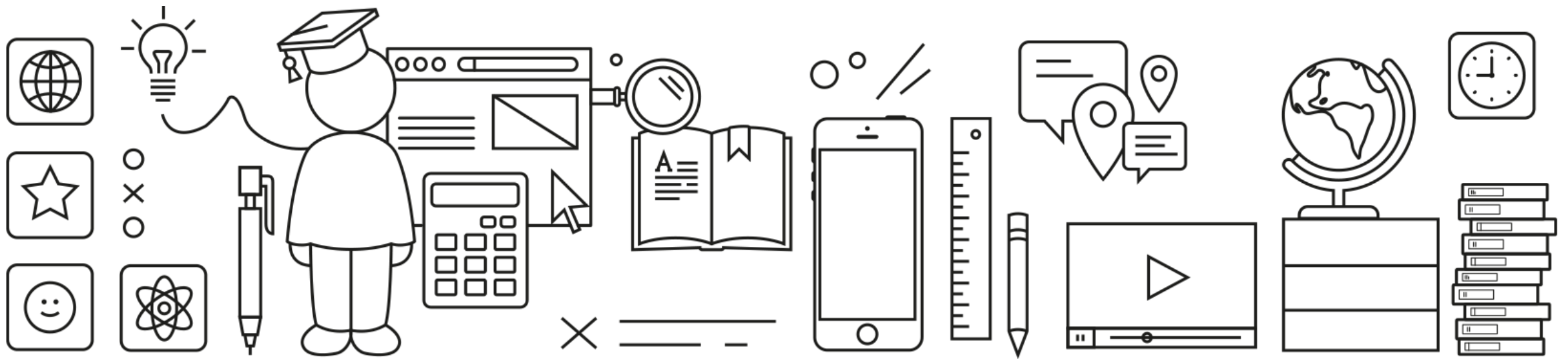**SAP Customer Experience**

SAP Commerce Cloud Backoffice Framework Developer Training
Orchestrate a Cockpit

THE BEST RUN **SAP**

Orchestration Overview
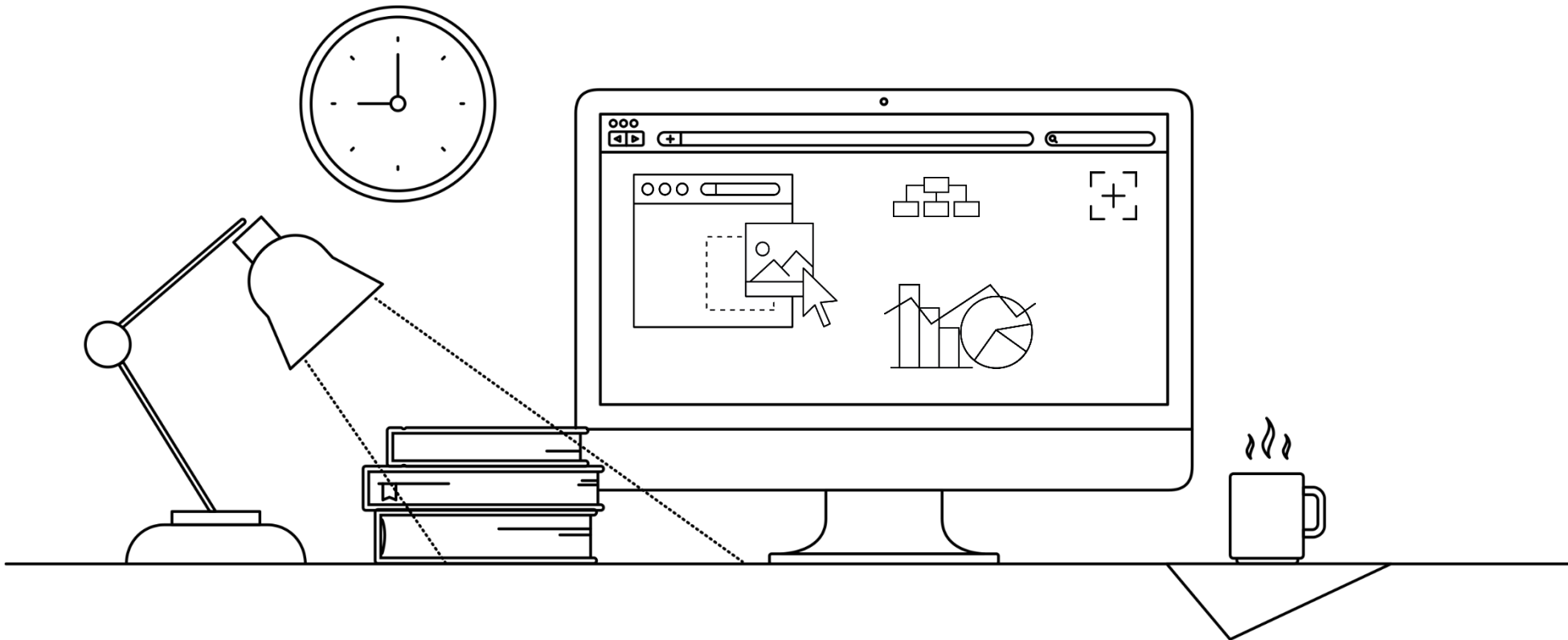
What is Orchestration?

Orchestration – run-time process of combining two or more software components to automate a business process or synchronize data

- Integration approach that allows for decoupling applications and software components
- Provides message routing, security, data transformation, and reliability
- Allows for the central management and monitoring of your integrations

Orchestrating in Backoffice

Orchestrating a Backoffice cockpit is:

– building out one or more business processes by adding and connecting widgets (*through Application Orchestrator or through* `widgets.xml`)

Steps to Orchestrate a Cockpit

1.

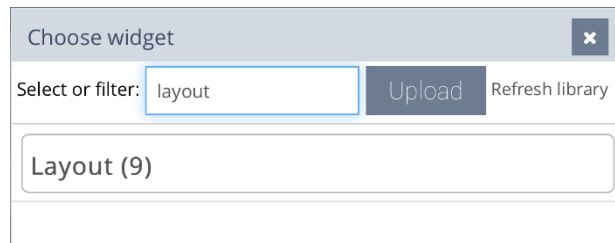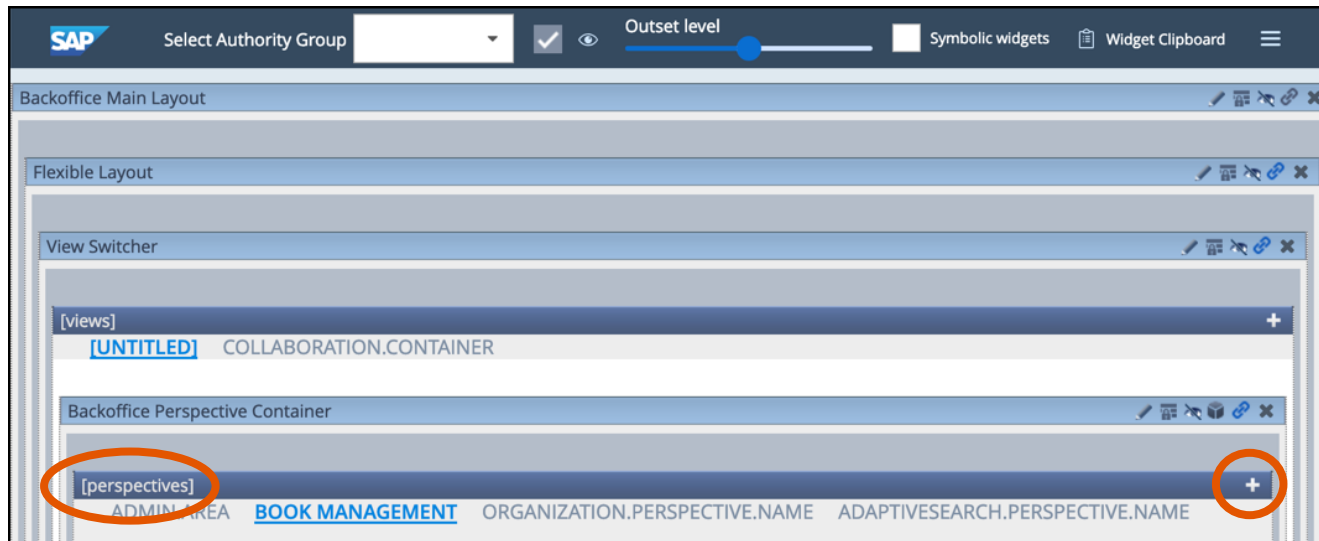Create a New Cockpit

2.

