Copy Views

Building a copy views webpart and extension with SPFx

About me

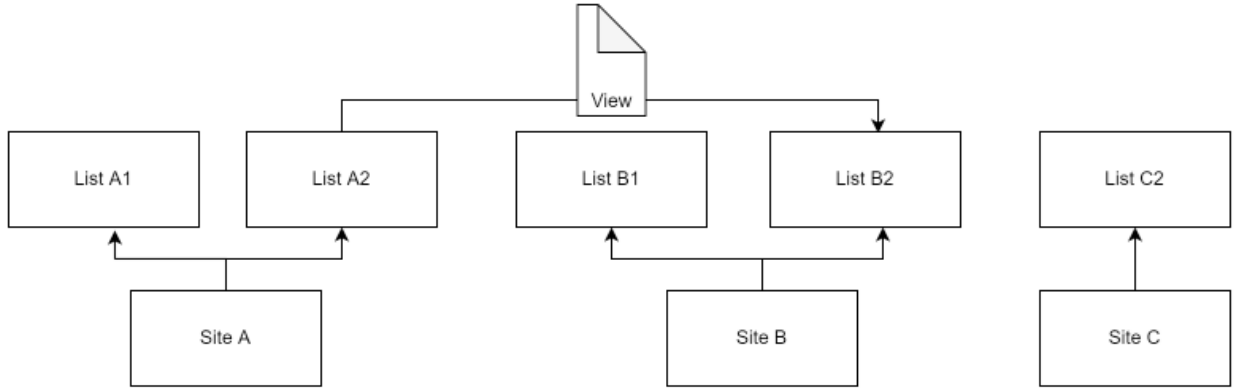
- Martin Lingstuyl
- Microsoft 365 Architect
- Microsoft MVP
- I4-YOU Business Solutions (The Netherlands)
- Family man 👨‍👩‍👧‍👧
- Cycling 🚴 💨

- **Twitter:** @martinlingstuyl
- **Blog:** https://www.blimped.nl
- **GitHub:** @martinlingstuyl

PRODUCTIVITY EXPERTS

Scenario

- Allow users to copy views that exist in a list or library towards other lists or libraries.

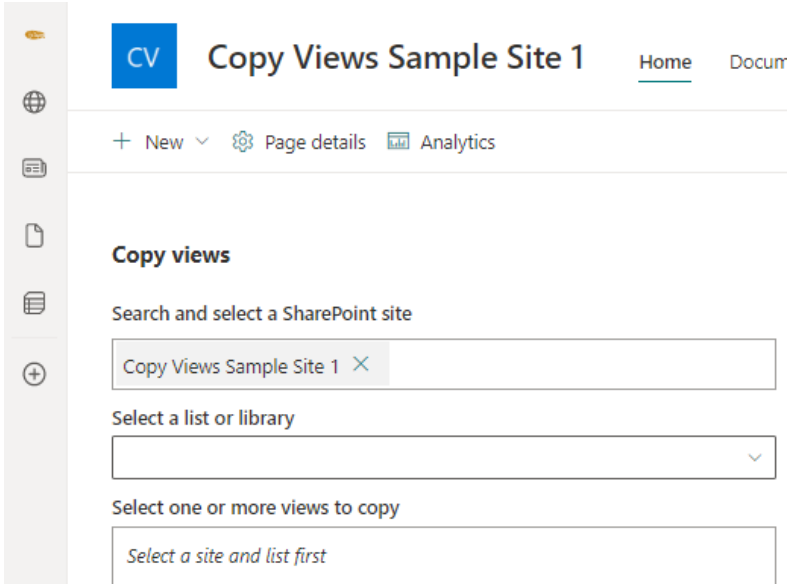


- Will copy:
  - Which columns are shown
  - Sorting
  - Group by
  - View formatting
  - Filtering

