Exercise – Use PnpJs for SharePoint API interaction

In this exercise, you'll create a SharePoint Framework (SPFx) web part that will get and display data from a SharePoint list using the PnPJs library.

This exercise starts with the project at the end of Exercise 1. If you would rather start fresh, complete Exercise 1 first.

## Step 1 - Create a new copy of Exercise 1

Copy the folder that contains Exercise 1 to a new location.

## Update the solution dependencies

Add the PnPJs libraries with the following command

npm install @pnp/sp --save

Open the project in VS Code, and open the SpFxHttpClientDemoWebPart.ts file

Add the following under the existing imports:

import { sp } from "@pnp/sp/presets/all";

To reduce the bundle size, the above line can be replaced to just import the modules needed instead of “all” modules

import { sp } from "@pnp/sp";

import "@pnp/sp/webs";

import "@pnp/sp/lists";

import "@pnp/sp/items";

import { IItemAddResult } from "@pnp/sp/items";

In the SpFxHttpClientDemoWebPart class, add the following under the \_countries property:

protected async onInit(): Promise<void> {

await super.onInit();

// other init code may be present

sp.setup(this.context);

}

In the SpFxHttpClientDemoWebPart class, perform the following changes:

Comment out the \_getListItems function below and replace with the following:

  private async \_getListItems() {

    return await sp.web.lists.getByTitle("Countries").items.select("Title", "Id").get();

  }

Comment out the \_addListItem function below and replace with the following:

private async \_addListItem() {

    return await sp.web.lists.getByTitle("Countries").items.add({

      Title: new Date().toUTCString()

    });

  }

Comment out the \_updateListItem function below and replace with the following:

private async \_updateListItem() {

  const items: any[] = await sp.web.lists.getByTitle("Countries").items.top(1).filter("Title eq 'United States'").get();

  if (items.length > 0) {

    const updatedItem = await sp.web.lists.getByTitle("Countries").items.getById(items[0].Id).update({

      Title: "USA",

    });

  }

  return

}

Comment out the \_deleteListItem function below and replace with the following:

private async \_deleteListItem() {

  // get the last item

  const items: any[] = await sp.web.lists.getByTitle("Countries").items.orderBy("Id", false).top(1).get();

  if (items.length > 0) {

    await sp.web.lists.getByTitle("Countries").items.getById(items[0].Id).delete();

  }

}

## Step 3 - Test the web part

Start the local web server and test the web part in the hosted workbench:

gulp serve

The browser will loads the local workbench, but you cannot use this for testing because there is no SharePoint context in the local workbench. Instead, navigate to the SharePoint Online site where you created the **Countries** list, and load the hosted workbench at **https://[sharepoint-online-site]/\_layouts/workbench.aspx**.

Add the web part to the page: Select the **Add a new web part** control...

...then select the expand toolbox icon in the top-right...

...and select the **SPFxHttpClientContent** web part to add the web part to the page.

The web part will appear on the page with a single button and no data in the list.

Select the **Get Countries** button and examine the results returned. Scroll to the bottom of the list and notice there's no entry with a timestamp for the **Title**.

Select the **Add List item** button and scroll to the end of the results returned. Notice the new item that appears with a timestamp as the **Title**.

Ensure there's an item in the list with the title equal to **United States**. Test the update process by selecting the **Update List Item** button.

Notice after selecting the button, the title has changed from **United States** to **USA**.

Note the title of the last item in the list. Test the delete process by selecting the **Delete List Item** button.

Notice after selecting the button, the last item in the list has been removed.

Stop the local web server by pressing CTRL+C in the console/terminal window.

## Summary

In this exercise, you extended the SharePoint Framework to interact with data using the PnpJs library. Notice how much simpler the code is and possibly faster too!