Add Widget Instances
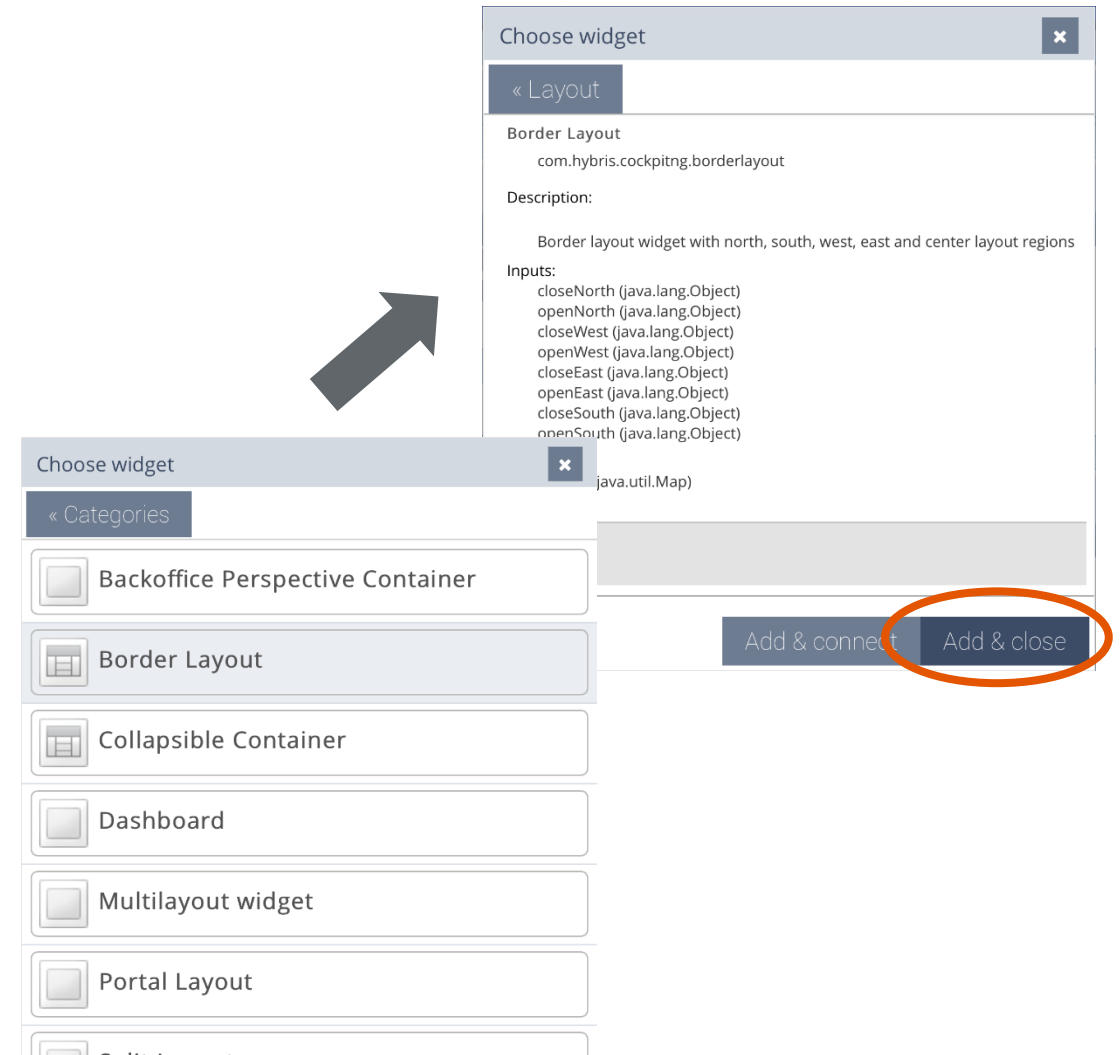
3.

Connect the Widget Instances
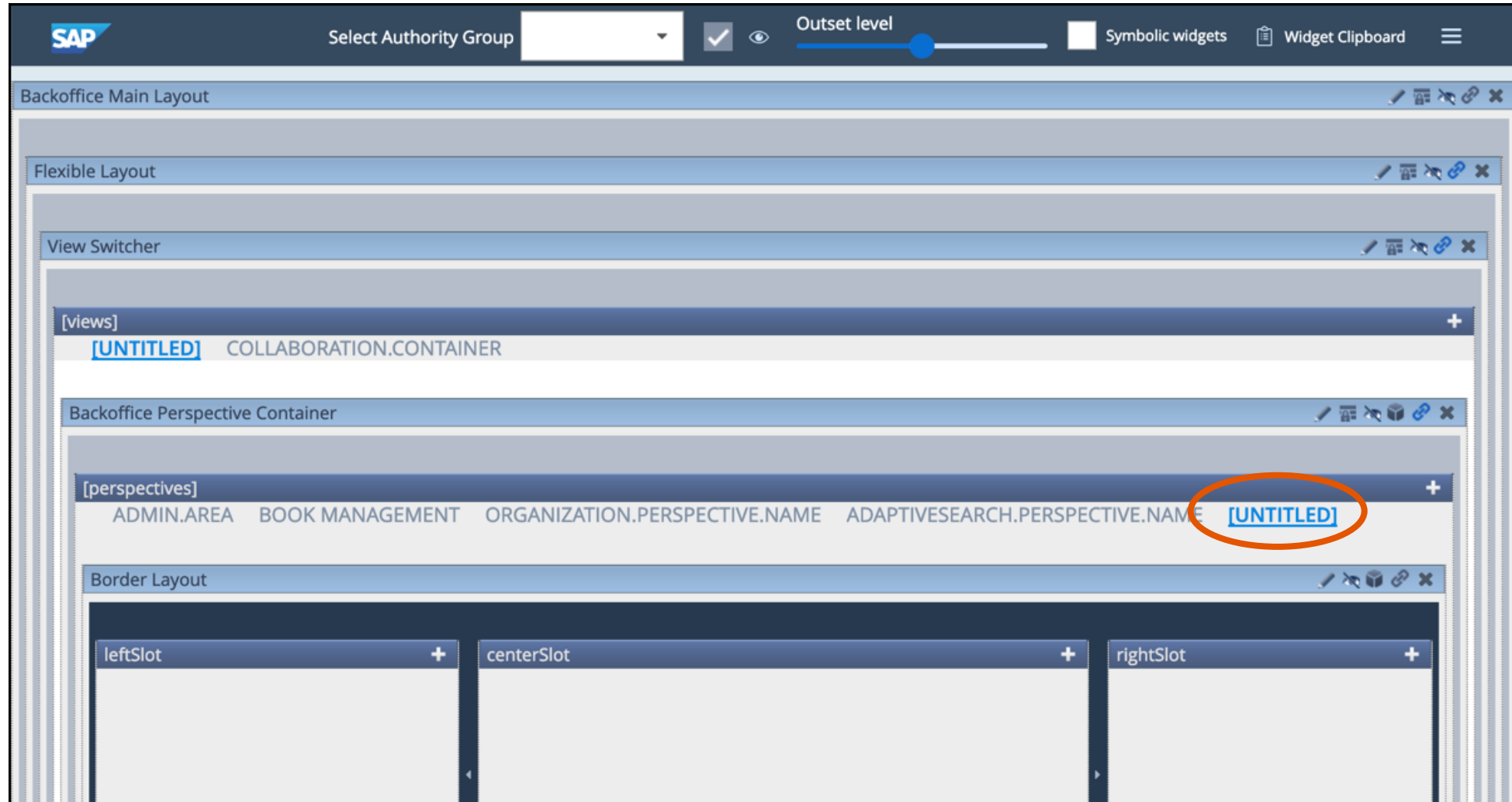
Step 1: Create a New Cockpit