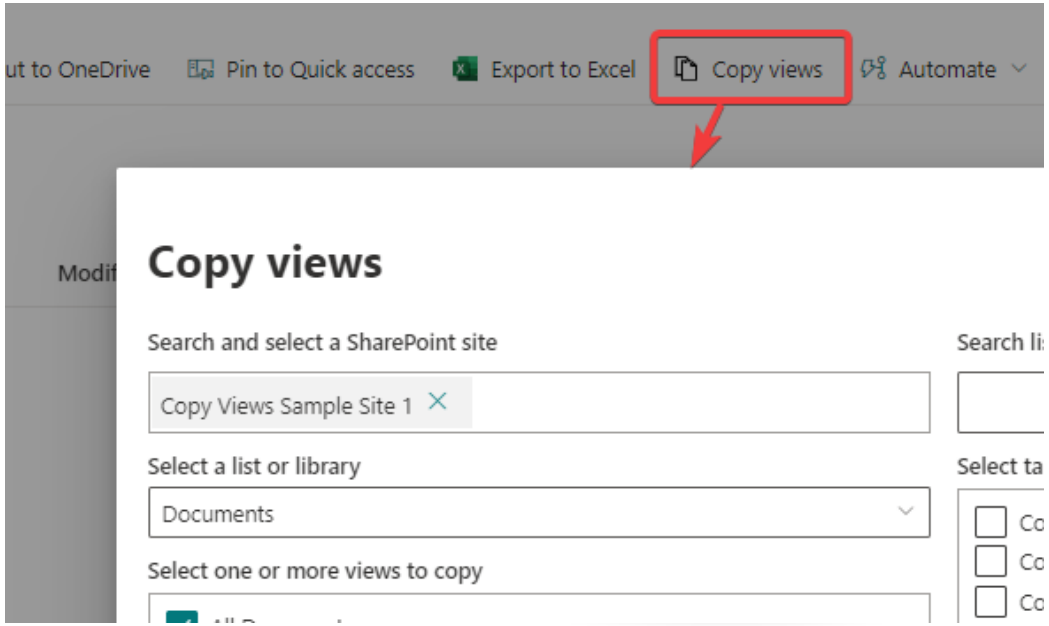
- When a column is missing in the target list, that part of the view is not copied

SPFx

- **Webpart**
  To display on a specific site or page for view management purposes.



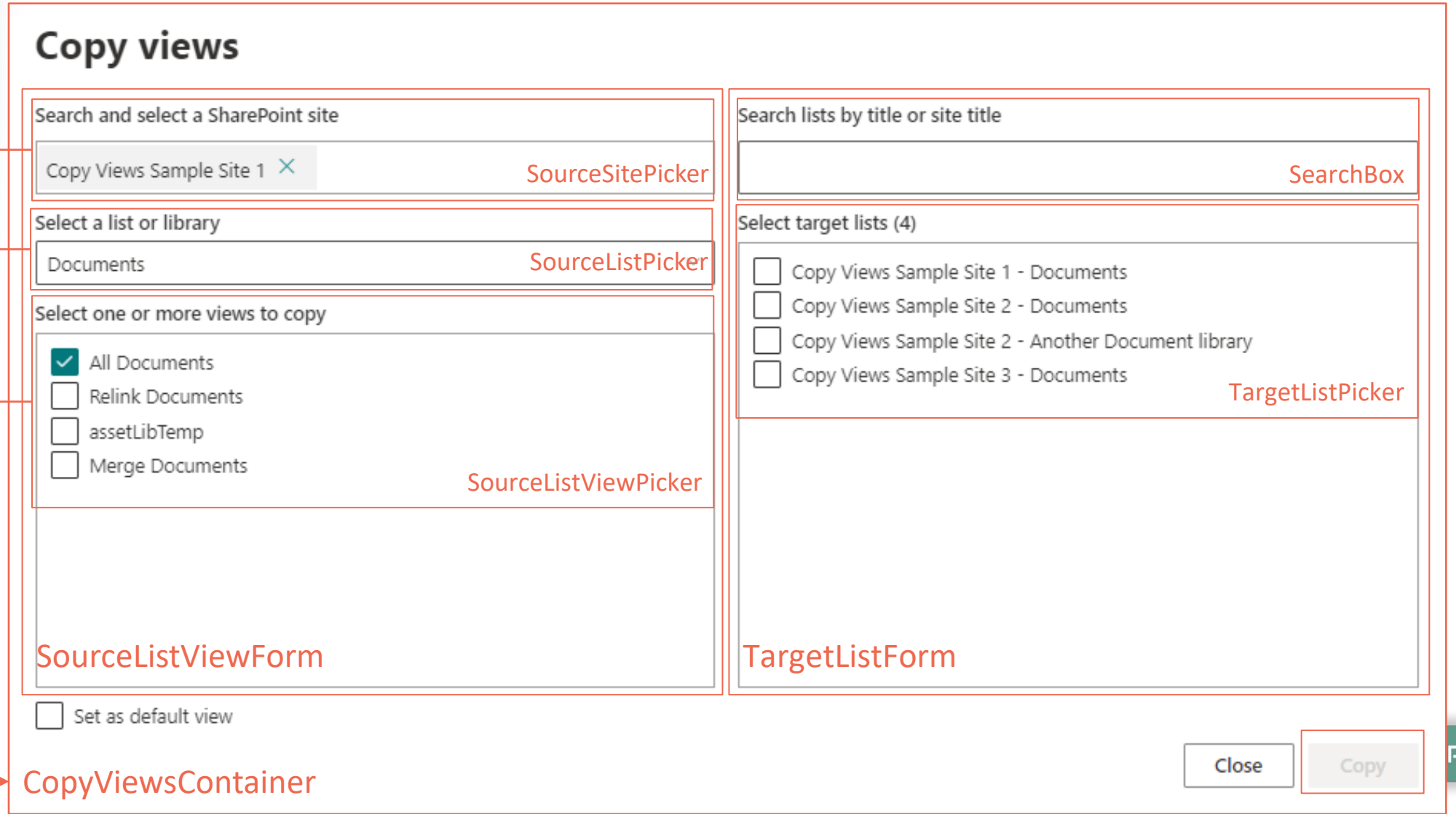
- **Extension**
  As a dialog opened by a ListView Command set extension on any list or library, tenant wide or site scoped.



