Pages Display Documentation  
1. **PagesDisplayWebPart.ts**  
  
**Major Parts of the Code**

1. **selectedView Property**:
   * **Purpose**: Stores the ID of the selected view from the "Site Pages" library.
   * **Usage**: This property determines which view is applied to filter the pages displayed by the web part. It's configured in the property pane using a dropdown populated with available views.
2. **feedbackLink Property**:
   * **Purpose**: Holds the URL for a feedback page or form.
   * **Usage**: This link is passed down to the PagesDisplay React component, which can be used to direct users to a feedback form related to the displayed pages.
3. **viewOptions Array**:
   * **Purpose**: Stores the available views from the "Site Pages" list.
   * **Usage**: Populated by the getViews method, this array is used to create the dropdown options in the property pane, allowing users to select a view to filter pages.
4. **render Method**:
   * **Purpose**: Renders the React component (PagesDisplay) and passes necessary properties like selectedViewId and feedbackPageUrl.
   * **Usage**: Ensures that the selected view and feedback link are passed to the child component, which displays the filtered pages and potentially includes a feedback mechanism.
5. **getViews Method** :
   * **Purpose**: Fetches available views from the "Site Pages" library using the SharePoint REST API.
   * **Usage**: Populates the viewOptions array with the retrieved views, allowing the property pane dropdown to display these options for user selection.
6. **getPropertyPaneConfiguration Method**:
   * **Purpose**: Configures the property pane, where users can select a view and input a feedback link.
   * **Usage**: Defines how the user interacts with the web part settings, specifically choosing which view to use for filtering pages and where the feedback should be directed. The dropdown and text fields here are tied to selectedView and feedbackLink, respectively.

2. PagesList.tsx

## **Functions**

### **resetFilters**

**Description:**Resets all active filters by clearing the filter details and taxonomy filters, as well as clearing the search text. It then fetches the pages using default parameters.

**Parameters:**None

**Returns:**void

### **fetchPages**

**Description:**Fetches the pages from the SharePoint site based on the provided parameters. It applies non-taxonomy filters server-side and taxonomy filters client-side.

**Parameters:**

* page (number) [default: 1]: The page number to fetch.
* pageSizeAmount (number) [default: pageSize]: The number of items per page.
* sortBy (string) [default: "Created"]: The column to sort by.
* isSortedDescending (boolean) [default: isDecending]: Determines if the sorting is in descending order.
* searchText (string) [default: ""]: The text to filter by.
* category (string) [default: catagory]: The category to filter by.
* filterDetails (FilterDetail[]): The filter details to apply.
* taxonomyFilters (FilterDetail[]) [default: []]: The taxonomy filters to apply locally.
* columns (IColumnInfo[]) [default: columnInfos]: The columns information.

**Returns:**Promise<void>

### **getPages**

**Description:**Fetches the initial pages based on the given path and updates the state with these pages.

**Parameters:**

* path (string | null): The path to the SitePages library.
* columns (IColumnInfo[]): The columns information.

**Returns:**Promise<void>

### **applyFilters**

**Description:**Applies the given filter details to filter the pages. Updates the current filter details state and fetches the pages based on the updated filters.

**Parameters:**

* filterDetail (FilterDetail): The filter detail object containing the filter criteria.

**Returns:**void

### **sortPages**

**Description:**Sorts the pages based on the specified column and fetches the pages list with the new sort criteria.

**Parameters:**

* column (IColumn): The column to sort by.

**Returns:**void

### **handlePageChange**

**Description:**Handles pagination by updating the current page number and calculating the paginated data based on the new page number and page size.

**Parameters:**

* page (number): The page number to navigate to.
* pageSizeChanged (number) [default: pageSize]: The number of items per page.

**Returns:**void

### **handleSearch**

**Description:**Handles the search functionality by fetching pages based on the current search text and other filters.

**Parameters:**None

**Returns:**void

### **goToFirstPage**

**Description:**Navigates to the first page of paginated data.

**Parameters:**None

**Returns:**void

### **goToLastPage**

**Description:**Navigates to the last page of paginated data.

**Parameters:**None

**Returns:**void

### **goToPreviousPage**

**Description:**Navigates to the previous page of paginated data. If the current page is the first page, no action is taken.

**Parameters:**None

**Returns:**void

### **goToNextPage**

**Description:**Navigates to the next page of paginated data. If the current page is the last page, no action is taken.

**Parameters:**None

**Returns:**void

### **handleInputChange**

**Description:**Handles the input change event for page number navigation. Parses the input value to an integer and calls handlePageChange with the parsed value.

**Parameters:**

* e (any): The event object.

**Returns:**void

### **handlePageSizeChange**

**Description:**Handles the change event of the page size dropdown by updating the page size state and calling handlePageChange with the new page size.