Step 1: Create a new Cockpit

Type the layout widget name directly, or filter by keyword, **layout**

Step 1: Create a new Cockpit – (continued)

Step 1: Create a new Cockpit - Settings

Configure settings for your new cockpit, such as

- Title (*tab display name*)

- Widget ID (for widget instance)
  - Needs to be unique
  - Replace autogenerated value to make it meaningful

Available settings will be determined by the specific widget

Settings for com.hybris.cockpitng.borderlayout ✕

◀ **GENERAL** LOCALIZATION VIRTUAL SOCKETS ACCESS RESTRICTION ▸

Title:	My Cockpit
Widget ID:	custom-borderlayout-cockpit
autoCloseWest:	false ▾ ✖
border:	none ✖
broadcastHorizontalResize:	true ▾ ✖
centerBorder:	none ✖
eastBorder:	none ✖

ADD SETTING

CLOSE

Step 1: Create a new Cockpit - Generated XML

As you work in Application Orchestrator, you can display the merged `widgets.xml` file to view the XML that gets generated

- Search for your cockpit's XML code by its unique Widget ID (`mycustom-cockpit`)
- Cockpits can be identified in the XML by the slotId=*"perspectives"* attribute on a <widget …> tag
- All widget instances added to a cockpit can be found inside the cockpit's <widget …> </widget> tags

`custombackoffice-backoffice-widgets.xml`

```
<widgets>
  …
  <widget id="mycustom-cockpit" widgetDefinitionId="com.hybris.cockpitng.borderlayout"                    slotId="perspectives"
title="Custom Layout Cockpit" template="false">
    <widget id="collapsible" …
    …
    </widget>
  </widget>
  …
</widgets>
```

A Short Detour: Common Layout Widgets

Before we move on to Step 2 : *How to Orchestrate a Cockpit*,

let's take a look at some of the most common layout widgets

that you can choose for your custom cockpit

Border Layout Widget

- Most common layout used
- Contains 5 slots: north(*top*), south(*bottom*), center, west (*left*), and east(*right*) for placing other widgets
- Lots of available widget settings
  - enable/disable specific slots
  - set slot widths
  - allow/disallow slots to be collapsible
  - apply a border
  - ...
- This is the layout used for the Administrator Cockpit

With *north* and *south* slots **disabled**
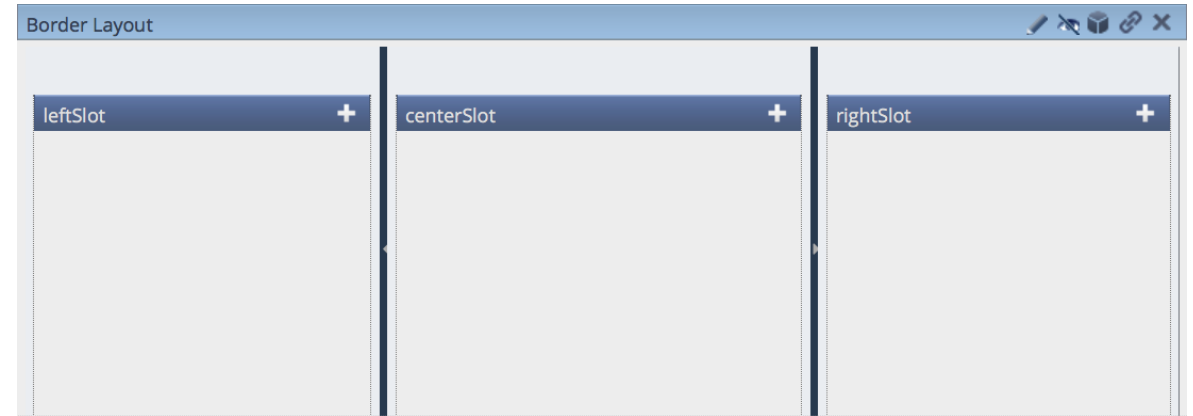
With *all* slots **enabled**

Collapsible Container Widget

- **2 or 3 slots that can be collapsed/expanded**
- **Settings**
  - enable/disable 3rd slot
  - title of each slot
  - height (of top & center slots) in %
  - allow locking
  - initial slot view state – collapsed/expanded
 - initial lock state of each slot – locked/unlocked

Portal Layout Widget

- No set layout! 1 large slot

- Add multiple widget instances to the slot

- Allows for dynamic layout by end user via drag and drop

- All widgets placed in the slot are automatically draggable
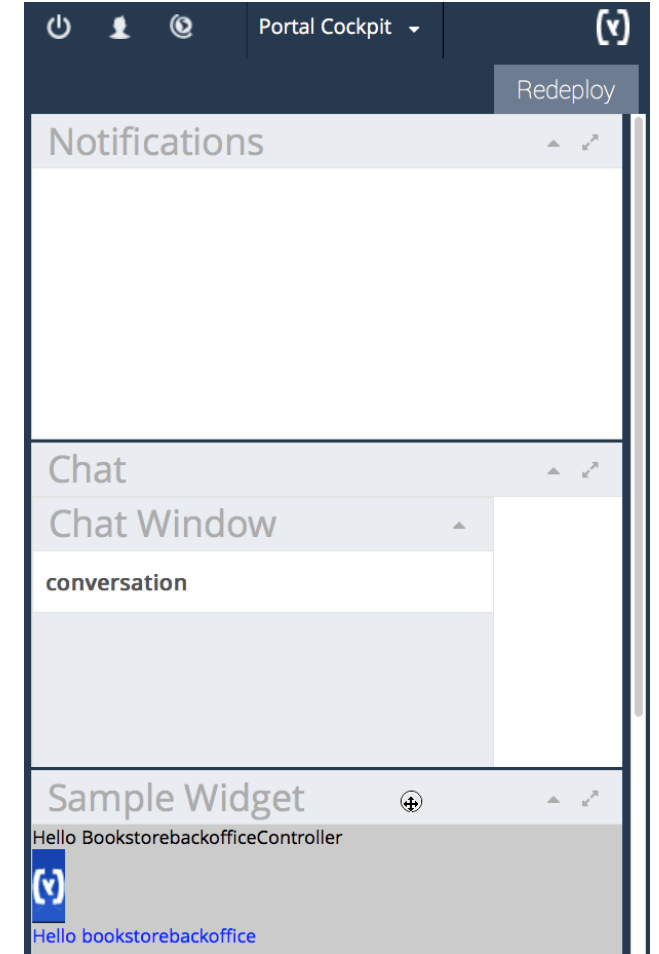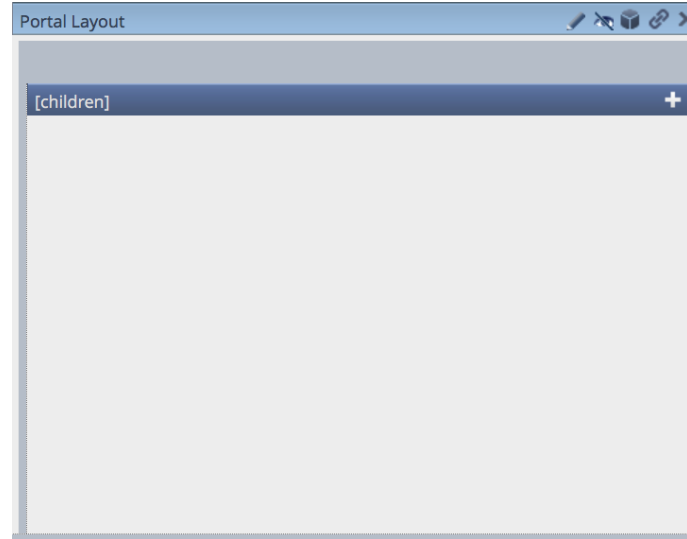
Split Layout Widget

