|  |
| --- |
| import { override } from '@microsoft/decorators'; |
|  | import { Log } from '@microsoft/sp-core-library'; |
|  | import { |
|  | BaseListViewCommandSet, |
|  | Command, |
|  | IListViewCommandSetListViewUpdatedParameters, |
|  | IListViewCommandSetExecuteEventParameters |
|  | } from '@microsoft/sp-listview-extensibility'; |
|  | import { Dialog } from '@microsoft/sp-dialog'; |
|  |  |
|  | import \* as strings from 'MyListEventsCommandSetStrings'; |
|  |  |
|  | /\*\* |
|  | \* If your command set uses the ClientSideComponentProperties JSON input, |
|  | \* it will be deserialized into the BaseExtension.properties object. |
|  | \* You can define an interface to describe it. |
|  | \*/ |
|  | export interface IMyListEventsCommandSetProperties { |
|  | // This is an example; replace with your own properties |
|  | sampleTextOne: string; |
|  | sampleTextTwo: string; |
|  | } |
|  |  |
|  | const LOG\_SOURCE: string = 'MyListEventsCommandSet'; |
|  |  |
|  | export default class MyListEventsCommandSet extends BaseListViewCommandSet<IMyListEventsCommandSetProperties> { |
|  |  |
|  | @override |
|  | public onInit(): Promise<void> { |
|  | Log.info(LOG\_SOURCE, 'Initialized MyListEventsCommandSet'); |
|  | return Promise.resolve(); |
|  | } |
|  |  |
|  | @override |
|  | public onListViewUpdated(event: IListViewCommandSetListViewUpdatedParameters): void { |
|  |  |
|  | const GetItemButton: Command = this.tryGetCommand('GetItemButton'); |
|  | const GetItemsCommand: Command = this.tryGetCommand('MultipleItemsButton'); |
|  | const AlertMessageCommand: Command = this.tryGetCommand('AlertMessageButton'); |
|  |  |
|  | if (GetItemButton) { |
|  | GetItemButton.visible = event.selectedRows.length === 1; |
|  | } |
|  | if(GetItemsCommand) |
|  | { |
|  | GetItemsCommand.visible = event.selectedRows.length > 0; |
|  | } |
|  | if(AlertMessageCommand) |
|  | { |
|  | AlertMessageCommand.visible = event.selectedRows.length === 0; |
|  | } |
|  | } |
|  |  |
|  | @override |
|  | public onExecute(event: IListViewCommandSetExecuteEventParameters): void { |
|  | let title = ""; |
|  | switch (event.itemId) { |
|  | case 'GetItemButton': |
|  | Dialog.alert(`${event.selectedRows[0].getValueByName("Title")}`); |
|  | break; |
|  | case 'AlertMessageButton': |
|  | Dialog.alert(`No Item Selected`); |
|  | break; |
|  | case 'MultipleItemsButton': |
|  | this.multipleitemsevent(event); |
|  | break; |
|  | case 'DeleteButton': |
|  | Dialog.prompt(`Please enter the list name to move:`).then((value:string)=> |
|  | { |
|  | title = value; |
|  | this.sendvalue(title,event); |
|  | } |
|  | ); |
|  |  |
|  | break; |
|  | default: |
|  | throw new Error('Unknown command'); |
|  | } |
|  | } |
|  |  |
|  | private sendvalue(title:string,event:IListViewCommandSetExecuteEventParameters) |
|  | { |
|  | let selectedtitle = event.selectedRows[0].getValueByName("Title"); |
|  | Dialog.alert(title + "|" + selectedtitle); |
|  | } |
|  |  |
|  | private multipleitemsevent(event:IListViewCommandSetExecuteEventParameters) |
|  | { |
|  | let title = ""; |
|  | let eventdetails =""; |
|  | for(var i=0;i<event.selectedRows.length;i++) |
|  | { |
|  | title += " " + event.selectedRows[i].getValueByName("Title") + " "; |
|  | } |
|  | Dialog.alert(`${title}`); |
|  | } |
|  | } |