Demo

PRODUCTIVITY EXPERTS

Component Structure 1/3



Component Structure 2/3

```
<CopyViewsContainer … />

    <SourceListViewForm … />

        <SourceSitePicker … />
        <SourceListPicker … />
        <SourceListViewPicker … />


<PrimaryButton … />
```

Passing props & defaults values down

Passing choices up (through functions)

Component Structure 3/3

Starting copy tasks

```
121
122     const copyTasks: ICopyTask[] = [];
123     let index = 0;
124
125     sourceViews.forEach(sourceView => {
126       targetLists.forEach(targetList => {
127         index++;
128         copyTasks.push({ index, sourceView, targetList, state: CopyTaskState.Busy });
129       });
130     });
131
132     this.setState({ showCopyStatus: true, copyTasks }, async () => {
133       await Promise.all(copyTasks.map(copyTask => this._copyView(copyTask)));
134     });
```

- Get a list of ‘copy tasks’ (one view to one list)
- Set the tasks on the state.
- await setState completion because we want the screen to display
- Parallel execution using await Promise.all

Copying a View (PnP.JS) 1/6

```
64          const sourceWeb = Web([this._sp.web, sourceView.siteUrl]);
65          const sourceList = sourceWeb.lists.getById(sourceView.listId);
66          const sourceViewInfo = await sourceList.getView(sourceView.id)();
67          const sourceViewFields = await sourceList.getView(sourceView.id).fields();
68
69          const targetWeb = Web([this._sp.web, targetSiteUrl]);
70          const targetList = targetWeb.lists.getById(targetListId);
71          const targetListInfo = await targetList.expand("Views", "Fields").select("Views/Id", "Views/ServerRelativeUrl", "Fields/InternalName")();
72
73          const targetView = targetListInfo.Views.filter(view => {
74              const viewFileName = view.ServerRelativeUrl.substring(view.ServerRelativeUrl.lastIndexOf('/') + 1);
75              return viewFileName === sourceView.fileName;
76          })[0];
77
```

- **Step 1**: get everything that we need:
  - Source list, selected view and view fields
  - Target list, view(s), view fields

Copying a View (PnP.JS) 2/6

```
92    private _buildProperties = (sourceView: IViewInfo, targetList: IListInfo, setAsDefaultView: boolean): IViewInfo => {
93        const properties = {
94            CustomFormatter: sourceView.CustomFormatter,
95            RowLimit: sourceView.RowLimit,
96            Hidden: sourceView.Hidden,
97            IncludeRootFolder: sourceView.IncludeRootFolder,
98            JSLink: sourceView.JSLink,
99            Paged: sourceView.Paged,
100           Scope: sourceView.Scope,
101           TabularView: sourceView.TabularView,
102           Title: sourceView.Title,
103           ViewQuery: this._buildViewQuery(sourceView, targetList),
104           ViewType2: sourceView.ViewType2
105       } as IViewInfo;
106
107       if (setAsDefaultView) {
108           properties.DefaultView = true;
109       }
110
111       return properties;
112   }
```