- 1 slot for all widget instances

- All contained widget instances are separated by splitters

- End user can resize widget instances with splitter

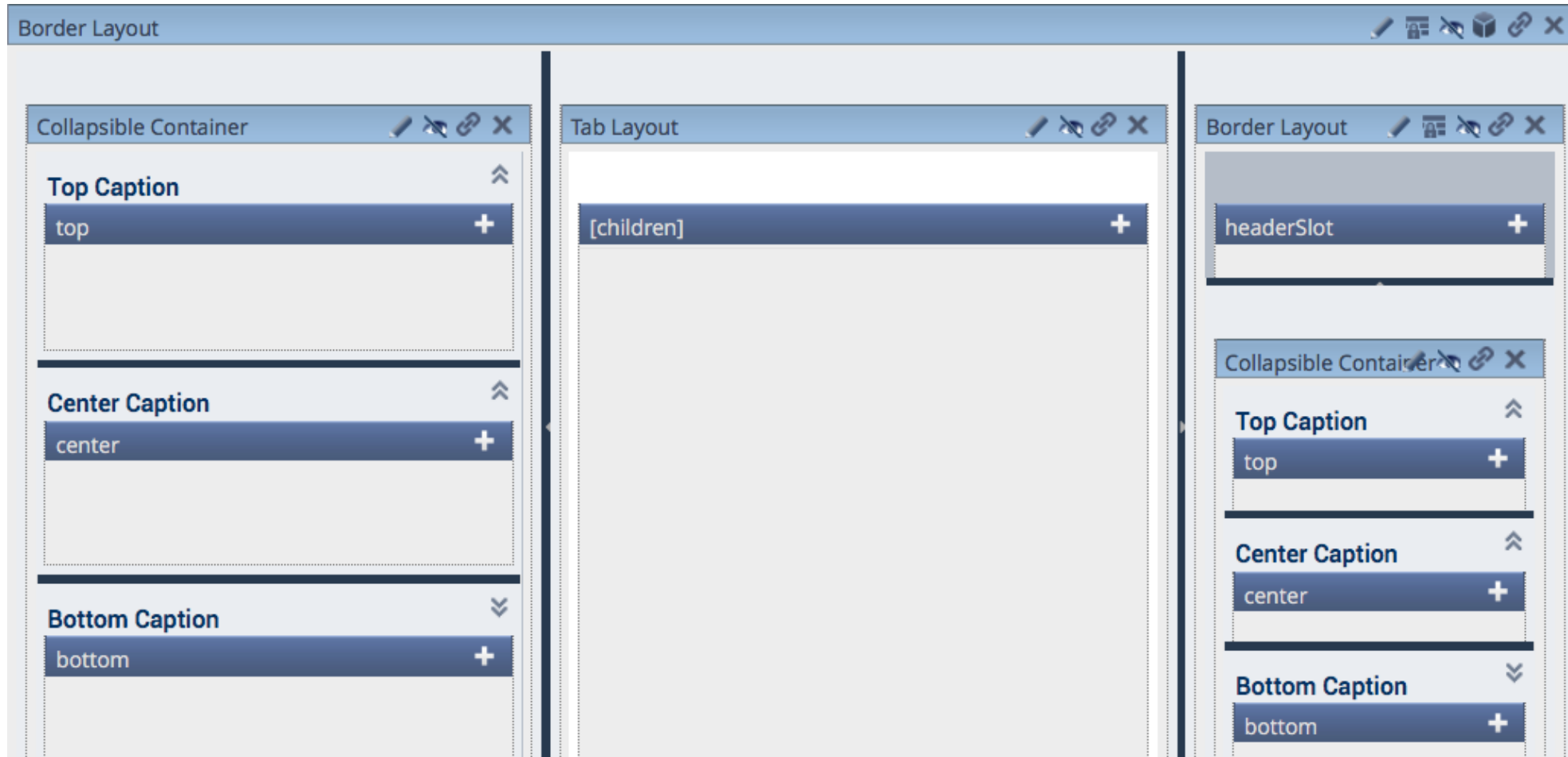- Settings
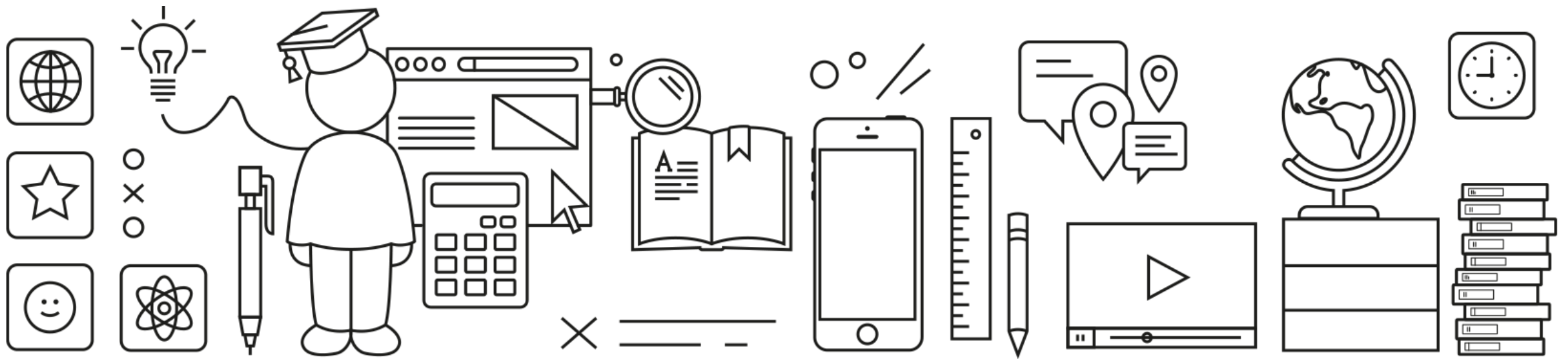 - orient (*v_split* or *h_split*)
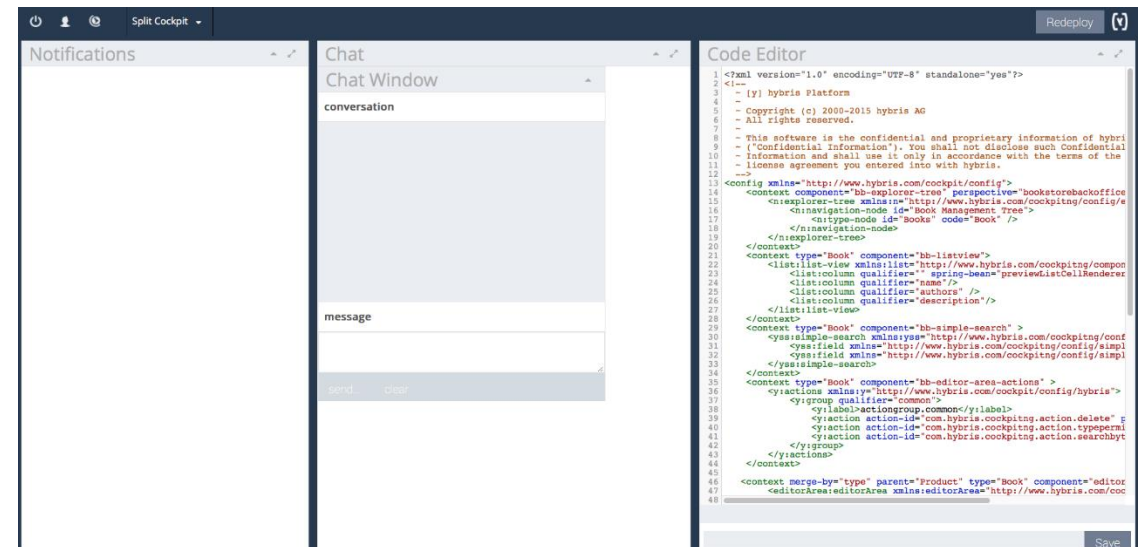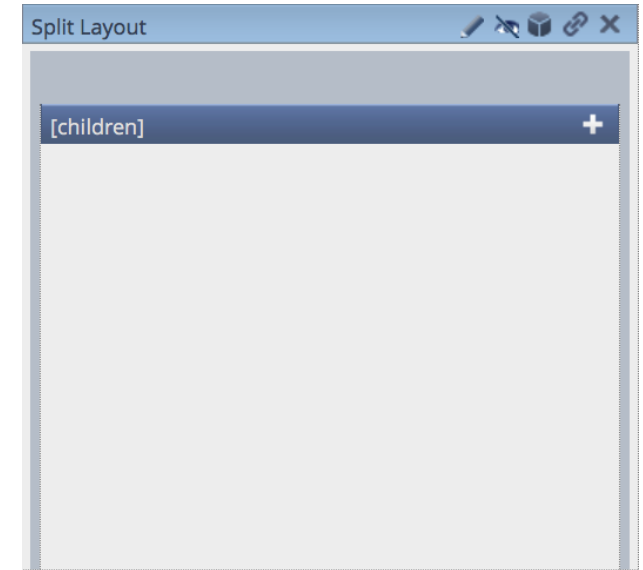
Tab Layout Widget

