Connecting to PnP

Nicolae Caprarescu
FULL-STACK ENGINEER
www.properjava.com

Module Overview

Install CLI and PnP cmdlets

Rename our project

Manage the application lifecycle using PnP

Update the application content using PnP

Globomantics Requirements

Globomantics wants to be able to release updates to the app regularly

Globomantics wants to more personal information about the current user to help them perform their jobs efficiently

Globomantics also wants to be able to remove the app from the app catalog

PnP CLI

Open source initiative

Manage your Microsoft 365 tenant

Can be installed on any platform

Manage SPFx projects

Build automation scripts

Demo

Install the CLI

PnP PowerShell cmdlets

Manage your Microsoft 365 tenant

Can be installed on Windows only

Demo

Install the PowerShell PnP cmdlets library

Demo

Rename our skeleton application

Demo

Deploy our application using PnP cmdlets

Demo

Update the application

Demo

Redeploy the application using PnP cmdlets

Demo

Remove the application using PnP CLI

Demo

Further updates to the application using PnP to connect to Microsoft Graph

```
npm install react@16.8.5

npm install react-dom@16.8.5

npm install @types/react@16.8.8 --save-dev

npm install @types/react-dom@16.8.3 --save-dev

npm install office-ui-fabric-react@6.214.0
```

◀ We want to add React components to our application, and to do this we need to install more npm packages.

```
{
  "resource": "Microsoft Graph",
  "scope": "Calendars.Read"
},
{
  "resource": "Microsoft Graph",
  "scope": "MailboxSettings.Read"
}
```

- We also need more permissions from Microsoft Graph
- Open config/package-solution.json and add these permissions
- These allow us to read and write calendar entries and to read the mailbox settings to get the user's timezone.

```
"AllDay": "All day",

"Error": "An error has occurred while retrieving your up
coming meetings",

"Heading": "Today's Meetings",

"Hour": "hour",

"Hours": "hours",

"Loading": "Retrieving your upcoming meetings",

"Minutes": "minutes",

"NoMessages": "You have no upcoming meetings",

"ViewAll": "View the full list"
```

- Now we need to add to the environment strings for the app
- First, we add these to the en-us.js file

```
AllDay: string;

Error: string;

Heading: string;

Hour: string;

Hours: string;

Loading: string;

Minutes: string;

NoMeetings: string;

ViewAll: string;
```

- **Then we modify the mystring.d.ts file to map them to variables**

```
export interface IMeetings {
  value: IMeeting[];
 }

 export interface IMeeting {
  end: IMeetingTime;
  isAllDay: boolean;
  location: {
    displayName: string;
  };
  showAs: string;
  start: IMeetingTime;
  subject: string;
  webLink: string;
 }

 export interface IMeetingTime {
  dateTime: string;
  timeZone: string;
 }
```

◀ **This class will be used for the extracted details for each meeting**

IPersonalCalendarProps.ts

```
import { MSGraphClient } from "@microsoft/sp-http";

export interface IPersonalCalendarProps {
  graphClient: MSGraphClient;
}
```

IPersonalCalendarState.ts

```
import { IMeeting } from '.';

export interface IPersonalCalendarState {
  error: string;
  loading: boolean;
  meetings: IMeeting[];
}
```

PersonalCalendar.module.scss

```
@import '~office-ui-fabric-react/dist/sass/References.scss';

.personalCalendar {
  @include ms-font-m;

  .list {
    margin-top: 1em;
    margin-bottom: 1em;
  }

  .meetingWrapper {
    padding-left: 7px;

    &:global(.tentative) {
      background: $ms-color-themePrimary url('data:image/gif;base64,R0IGODlhBwA...')
    }

    &:global(.busy) {
      background: $ms-color-themePrimary;
    }
```

(truncated for brevity)

PersonalCalendar.tsx (part 1)

```
import * as React from 'react';
import styles from './PersonalCalendar.module.scss';
import * as strings from 'GloboSkeletonWebPartStrings';
import { IPersonalCalendarProps, IPersonalCalendarState, IMeeting, IMeetings } from '.';
import { Spinner, SpinnerSize } from 'office-ui-fabric-react/lib/components/Spinner';
import { List } from 'office-ui-fabric-react/lib/components/List';
import { Link } from 'office-ui-fabric-react/lib/components/Link';

export default class PersonalCalendar extends React.Component<IPersonalCalendarProps, IPersonalCalend
arState> {
  constructor(props: IPersonalCalendarProps) {
    super(props);

    this.state = {
      meetings: [],
      loading: true,
      error: undefined
    };
    this._loadMeetings();
  }
}
```

PersonalCalendar.tsx (part 2)

```
private _getTimeZone(): Promise<string> {
  return new Promise<string>((resolve, reject) => {
    this.props.graphClient
      // get the mailbox settings
      .api(`me/mailboxSettings`)
      .version("v1.0")
      .get((err: any, res: microsoftgraph.MailboxSettings): void => {
        resolve((res.timeZone || 'GMT Standard Time'));
      });
  });
}
```