- **Step 2**: build a view properties object

Copying a View (PnP.JS)


```
<GroupBy Collapse="TRUE" GroupLimit="30">
    <FieldRef Name="SomeColumnThatOnlyExsistsOnSourceList" />
</GroupBy>
<OrderBy>
    <FieldRef Name="FileLeafRef" />
</OrderBy>
<Where>
    <And>
        <Neq>
            <FieldRef Name="SomeColumnThatOnlyExsistsOnSourceList" />
            <Value Type="Text">SomeValue</Value>
        </Neq>
        <Gt>
            <FieldRef Name="ID" />
            <Value Type="Counter">0</Value>
        </Gt>
    </And>
</Where>
```

```
<OrderBy>
    <FieldRef Name="FileLeafRef" />
</OrderBy>
<Where>
    <Gt>
        <FieldRef Name="ID" />
        <Value Type="Counter">0</Value>
    </Gt>
</Where>
```

Copying a View (PnP.JS) 4/6

```
117     private _buildViewQuery = (sourceView: IViewInfo, targetList: IListInfo): string => {
118
119         const domParser = new DOMParser();
120         const sourceViewDoc = domParser.parseFromString("<Root>" + sourceView.ViewQuery + "</Root>", "text/xml");
121         const elementsToRemove = this._getFieldRefsToRemove(sourceViewDoc, targetList);
122
123         if (elementsToRemove.length === 0) {
124             return sourceView.ViewQuery;
125         }
126         else {
127             for (let i = 0; i < elementsToRemove.length; i++) {
128                 this._recursiveDeleteElement(elementsToRemove[i]);
129             }
130
131             this._ensureOrderByElement(sourceViewDoc);
132             this._fixAndOrConditions(sourceViewDoc);
133
134             return sourceViewDoc.firstElementChild.innerHTML;
135         }
136     }
```

- **Step 3**: build the view query, XML-manipulation necessary employ the DOMParser

Copying a View (PnP.JS) 5/6

- Get all elements by tag name:

```
const fieldRefs = sourceViewDoc.getElementsByTagName("FieldRef");
```

- Get sibling 'Value' elements:

```
fieldRefs[i].nextElementSibling.nodeName === "Value"
```

- Remove those elements:

```
const parentElement = element.parentElement;
parentElement.removeChild(element);
```

- For And/Or replace with child:

```
elements[i].replaceWith(elements[i].firstChild);
```

- Reference CAML:

```
<GroupBy Collapse="TRUE" GroupLimit="30">
    <FieldRef Name="SomeColumnThatOnlyExsistsOnSourceList" />
</GroupBy>
<OrderBy>
    <FieldRef Name="FileLeafRef" />
</OrderBy>
<Where>
    <And>
        <Neq>
            <FieldRef Name="SomeColumnThatOnlyExsistsOnSourceList" />
            <Value Type="Text">SomeValue</Value>
        </Neq>
        <Gt>
            <FieldRef Name="ID" />
            <Value Type="Counter">0</Value>
        </Gt>
    </And>
</Where>
```