- 1 slot for all widget instances

- By default, renders contained widget instances as tabs

- Based on setting, can also render contained widgets as:
  – portal
  – list
  – h_split / v_split
  – …

- Settings
  – type

Step 1: Create a New Cockpit
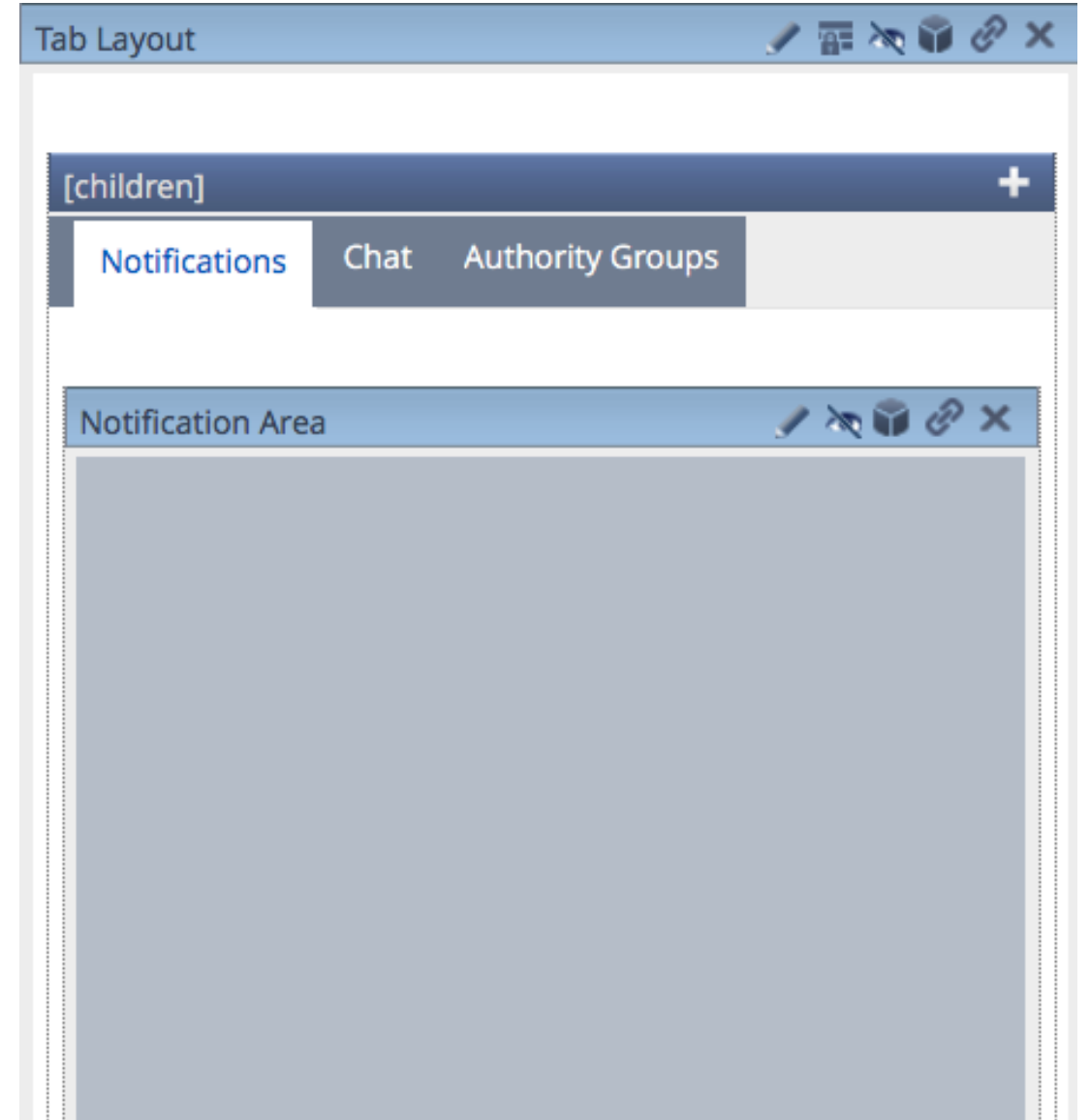
Complex Cockpit Layouts

You can create complex cockpit layouts by nesting layout widgets
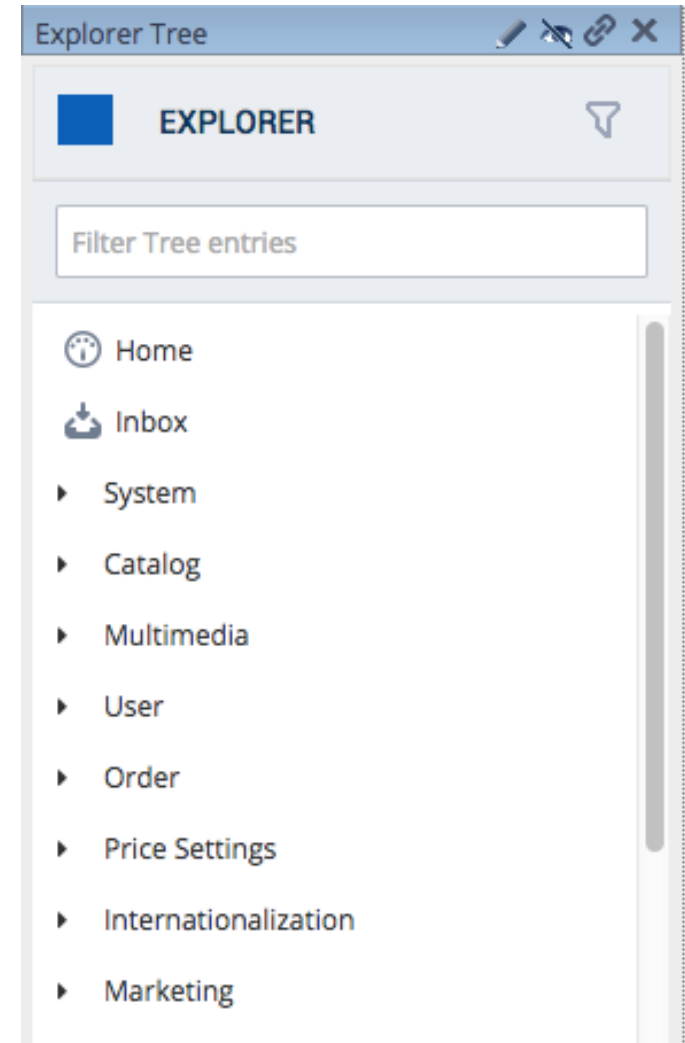
Step 2: Add Widget Instances

Step 2: Add Widget Instances

Border Layout

leftSlot

Choose widget ✕

« Categories

☐ Backoffice Perspective Chooser

☐ Explorer Tree

Shows a hierarchic...