**Parameters:**

* e (any): The event object.

**Returns:**void

### **dismissPanel**

**Description:**Dismisses the filter panel by setting the showFilter state to false.

**Parameters:**None

**Returns:**void

### **getColumns**

**Description:**Fetches the columns for the specified view and updates the state with the retrieved columns.

**Parameters:**

* selectedViewId (string): The ID of the selected view.

**Returns:**Promise<IColumnInfo[]>

## **React Hooks**

#### **React.useEffect (1st Instance)**

* **Description:** Sets up a polling mechanism to monitor the URL of the subscribeIframeRef. If the URL of the iframe changes to something other than the expected subscribeLink or contains "blank", it hides the alert dialog and triggers a function to fetch pages based on the selected category and column information.
* **Dependencies:**
  + [subscribeLink, setHideAlertMeDialog, catagory, columnInfos]
  + This hook re-runs if any of these dependencies change.

#### **React.useEffect (2nd Instance)**

* **Description:** Sets up a polling mechanism to monitor the URL of the feedBackIframeRef. If the URL of the iframe changes to something other than the expected props.feedbackPageUrl or contains "blank", it hides the feedback dialog.
* **Dependencies:**
  + [props.feedbackPageUrl, setHideFeedBackDialog]
  + This hook re-runs if any of these dependencies change.

### **React.useEffect (3rd Instance)**

**Description:**Sets up an event listener for category selection and fetches the list ID and initial columns on component mount.

**Dependencies:  
[columnInfos]**

### **React.useEffect (4th Instance)**

**Description:**Fetches columns and pages when the selected view ID changes.

**Dependencies:**[selectedViewId]

## **State Variables**

* filterDetails (FilterDetail[]): Stores the non-taxonomy filter details.
* taxonomyFilters (FilterDetail[]): Stores the taxonomy filter details.
* searchText (string): Stores the current search text.
* pageSize (number): Stores the number of items per page.
* sortBy (string): Stores the current sorting column.
* isDecending (boolean): Stores the sorting order (ascending/descending).
* totalItems (number): Stores the total number of filtered items.
* totalPages (number): Stores the total number of pages based on the current page size.
* paginatedPages (any[]): Stores the data for the current page.
* pages (any[]): Stores the full filtered result.
* initialPages (any[]): Stores the initial pages fetched from the API.
* currentPageNumber (number): Stores the current page number.
* showFilter (boolean): Controls the visibility of the filter panel.
* catagory (string): Stores the currently selected category.
* viewId (string): Stores the ID of the currently selected view.
* selectedViewId (string): Stores the ID of the selected view.
* listId (string): Stores the ID of the "Site Pages" list.
* columnInfos (IColumnInfo[]): Stores the information about the columns.

## **External Dependencies**

* pagesService: An external service that provides methods for interacting with the SharePoint site, such as fetching pages and columns.
* context: Provides the current SharePoint context, including the web's server-relative URL.

## **Events**

* catagorySelected: An event that triggers when a category is selected, used to fetch pages for the selected category.

### **Render Structure Documentation**

This JSX code represents a UI component that renders a list of articles with filtering, sorting, pagination, and various actions. The component includes a filter panel, a search input, and pagination controls.

### **1. Filter Panel**

* **Conditionally Rendered:**
  + The FilterPanelComponent is rendered if showFilter is true.
* **Props:**
  + isOpen: Controls the visibility of the filter panel.
  + headerText: The title of the filter panel ("Filter Articles").
  + applyFilters: Function to apply the selected filters.
  + dismissPanel: Function to close the filter panel.
  + selectedItems: The currently selected filter items, determined by the filterColumn.
  + columnName, columnType: The column being filtered and its type.
  + pagesService: Service to manage page-related operations.
  + data: Initial data for the filter panel.

### **2. Top Section**

* **Category Display:**
  + Displays the selected category if available.
* **Search Input:**
  + An input field for searching articles. The search triggers when the user presses "Enter".

### **3. Header Section**

* **Conditional Rendering:**
  + Displays actions like "Alert Me", "Manage My Alerts", and "Add Feedback" if selectionDetails contains selected items.
* **Clear Filters Button:**
  + Displays the "Clear" button if there are active filters (filterDetails or taxonomyFilters).
* **Results Count:**
  + Shows the total number of items (totalItems). If no items are found, it shows "No articles to display".

### **4. Main Content**

* **Loading State:**
  + Displays a spinner while loading data (isLoading).
* **List Display:**
  + Renders a ReusableDetailList component to display paginated items with sorting and selection functionalities.
* **Selection Handlers:**
  + setShowFilter: Toggles the filter panel and sets the filterColumn and filterColumnType.
  + updateSelection: Updates the selection details based on user interaction.