PersonalCalendar.tsx (part 3)

```
private _loadMeetings(): void {
  if (!this.props.graphClient) {
    return;
  }

  this.setState({
    error: null,
    loading: true,
    meetings: []
  });

  const date: Date = new Date();
  const now: string = date.toISOString();
  date.setUTCHours(23);
  date.setUTCMinutes(59);
  date.setUTCSeconds(0);
  date.setDate(date.getDate());
  const midnight: string = date.toISOString();

this._getTimeZone().then(timeZone => {
    this.props.graphClient
      .api(`me/calendar/calendarView?startDateTime=${now}&end
DateTime=${midnight}`)
      .version("v1.0")
      .select('subject,start,end,showAs,webLink,location,isAllDay')
      .top(20)
      .header("Prefer", "outlook.timezone=" + '"' + timeZone + '"')
      .orderby("start/dateTime")
      .get((err: any, res: IMeetings): void => {
        if (err) {
          this.setState({
            error: err.message ? err.message : strings.Error,
            loading: false
          });
          return;
        }

        if (res && res.value && res.value.length > 0) {
          this.setState({
            meetings: res.value,
            loading: false
          });
        }
        else {
          this.setState({
            loading: false
          });
        }
      });
  });
}
```

PersonalCalendar.tsx (part 4)

```
private _getDuration = (meeting: IMeeting): string => {
  if (meeting.isAllDay) {
    return strings.AllDay;
  }

  const startDateTime: Date = new Date(meeting.start.dateTime);
  const endDateTime: Date = new Date(meeting.end.dateTime);
  // get duration in minutes
  const duration: number = Math.round((endDateTime as any) - (startDateTime as any)) / (1000 * 60);
  if (duration <= 0) {
    return '';
  }

  if (duration < 60) {
    return `${duration} ${strings.Minutes}`;
  }

  const hours: number = Math.floor(duration / 60);
  const minutes: number = Math.round(duration % 60);
  let durationString: string = `${hours} ${hours > 1 ? strings.Hours : strings.Hour}`;
  if (minutes > 0) {
    durationString += ` ${minutes} ${strings.Minutes}`;
  }

  return durationString;
}
```

PersonalCalendar.tsx (part 5)

```
private _onRenderCell = (item: IMeeting, index: number | undefined): JSX.Element => {
  const startTime: Date = new Date(item.start.dateTime);
  const minutes: number = startTime.getMinutes();

  return <div className={`${styles.meetingWrapper} ${item.showAs}`}>
    <Link href={item.webLink} className={styles.meeting} target='_blank'>
      <div className={styles.start}>{`${startTime.getHours()}:${minutes < 10 ? '0' + minutes : minutes}`}</div>
      <div className={styles.subject}>{item.subject}</div>
      <div className={styles.duration}>{this._getDuration(item)}</div>
      <div className={styles.location}>{item.location.displayName}</div>
    </Link>
  </div>;
}
```

PersonalCalendar.tsx (part 6)

```
public render(): React.ReactElement<IPersonalCalendarProps> {
  return (
    <div className={styles.personalCalendar}>
    <div id='root'></div>
      {
        this.state.loading &&
        <Spinner label={strings.Loading} size={SpinnerSize.large} />
      }

      {
        this.state.meetings &&
          this.state.meetings.length > 0 ? (
            <div>
              <h2>{strings.Heading}</h2>
              <List items={this.state.meetings}
                onRenderCell={this._onRenderCell} className={styles.list} />
              <Link href='https://outlook.office.com/owa/?path=/calendar/view/Day' target='_blank'>{strings.ViewAll}</Link>
            </div>
          ) : (
            !this.state.loading && (
              this.state.error ?
                <span className={styles.error}>{this.state.error}</span> :
                <span className={styles.noMeetings}>{strings.NoMeetings}</span>
            )
          )
      }
    </div>
  );
}
```

index.ts

```
export * from './IMeeting';
export * from './IPersonalCalendarProps';
export * from './IPersonalCalendarState';
export * from './PersonalCalendar';
```

GloboSkeletonWebPart.ts (part 1)

```
import * as React from 'react';

import * as ReactDom from 'react-dom';

import PersonalCalendar from './components/PersonalCalendar';

import { IPersonalCalendarProps } from './components/IPersonalCalendarProps';
```

GloboSkeletonWebPart.ts (part 2)

```
private _renderAgenda(client: MSGraphClient): void {
  const element: React.ReactElement<IPersonalCalendarProps> = React.createElement(
    PersonalCalendar,
    {
      graphClient: client
    }
  );

  ReactDom.render(element, this.domElement);
}
```

GloboSkeletonWebPart.ts (part 3)

```
.get((error, userProfile: any, rawResponse?: any) => {
  const spRoot: Element = this.domElement.querySelector('#root');
  spRoot.innerHTML = `




this._renderAgenda(client);
```

Demo

See the updated web part

Microsoft Graph Facts

Launched in 2015

Builds on Microsoft 365 APIs

Allows developers to integrate services with the Microsoft online ecosystem

https://docs.microsoft.com/graph

Module Summary

Installed the CLI and PnP cmdlets

Used the CLI and PnP cmdlets to manage the application

Added content to show the current user’s agenda on the web part