☐ Next Item Selector

☐ Previous Item Selector

☐ Toolbar

Choose widget ✕

Select or filter: [　　　　] Upload   Refresh library

Groups (3)

Layout (9)

List (2)

Misc (1)

Navigation (5)

Permissions (1)

Preview (1)

repairprocessadapter (1)

Choose widget ✕

« Navigation

Explorer Tree
    com.hybris.cockpitng.widgets.common.explorertree

Description:

    Shows a hierarchical structure.

Inputs:
    nodeSelected (com.hybris.backoffice.navigation.NavigationNode)
    nodeIdSelected (com.hybris.backoffice.navigation.TreeNodeSelector)

Outputs:
    nodeSelected (com.hybris.backoffice.navigation.NavigationNode)
    dataSelected (java.lang.Object)

Advanced

Add & connect   Add & close

Step 2: Add Widget Instances - ctd

Now you should see an instance of the widget you chose replacing the empty slot

Step 2: Add Widget Instances - Settings

Configure settings for the new widget instance, such as

- Title (*tab display name*)
- Widget ID (*needs to be unique*)

Settings for com.hybris.cockpitng.widgets.common.explorertree ☒

| General | Localization | Virtual sockets | Access Restrictions | About |

Title: My Custom Explorer

Widget ID: my-custom-explorer

allowFilteringEntries: true ▾ ✕

explorerTreeConfigCtx: explorer-tree ✕

widgetStyleAttribute: ✕

widgetStyleClass: ✕

Add Setting

Close

Step 2: Add Widget Instances - Generated XML

Similar to the layout widget, you can display the merged `widgets.xml` file to view the XML that gets generated for the new widget instance

- Search for the widget's XML code by its unique Widget ID (`my-custom-explorer`)

- All widget instances added to a cockpit can be found inside the cockpit's `<widget ...> </widget>` tags

- To preserve the changes, copy them to your `custombackoffice-backoffice-widgets.xml`

custombackoffice-backoffice-widgets.xml

```xml
<widgets>
  ...
  <widget id="mycustom-cockpit" widgetDefinitionId="com.hybris.cockpitng.borderlayout"                              slotId="perspectives"
title="Custom Layout Cockpit" template="false">
          <widget id="my-custom-explorer"
                  widgetDefinitionId="com.hybris.cockpitng.widgets.common.explorertree"
                  slotId="leftSlot" title="My Custom Explorer" template="false">
                  ...
          </widget>
          ...
      </widget>
  ...
</widgets>
```
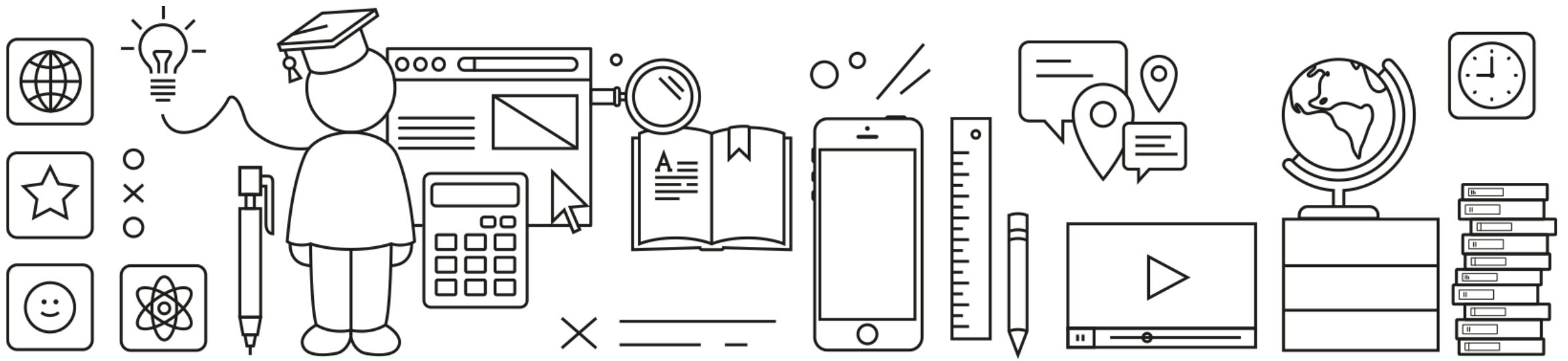
Step 3: Connect the Widget Instances

Communication in Backoffice

- Widgets use input/output sockets to communicate

- To establish a connection, you should connect one widget's output socket to another's input socket

- For a connection to be feasible, the *data types* of the input/output sockets on both ends must match or be compatible

Backoffice Business Application

Backoffice Widget

Outgoing message

Communication mapping

Receives messages

Backoffice Widget

The Backoffice Framework

Step 3: Connect the Widgets

Step 3: Connect the Widgets – (continued)

Choose sockets to connect   ✖

Source: com.hybris.cockpitng.advancedsearch

Target: com.hybris.cockpitng.widgets.common.advancedsearchengine

autosuggestionQuery   out

Type: com.hybris.cockpitng.search.data.AutosuggestionQueryData

searchData   in

Type:

com.hybris.backoffice.widgets.advancedsearch.impl.AdvancedSearchDat;

searchData   out

Type:

com.hybris.backoffice.widgets.advancedsearch.impl.AdvancedSearchDat;

reset   out

Type: java.util.Map

Source	out socket	Target	In socket	

Step 3: Connect the Widgets – Generated XML

In Backoffice orchestrator mode, you can display the merged `widgets.xml` file to view the XML that gets generated for the new connection

- Search for either source or target widget's XML code by the unique Widget IDs (`custombackoffice-advancedsearch` or `custombackoffice-advancedsearchengine`)

- All widget connections are put inside <widget-connection …> </widget-connection> tags

- To preserve the changes, copy them to your `custombackoffice-backoffice-widgets.xml`

Custombackoffice-backoffice-widgets.xml

```
…

<widget-connection sourceWidgetId="custombackoffice-advancedSearch" outputId="searchData"
          targetWidgetId="custombackoffice-advancedSearchEngine" inputId="searchData"/>
…
```
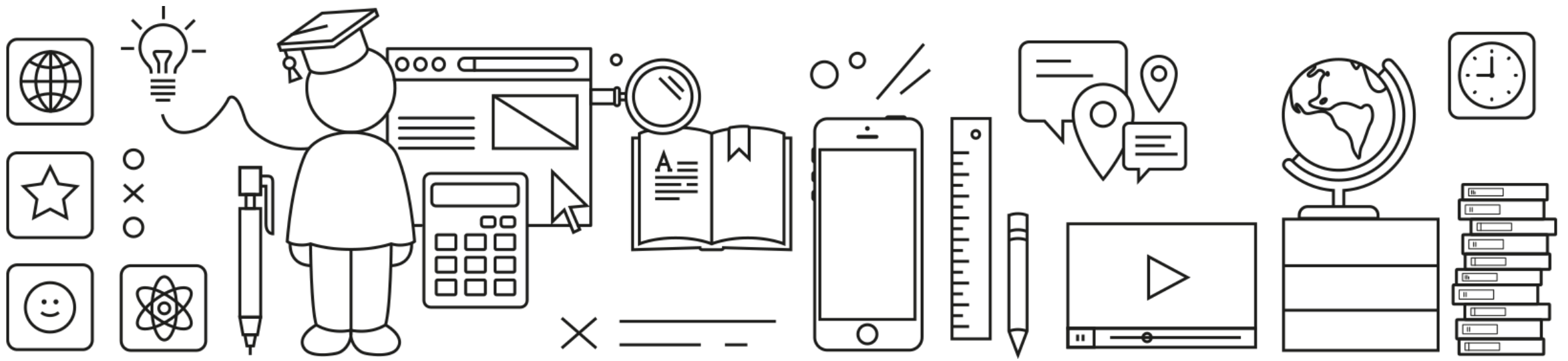
Orchestration Essentials

