

[rowsPerPageOptions]="[10, 20, 50]"

[responsiveLayout]="'scroll'"

[styleClass]="'p-datatable-striped'"

>

<ng-template pTemplate="header">

<tr>

<th \*ngFor="let col of columns">{{ col.header }}</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-rowData>

<tr>

<td \*ngFor="let col of columns">{{ rowData[col.field] }}</td>

</tr>

</ng-template>

</p-table>

</div>

**Functionality**

* **Search Filtering**: Filters the dataset using searchText across all fields.
* **Dynamic Columns**: Adapts to the configuration passed by the parent.
* **Pagination**: Built-in via PrimeNG.
* **Responsive Layout**: Supports scroll for smaller screens.

**Styling**

Defined in generic-ptable.component.scss, includes:

* Themed headers and striped rows.
* Hover effects for interactivity.
* Custom pagination styles.

**2. Parent2Component**

**Purpose**

This component demonstrates using the **GenericPtableComponent** with a detailed dataset (TableData interface) of electronic products.

**Key Features**

* Generates a large dataset (100 rows) using mock data.
* Passes columns and data dynamically to the generic table.

**TypeScript Logic**

ngOnInit(): void {

this.tableData = this.generateElectronicProductsData(100);

this.tableColumns = [

{ field: 'id', header: 'ID' },

{ field: 'brand', header: 'Brand' },

... // other columns

];

}

**Data Generation**

**generateElectronicProductsData()** creates a list of products with randomized attributes (brand, model, price, specs, etc.) for demonstration.

**3. Model: TableData**

Defines the structure of data expected for Parent2Component:

export interface TableData {

id: number;

brand: string;

model: string;

category: string;

price: string;

stock: number;

rating: string;

warranty: string;

releaseDate: string;

color: string;

weight: string;

dimensions: string;

batteryLife: string;

connectivity: string;

operatingSystem: string;

processor: string;

gpu: string;

description: string;

}

**4. Styling Details (generic-ptable.component.scss)**

**Table Container**

* Full width with horizontal scrolling.
* White background and rounded corners.

**Header & Rows**

* Bold colored headers.
* Zebra-striped rows.
* Row hover effect for improved UX.

**Pagination**

* Styled page buttons.
* Highlighted active page.

**Responsive Design**

* Ensures usability across devices.
* Min-width and scroll enable usability for many columns.

**5. Reusability and Extensibility**

**Easily Extendable Features:**

* Column filters
* Row selection
* Export to CSV/Excel
* Inline editing
* Sorting by multiple fields

**Reusability Guidelines**

* Use interface models to type-check incoming data.
* Pass minimal configuration from parent.
* Keep layout styling within the generic component.

**6. Usage Examples**

**Example 1: Parent1Component (Simplified Columns)**

<app-generic-ptable [columns]="simpleColumns" [tableData]="simpleData"></app-generic-ptable>

**Example 2: Parent2Component (Full Product Table)**

<app-generic-ptable [columns]="tableColumns" [tableData]="tableData"></app-generic-ptable>

**Conclusion**

This documentation outlines a clean, reusable Angular component architecture using PrimeNG’s p-table. The component's modularity enables consistent UI design and reduces repetitive code. By centralizing logic and visual behavior, it helps maintain and scale Angular applications with efficiency.

For further improvements, consider encapsulating search logic in a service, adding advanced filters, and using reactive forms for more control.