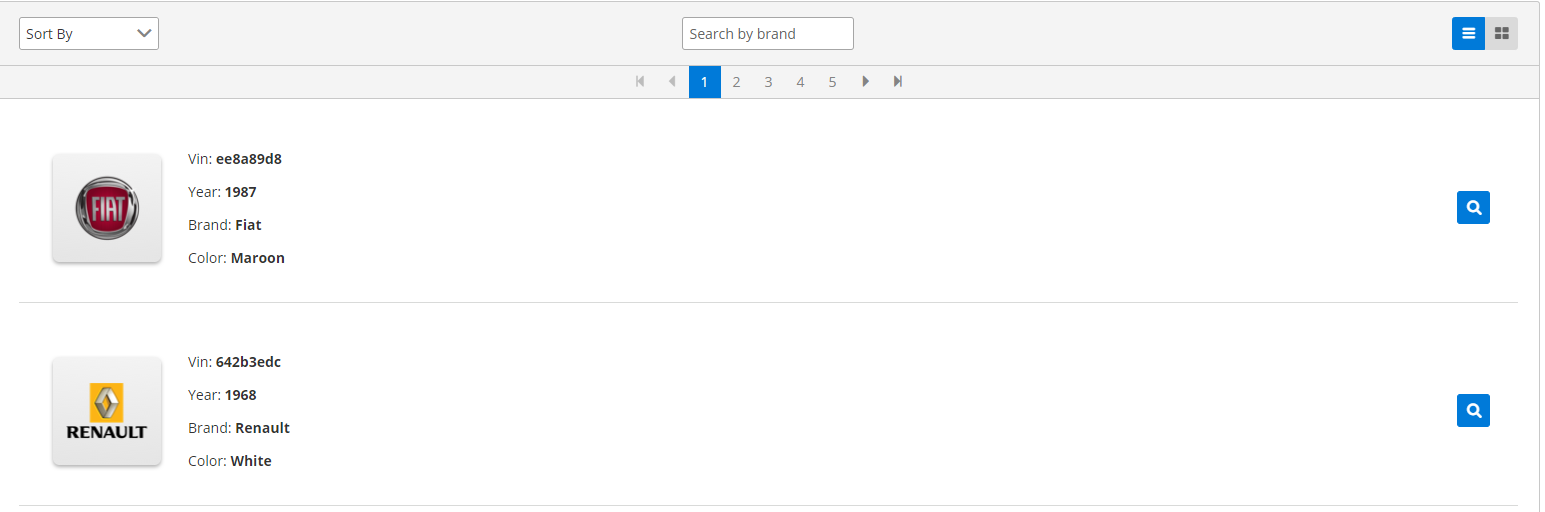
|  |  |
| --- | --- |
| html | <p-dataView #dv [value]="cars" [paginator]="true" [rows]="20" paginatorPosition="both" filterBy="brand"  [sortField]="sortField" [sortOrder]="sortOrder">  <p-header>  <div class="ui-helper-clearfix">  <div class="ui-g">  <div class="ui-g-12 ui-md-4">  <p-dropdown [options]="sortOptions" [(ngModel)]="sortKey" placeholder="Sort By" (onChange)="onSortChange($event)" [style]="{'min-width':'140px'}"></p-dropdown>  </div>  <div class="ui-g-6 ui-md-4 filter-container">  <div style="position:relative">  <input type="search" pInputText placeholder="Search by brand" (input)="dv.filter($event.target.value)">  </div>  </div>  <div class="ui-g-6 ui-md-4" style="text-align:right">  <p-dataViewLayoutOptions></p-dataViewLayoutOptions>  </div>  </div>  </div>  </p-header>  <ng-template let-car pTemplate="listItem">  <div class="ui-g-12">  <div class="car-details">  <div>  <img src="assets/showcase/images/demo/car/{{car.brand}}.png">  <div class="ui-g">  <div class="ui-g-12">Vin: <b>{{car.vin}}</b></div>  <div class="ui-g-12">Year: <b>{{car.year}}</b></div>  <div class="ui-g-12">Brand: <b>{{car.brand}}</b></div>  <div class="ui-g-12">Color: <b>{{car.color}}</b></div>  </div>  </div>  <button pButton type="button" icon="pi pi-search" (click)="selectCar($event, car)"></button>  </div>  </div>  </ng-template>  <ng-template let-car pTemplate="gridItem">  <div style="padding:.5em" class="ui-g-12 ui-md-3">  <p-panel [header]="car.vin" [style]="{'text-align':'center'}">  <img src="assets/showcase/images/demo/car/{{car.brand}}.png" width="60">  <div class="car-detail">{{car.year}} - {{car.color}}</div>  <hr class="ui-widget-content" style="border-top:0">  <button pButton type="button" icon="pi pi-search" (click)="selectCar($event, car)" style="margin-top:0"></button>  </p-panel>  </div>  </ng-template>  </p-dataView> |
| component | export class DataViewDemo implements OnInit {  cars: Car[];  selectedCar: Car;  displayDialog: boolean;  sortOptions: SelectItem[];  sortKey: string;  sortField: string;  sortOrder: number;  constructor(private carService: CarService) { }  ngOnInit() {  this.carService.getCarsLarge().then(cars => this.cars = cars);  this.sortOptions = [  {label: 'Newest First', value: '!year'},  {label: 'Oldest First', value: 'year'},  {label: 'Brand', value: 'brand'}  ];  }  selectCar(event: Event, car: Car) {  this.selectedCar = car;  this.displayDialog = true;  event.preventDefault();  }  onSortChange(event) {  let value = event.value;  if (value.indexOf('!') === 0) {  this.sortOrder = -1;  this.sortField = value.substring(1, value.length);  }  else {  this.sortOrder = 1;  this.sortField = value;  }  }  onDialogHide() {  this.selectedCar = null;  }  } |