Widget Slots

- Components defined in the view (`.zul` file) of a widget to which other widgets can be assigned

- Enable a widget to have child widgets

- Used to create composed widgets

- Used by the layouts to have different slots in different parts of the screen

some-widget.zul

```
<widget ...>
    ...
    <widgetslot slotId="additionalArea" width="100%" height="100%" />

    ...
</widget>
```

Invisible Widgets

- Many widgets have a single invisible slot called `cockpitWidgetChildrenInvisible`

- Any widget placed on this slot will NOT be displayed

- Only accessible through its sockets

- Used for holding widgets that do *only* data manipulation

- (thus, they have no need for a view)
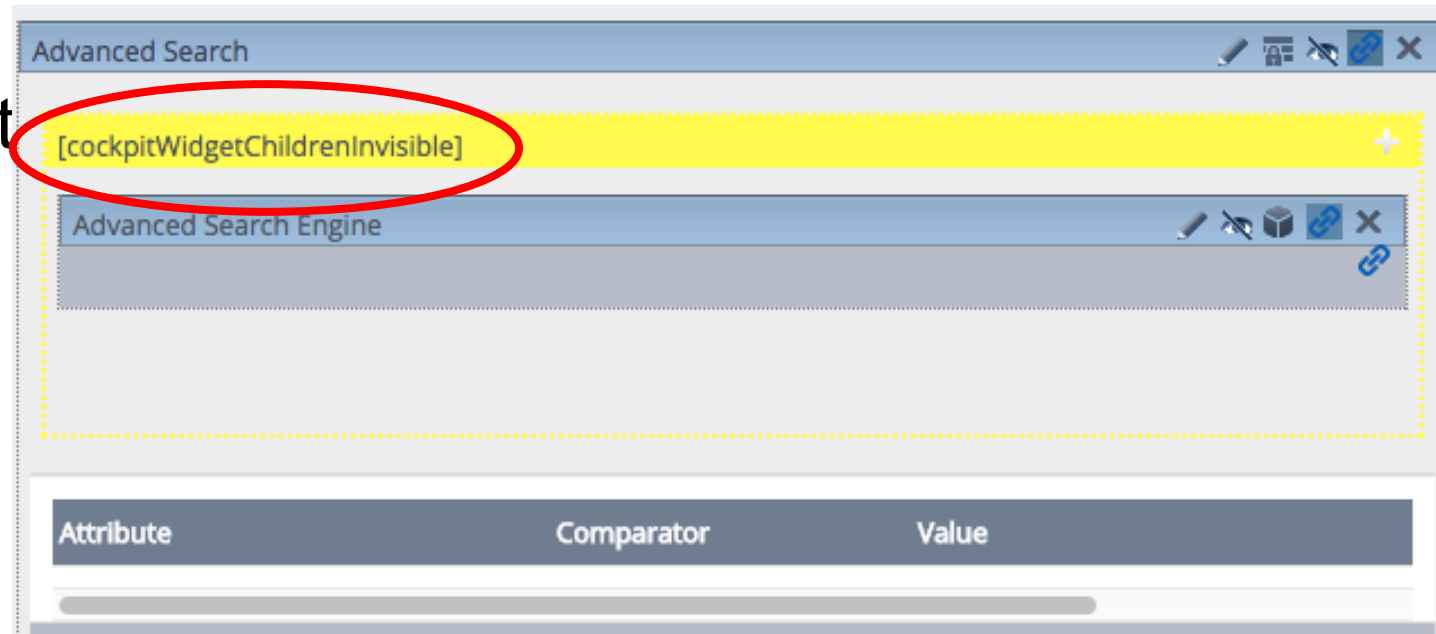
Application Orchestrator – Symbolic View

Symbolic view is a good place to see connections and dependencies
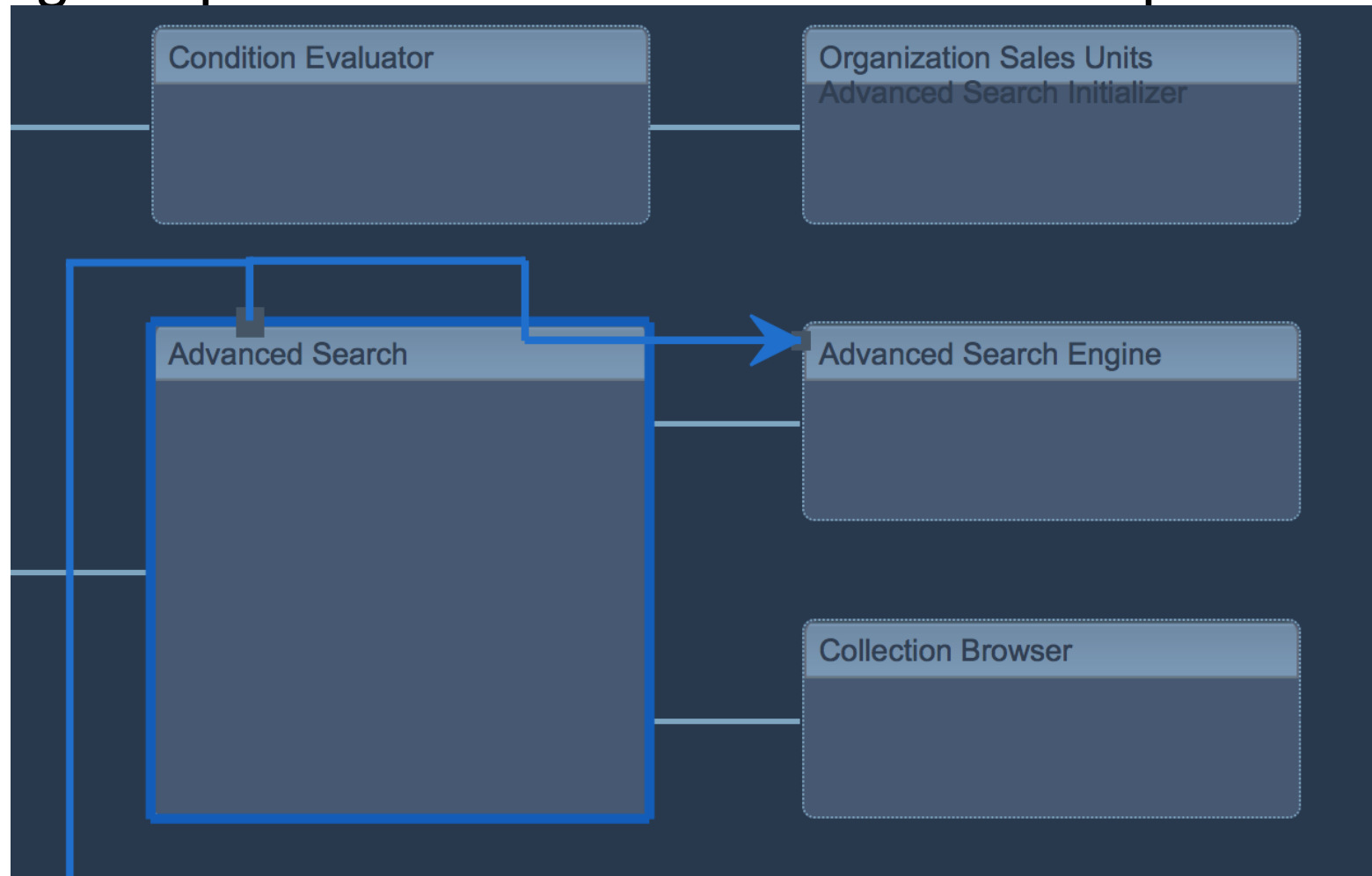
Remember These Widget Types from the Recap?