### **5. Pagination Controls**

* **Items Per Page:**
  + A dropdown to select the number of items displayed per page (pageSize).
* **Page Navigation:**
  + Controls for navigating between pages, including "First", "Previous", "Next", and "Last".

### **6. Dialogs**

* **Feedback Dialog:**
  + Displays a dialog for submitting feedback using the ListForm component.
* **Alert Me Dialog:**
  + Displays an iframe for subscribing to alerts.
* **Manage Alerts Dialog:**
  + Displays an iframe for managing existing alerts.
* **Dialog Footers:**
  + Each dialog has a "Close" button to dismiss it, with additional actions for the "Manage Alerts" dialog to refresh the page data.

**3. Panel Component**   
  
**FilterPanelComponent**

The FilterPanelComponent is a React component that provides a filtering interface within a panel. It allows users to search and select filter options from a list of distinct values. This component is designed to work with data retrieved from a PagesService and supports filtering based on various column types.

#### **Props**

* **isOpen: boolean**
  + Controls whether the filter panel is visible or not.
* **dismissPanel: () => void**
  + Function to close the filter panel.
* **applyFilters: (filterDetail: FilterDetail) => void**
  + Function to apply the selected filters. It receives a FilterDetail object as an argument.
* **headerText: string**
  + The title text displayed at the top of the panel.
* **selectedItems: FilterDetail**
  + The currently selected filter items, passed to the panel as an initial state.
* **pagesService: PagesService**
  + The service used to fetch distinct values for the filterable column.
* **columnName: string**
  + The name of the column being filtered.
* **columnType: string**
  + The type of the column (e.g., text, number, taxonomy).
* **data: any[]**
  + The dataset used to determine the distinct values for filtering.

#### **State Variables**

* **checkedItems: FilterDetail**
  + Tracks the currently selected filter values. It is initialized with the selectedItems prop and updated based on user interactions.
* **searchText: string**
  + Stores the current text input value for filtering the options list.
* **filteredOptions: FilterOptions[]**
  + Contains the filtered list of options based on the user's search input.
* **options: FilterOptions[]**
  + Contains the complete list of filter options fetched from the PagesService.

#### **Functions**

* **apply()**
  + Applies the current filters by calling the applyFilters function with the checkedItems state.
* **resetFilters()**
  + Resets the filter state by clearing the checkedItems and calling applyFilters with an empty array of values.
* **handleSearch()**
  + Filters the options list based on the searchText input, updating the filteredOptions state.
* **constructCategoryFilters(categories: (number | string | ConstructedFilter)[]): FilterOptions[]**
  + Constructs filter options from the provided categories, updating the options and filteredOptions state with FilterOptions objects.
* **onRenderFooterContent()**
  + Renders the footer of the panel, including the "Apply" and "Clear" buttons.
* **isISODateString(value: string): boolean**
  + Utility function that checks if a given string is in ISO date format. This is used to format date strings appropriately.

#### **Effect Hook**

* **React.useEffect(() => {...}, [columnName])**
  + Runs when the columnName prop changes. This effect fetches distinct values for the column using the pagesService and constructs the filter options using the constructCategoryFilters function.

#### **Component Structure**

* **Panel**
  + The main container for the filter options. It includes a search input, a list of checkboxes for filter options, and footer buttons for applying or clearing the filters.
* **Input**
  + A text input field used for filtering the options list.
* **Stack**
  + A Fluent UI component used to layout the list of checkboxes vertically with a gap between each item.
* **Checkbox**
  + Represents each filter option. The checkbox is checked if the value is currently selected in checkedItems.
* **Footer Buttons**
  + **PrimaryButton**: Triggers the apply function to apply the selected filters.
  + **DefaultButton**: Triggers the resetFilters function to clear all selected filters.

4. Pages Service  
**a. PagesService Class Overview**

This service is designed to handle data retrieval and operations on SharePoint lists, particularly for "Site Pages". It uses the PnP JS library for accessing SharePoint's REST API.

### **b. Methods in PagesService**

#### **Constructor**

* Initializes the SPFI instance using the provided WebPartContext.
* this.\_sp = spfi().using(SPFx(this.context));

#### **getDistinctValues**

* **Purpose**: Fetch distinct values for a given column from a list of items.
* **Parameters**:
  + columnName: The name of the column to retrieve distinct values from.
  + columnType: The type of the column (e.g., TaxonomyFieldTypeMulti, DateTime, User).
  + values: The list of items to process.
* **Returns**: A promise that resolves to an array of distinct values.
* **Implementation**:
  + Iterates over the provided items and extracts distinct values based on the column type.
  + Handles different column types like Taxonomy, DateTime, User, Number, Choice, URL, and Computed.

