# Dynamics 365 for Retail POS Extensibility

Tyler Jacoby, Mugunthan Mani and Luke Graham

Microsoft

# Agenda

How to:

- Create a new view
- Extend an existing view
- Create a new operation
- Create a trigger
- Override a POS request handler

# Creating a new view

# The Basics

- Each extension view must be defined in the manifest
- Can define separate markup for the desktop and phone views
- The “viewControllerPath” field specifies which module contains the view controller for the page
- Import required types from “PosApi/Create/Views”
- Each extension view must inherit from *ExtensionViewControllerBase*
- Use PosUISdk library to maintain a consistent look and feel

```
"create": {
  "views": [
    {
      "title": "Store Hours View",
      "pageName": "StoreHoursView",
      "phonePageName": "StoreHoursView",
      "viewDirectory": "Views/",
      "viewControllerPath": "Views/StoreHoursView"
    }
  ],
```

# Pos.UI.SDK

- Library containing modules that enable extensions to use built-in POS controls
- Each module contains a PosControl class that defines the API for that type of control
  - Used as the data provided to the corresponding knockout binding.
  - Extension view will import the PosControl class from “PosUISdk/Controls/CONTROL_NAME”
- Each control also has a Knockout binding that uses the PosControl as the binding data
  - Each binding handler is prefixed with “msPos” in order to avoid name collisions
    - Ex. msPosAppBar, msPosAppBarCommand
- Supported Controls:
  - AppBar, AppBarCommand
  - DataList
  - DatePicker
  - HeaderSplitview
  - Loader
  - Menu, MenuCommand, ToggleMenu, ToggleMenuCommand
  - Pivot, PivotItem
  - TimePicker
  - ToggleSwitch

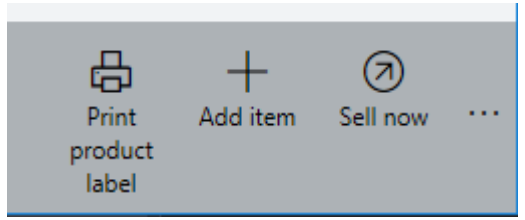
# POS Styles

- The POS styles listed below are supported for extensions to use. Other styles are not guaranteed to be backwards compatible.
- Text: h1-h6, ellipsis
- Layout: row, col, ratio, grow, width, height, margin, pad, top, bottom, left, right, positionRelative, scroll, height100Percent, width100Percent
- Themes: accentColor, accentBackground, accentBorder
- Icons: All

# Extending an existing view

# Custom App Bar Commands

- App Bar Commands provide a way to add a new button to the app bar on the Core POS Pages
- Currently supported on 4 pages
  - SimpleProductDetailsView, CustomerDetailsView, SearchView (Product and Customer Search) and ShowJournalView
  - CustomerAddEditView added in App Update 3
- Each extension command must be defined in its own module and listed in the manifest
- Should inherit from the extension command base class for the page to which it will be added
  - Page specific base classes are located in module for the page
    - Ex. import { ShowJournalExtensionCommandBase } from “PosApi/Extend/Views/ShowJournalView”
  - Page specific base classes inherit from ExtensionCommandBase



# Custom List Columns

- Custom List Columns allow extensions to override the default column set on existing POS pages in order to add, remove or reorder the columns that are displayed.
- Currently supported on 3 pages
  - SearchView (Product and Customer Search) ShowJournalView and InventoryLookupView
- Each list that supports custom column sets has a corresponding node in the manifest where the custom column set creator module should be specified.
- Modules should default export a function that returns a collection containing the custom columns.