Advanced Search

Explorer Tree

Collection Browser

Editor Area

Advanced Search Engine Widget

- A helper widget that executes the search on behalf of the *Advanced Search* widget

- Used together with the Advanced Search widget

- Put in the invisible slot: `cockpitWidgetChildrenInvisible`
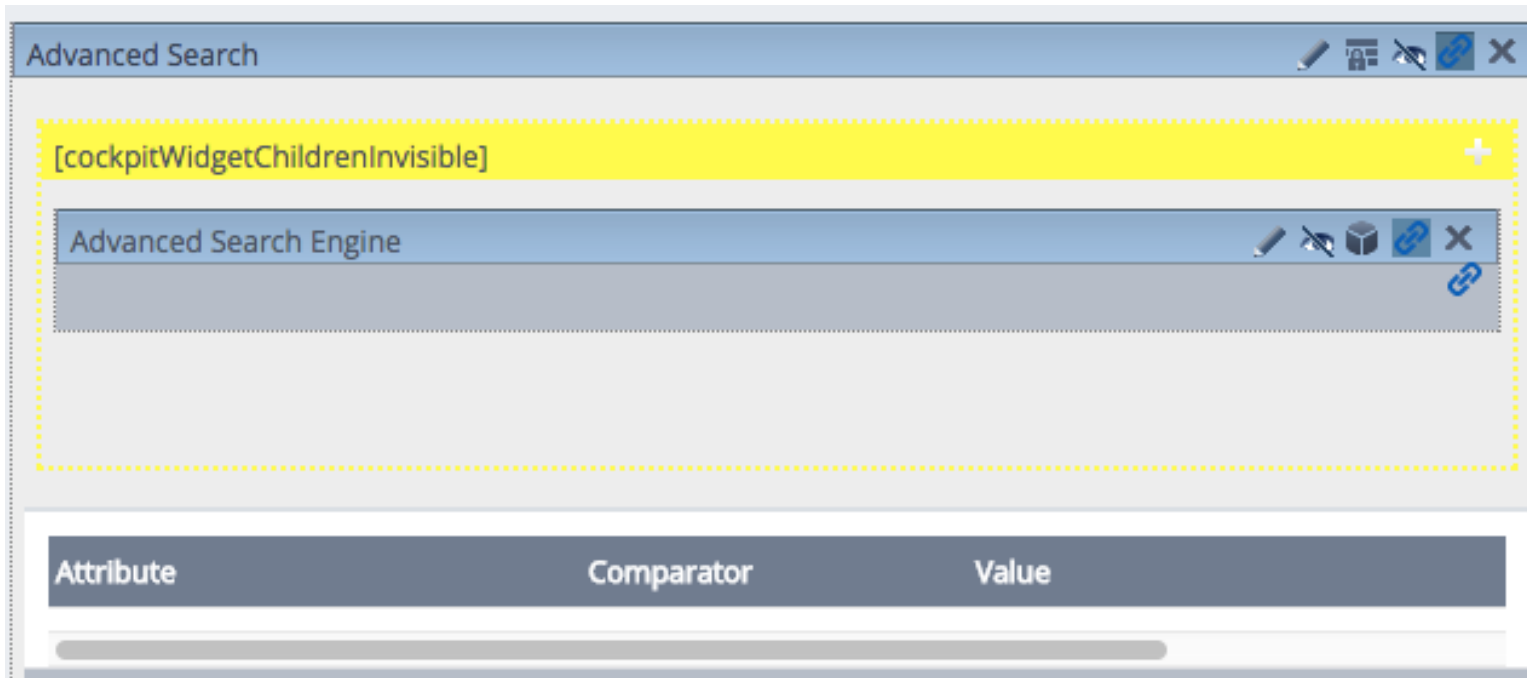
Backoffice Space Management
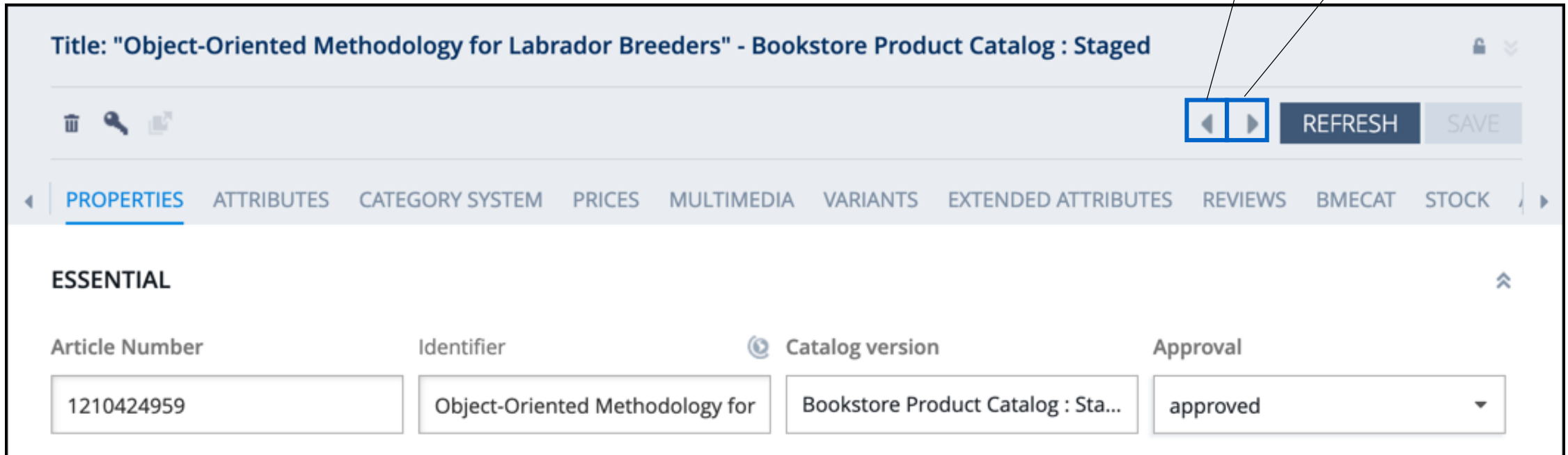
"The Backoffice Space Management widget enables you to automatically expand or collapse areas in the default Backoffice Application layout."

End-user actions such as selecting a node in the navigation tree, triggering a search query, and selecting an item in the list have an impact on the visibility of the following widgets:

- Advanced Search Widget
- Collection Browser Widget
- Editor Area Widget

Previous/Next Item Selector

The Previous/Next Item Selector widget enables you to get the previous/next item's attributes in the Editor Area displayed.

Demo

Exercise 4

Orchestrate a Cockpit

Exercise 4 – Orchestrate a Cockpit

Start making the Book Management cockpit your own!

The goal is to have a cockpit where the user can select the Book type node to view all Books in the system. The user can then click on a book to bring up its editor.

After Preparation

Three main widgets added:
- Explorer Tree
- Advanced Search
- Collection Browser

Enable Editing Books

- Add a new Editor Area for the books

- [OPTIONAL] Add navigation buttons to it

- Persist your changes in
  `bookstorecustombackoffice-backoffice-widgets.xml`

Exercise 4 – The End Goal

Thank you.