Have a image



Parameter to remember

|  |  |  |  |
| --- | --- | --- | --- |
| value | array | null | An array of objects to display. |
| layout | string | list | Layout of the items, valid values are "list" and "grid". Nếu dung Grid  thì nhớ thêm css phù hợp |
| paginator | boolean | false | When specified as true, enables the pagination. |
| rows | number | null | Number of rows to display per page. |
| totalRecords | number | null | Number of total records, defaults to length of value when not defined. |
| pageLinks | number | 5 | Number of page links to display in paginator. |
| rowsPerPageOptions | array | null | Array of integer/object values to display inside rows per page dropdown  of paginator |
| paginatorPosition | string | bottom | Position of the paginator, options are "top","bottom" or "both". |
| alwaysShowPaginator | boolean | true | Whether to show it even there is only one page. |
| paginatorDropdownAppendTo | any | null | Target element to attach the paginator dropdown overlay,  valid values are "body" or a local ng-template variable of another element. |
| paginatorDropdownScrollHeight | string | 200px | Paginator dropdown height of the viewport in pixels, a scrollbar is defined  if height of list exceeds this value. |
| currentPageReportTemplate | string | {currentPage} of {totalPages} | Text to display the current page information. |
| showCurrentPageReport | boolean | false | Whether to display current page report. |
| lazy | boolean | false | Defines if data is loaded and interacted with in lazy manner. |
| emptyMessage | string | No records found. | Text to display when there is no data. |
| style | string | null | Inline style of the component. |
| styleClass | string | null | Style class of the component. |
| trackBy | Function | null | Function to optimize the dom operations by delegating to ngForTrackBy, default algoritm checks for object identity. |
| filterBy | string | null | Comma separated list of fields in the object graph to search against. |
| loading | boolean | false | Displays a loader to indicate data load is in progress. |
| loadingIcon | string | fa-circle-o-notch | The icon to show while indicating data load is in progress. |
| first | number | 0 | Index of the first row to be displayed. |

Events

| **Name** | **Parameters** | **Description** |
| --- | --- | --- |
| onLazyLoad | event.first = First row offset event.rows = Number of rows per page | Callback to invoke when paging, sorting or filtering happens in lazy mode. |
| onPage | event.first: Index of first record in page event.rows: Number of rows on the page | Callback to invoke when pagination occurs. |
| onSort | event.field: Name of the sort field. event.order: Order of the sort. | Callback to invoke when sorting occurs. |

Styling

Following is the list of structural style classes, for theming classes visit [theming page](https://www.primefaces.org/primeng/#/theming).

| **Name** | **Element** |
| --- | --- |
| ui-dataview | Container element. |
| ui-dataview-header | Header section. |
| ui-dataview-footer | Footer section. |
| ui-dataview-content | Container of items. |
| ui-table-loading | Loader mask. |
| ui-dataview-loading | Loader background. |
| ui-dataview-loading-content | Loader content. |
| ui-dataview-emptymessage | Empty message element. |

Example1: Có onLadyLoad

<p-dataView [value]="cars" [paginator]="true" [rows]="10"

[lazy]="true" (onLazyLoad)="loadData($event)" [totalRecords]="totalRecords">

<ng-template let-car pTemplate="listItem">

<div>

{{car.id}}

</div>

</ng-template>

<ng-template let-car pTemplate="gridItem">

<div class="ui-g-12 ui-md-3">

{{car.year}}

</div>

</ng-template>

</p-dataView>

oadData(event) {

//event.first = First row offset

//event.rows = Number of rows per page

}

Examle2:

<p-dataView [value]="cars" [sortField]="sortField" [sortOrder]="sortOrder">

<p-header>

<p-dropdown [options]="sortOptions" [(ngModel)]="sortKey" placeholder="Sort By"

(onChange)="onSortChange($event)" [style]="{'min-width':'15em'}"></p-dropdown>

</p-header>

<ng-template let-car pTemplate="listItem">

<div>

{{car.id}}

</div>

</ng-template>

<ng-template let-car pTemplate="gridItem">

<div class="ui-g-12 ui-md-3">

{{car.year}}

</div>

</ng-template>

</p-dataView>

export class DataViewSortDemo implements OnInit {

cars: Car[];

sortOptions: SelectItem[];

sortKey: string;

sortField: string;

sortOrder: number;

constructor(private carService: CarService) { }

ngOnInit() {

this.carService.getCarsLarge().then(cars => this.cars = cars);

this.sortOptions = [

{label: 'Newest First', value: '!year'},

{label: 'Oldest First', value: 'year'},

{label: 'Brand', value: 'brand'}

];

}

onSortChange(event) {

let value = event.value;

if (value.indexOf('!') === 0) {

this.sortOrder = -1;

this.sortField = value.substring(1, value.length);

}

else {

this.sortOrder = 1;

this.sortField = value;

}

}

}

Example3

<p-dataView #dv [value]="cars" filterBy="brand">

<p-header>

<input type="search" pInputText placeholder="Search by brand" (input)="dv.filter($event.target.value, "contains")">

</p-header>

<ng-template let-car pTemplate="listItem">

{{car.id}}

</ng-template>

<ng-template let-car pTemplate="gridItem">

{{car.year}}

</ng-template>

</p-dataView>

Example4:

<p-dataview [value]="cars" [loading]="loading">

//content

</p-dataview>