#### **getFilteredPages**

* **Purpose**: Retrieve a page of filtered Site Pages items based on various filters and search criteria.
* **Parameters**:
  + pageNumber: The page number to retrieve (1-indexed).
  + pageSize: The number of items per page (default is 10).
  + orderBy: The column to sort by (default is "Created").
  + isAscending: Boolean to determine the sort order.
  + folderPath: The folder path to search within (default is root).
  + searchText: Text to search for in specific columns.
  + filters: An array of FilterDetail objects to apply to the query.
  + columnInfos: An array of IColumnInfo objects describing the columns.
* **Returns**: A promise that resolves to an array of filtered items.
* **Implementation**:
  + Constructs a dynamic filter query based on the provided filters and search text.
  + Uses PnP JS to retrieve filtered list items.
  + Enhances the results by checking if the current user is subscribed to alerts for specific pages.
  + Applies column information, including user and taxonomy fields, and formats the results.

#### **getColumns**

* **Purpose**: Retrieve the columns for a specified view in the "Site Pages" list.
* **Parameters**:
  + viewId: The ID of the view to retrieve columns for.
* **Returns**: A promise that resolves to an array of IColumnInfo objects.
* **Implementation**:
  + Fetches the fields associated with a particular view.
  + Retrieves detailed field information, including internal and display names, column type, and rendering logic.

#### **getListDetailsByName**

* **Purpose**: Retrieve details of a SharePoint list by its name.
* **Parameters**:
  + listName: The name of the list to retrieve.
* **Returns**: A promise that resolves to the list details.
* **Implementation**:
  + Uses the PnP JS library to fetch the list details by title.

### **createListItem**

**Purpose**:  
Create a new item in a specified SharePoint list.

**Parameters**:

* itemData (any): The data for the item to be created. This should be an object containing the field values for the new list item.
* listTitle (string): The title of the SharePoint list where the item will be created.

**Returns**:

* A promise that resolves to the created list item.

**Description**:  
This method adds a new item to a SharePoint list specified by its title. It uses the itemData parameter to populate the fields of the new item. Upon successful creation, the method returns the added item. If an error occurs during the process, it logs the error to the console and rethrows the error.

### **c. Key Interfaces**

#### **ITerm**

* Represents a taxonomy term with properties like Id, Name, parentId, and optional children.

#### **TermSet**

* Represents a set of taxonomy terms, with a setId and an array of ITerm objects.

#### **FilterDetail**

* Represents a filter applied to the list query, including the column name, column type, and values to filter by.

#### **IColumnInfo**

* Describes a column in the list, including internal and display names, minimum and maximum widths, column type, and a render function.

### **d. Utilities Used**

* **getColumnMaxWidth and getColumnMinWidth**: Utility functions to determine the maximum and minimum width of a column.
* **CellRender**: A common utility used to render a column based on its type and content.

5. Pages Columns  
  
**PagesColumns Function**

This function creates an array of column configurations (IColumn[]) for a DetailsList component. Each column configuration is derived from the IColumnInfo[] you pass in, which defines how the columns should be rendered.

### **Key Parameters**

1. **columns (IColumnInfo[]):**
   * An array of column definitions, each containing properties like InternalName, DisplayName, MinWidth, MaxWidth, and OnRender to control the rendering and layout of each column.
2. **onColumnClick (Function):**
   * A callback function triggered when a column header is clicked. Typically, this is used to handle sorting logic.
3. **sortBy (string):**
   * Specifies the column that is currently being used for sorting.
4. **isDescending (boolean):**
   * Indicates if the current sort order is descending.
5. **setShowFilter (Function):**
   * A callback function to toggle the visibility of filter options for a specific column.

### **Column Configuration (baseColumns)**

For each column, the function creates an IColumn object with the following attributes:

* **key, name, fieldName:** Derived from InternalName and DisplayName to map each column to its respective data field and label.
* **minWidth, maxWidth:** Set according to MinWidth and MaxWidth from the IColumnInfo array, ensuring the columns have appropriate width constraints.
* **isRowHeader, isResizable, isPadded:** These flags are set to ensure proper alignment, padding, and that rows are resizable.
* **isSorted, isSortedDescending:** Determines the sort state of each column by comparing the column name with the sortBy parameter.
* **onRenderHeader:**
  + A custom rendering function for the column header. It utilizes the HeaderRender method to render the header with sorting and filtering functionalities.
* **onRender:**
  + A custom cell rendering function. If the column has a custom OnRender function defined in IColumnInfo, it will be used; otherwise, it defaults to rendering the raw data in the cell.

### **Status Column**

The function adds an extra column called status:

* **Subscription Status (SubscriptionStatus Component):**
  + This column visually indicates the subscription status of an item using an Icon. If the item is subscribed, it shows a solid bell icon (RingerSolid), otherwise, a regular bell icon (Ringer).

### **Return Value**

The function returns an array of columns that include the dynamically generated columns based on IColumnInfo and an additional status column.