Copying a View (PnP.JS) 6/6

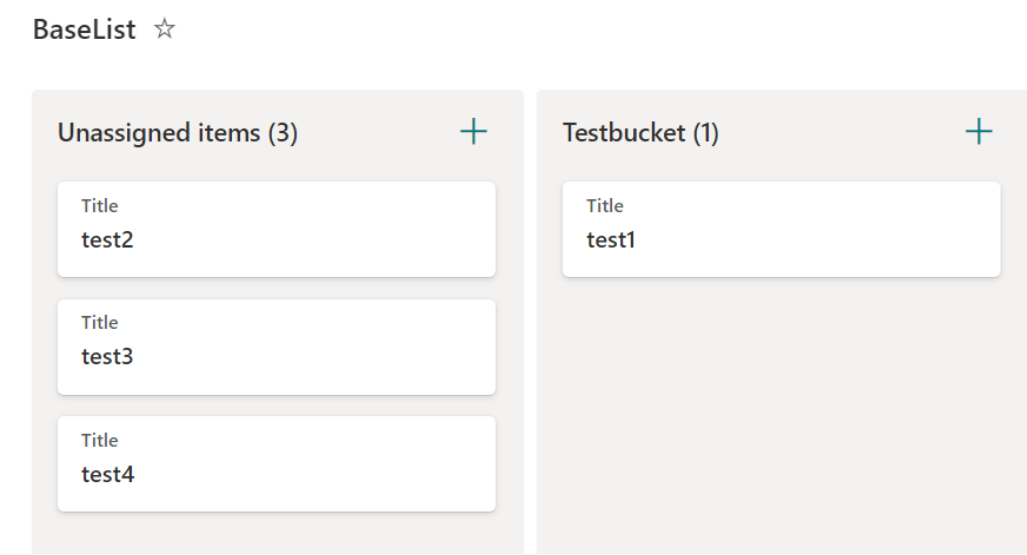
```
80          if (!ta
81              cor
82
83              awa
84          }
85          else {
86              cor
87
88              awa
89          }
```

```
200     private _updateViewFields = async (targetView: IView, sourceViewFields: { Items: string[]; SchemaXml: string; }, targetList: IListInfo)
201
202         const viewFields: string[] = [];
203         sourceViewFields.Items.forEach(item => {
204             if (targetList.Fields.some(field => field.InternalName === item)) {
205                 viewFields.push(item);
206             }
207         });
208
209
210         if (viewFields.length > 0) {
211             await targetView.fields.removeAll();
212
213             for (let i = 0; i < viewFields.length; i++) {
214                 await targetView.fields.add(viewFields[i]);
215             }
216         }
217     }
```

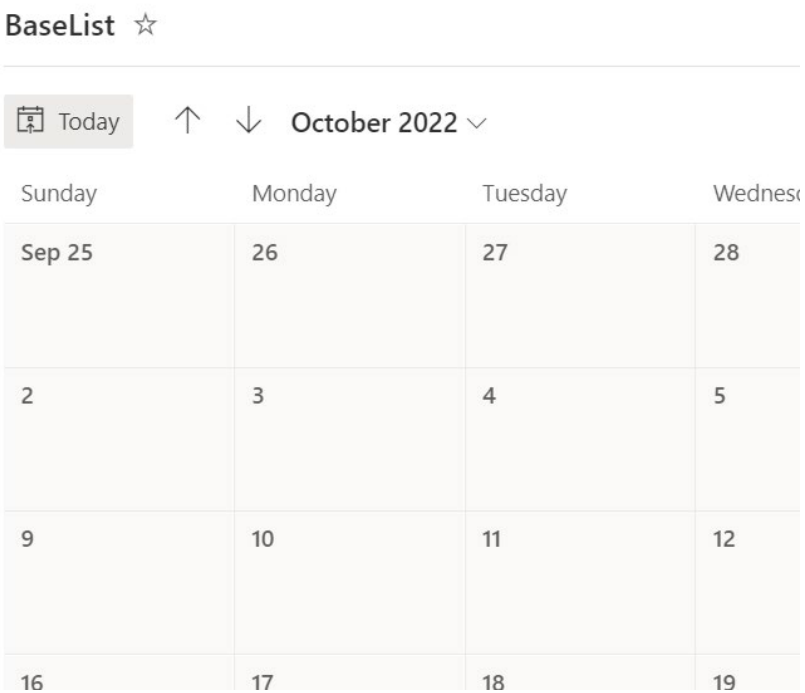
- **Step 4**: copy or update the view using PnPjs
  - + update view fields

What’s currently Not supported

- Kanban board type views



- Modern calendar type views



- Why: copying fields necessary
- Contributions?: 🙋

PRODUCTIVITY EXPERTS

Resources

- **PnP Sample**
  https://github.com/pnp/sp-dev-fx-webparts/tree/main/samples/react-copy-views

- **PnPjs library – Get, Create, Update Views**
  https://pnp.github.io/pnpjs/sp/views/

- **DOMParser Browser API**
  https://developer.mozilla.org/en-US/docs/Web/API/DOMParser