Exercise 4

Orchestrate a cockpit

INTRODUCTION

In this exercise, you will start working with a more populated cockpit. In the preparation script, some basic components are added to your cockpit. You will then add a new component to the cockpit to complete it… well, for now!

The preparation script will create new instances of the Explorer Tree, Advanced Search, Advanced Search Engine, Collection Browser, and Backoffice Space Management widgets. It also establishes the necessary socket connections between them. These widgets are set up so that you can

- navigate to books (Explorer Tree)
- search for books (Advanced Search and Advance Search Engine)
- see the result of the search (Collection Browser)
- expand/collapse areas to make better use of space in the cockpit (Backoffice Space Management)

The piece that you're going to add enables you to edit a book's attributes. For that, you will add an instance of the Editor Area widget.

What you will learn

This exercise focuses on using the Orchestrator for creating new widget instances and connecting them together. You will create a new Editor Area widget instance and connect it to the selected socket of the Collection Browser widget instance.

You will also see how you can preserve the changes you made in Application Orchestrator by editing the *widget layout application configuration* of your custom Backoffice extension (i.e., `bookstorecustombackoffice-backoffice-widgets.xml`).

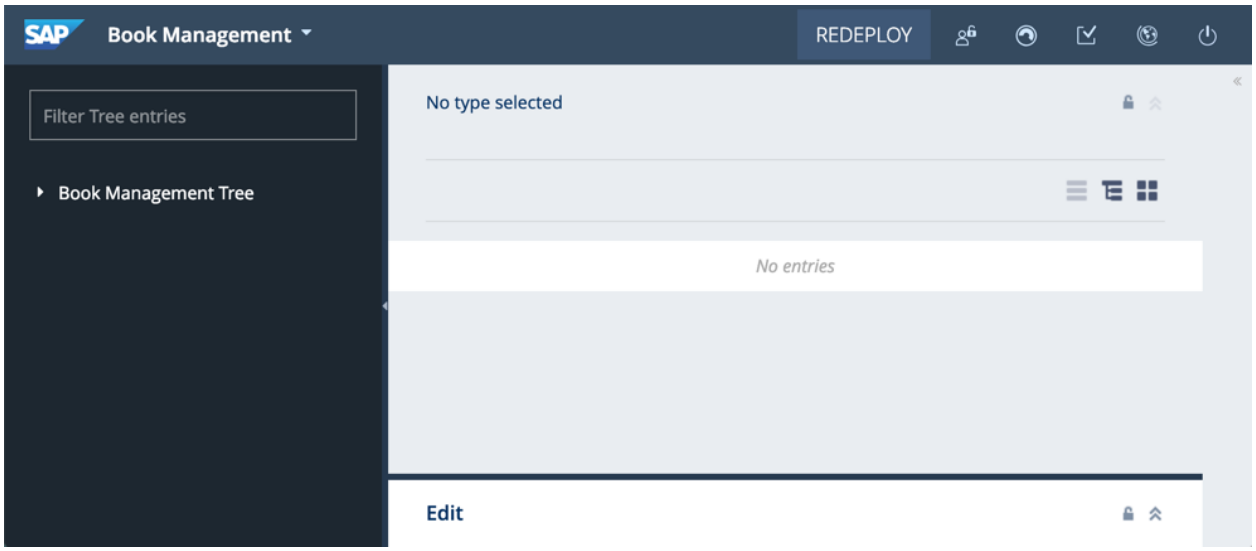
What you will need before starting

- Complete previous exercise
- Run the preparation script of exercise 4

INSTRUCTIONS

1. After the preparation script completes, start the commerce suite, or leave it running if it already is. The preparation script copied (and overwrote) the `bookstorecustombackoffice-backoffice-config.xml` and `bookstorecustombackoffice-backoffice-widgets.xml` file to your *bookstorecustombackoffice* extension's resources folder. Open the Backoffice application at https://bookstore:9002/backoffice/ (if you're already logged-in to Backoffice, log out and log back in to trigger a reset on `widgets.xml` and `cockpit-config.xml`).

   Go to the Book Management cockpit. It should look like the image below:



   NOTE: The preparation script for this exercise added an icon graphics file to your **bookstorecustombackoffice** extension under `/bookstorecustombackoffice/backoffice/resources/cng/images/`. In some versions of Backoffice (prior to the current version in use here), at runtime, this icon would be visible next to the ***Book Management*** option in the cockpit chooser dropdown menu.

   If you want to learn more about how to setup/change a cockpit's icon, have a look at this SAP help page:

   Adding Custom Icon for Backoffice Cockpit

2. Expand the "Book Management Tree" node and click on the ***Book*** child node. This will populate the main area of your cockpit with a list of all the ***Book*** items in the commerce suite. If you select a book, the ***Edit*** section at the bottom of the main area will expand to almost fill the main area but it will display no data.

   In this exercise, we want to change that, so the selected book's data populates this section.

3. Open Application Orchestrator by pressing `F4` (`fn+F4` in Mac OS).

4. In Application Orchestrator, within your *Book Management* cockpit (here, labeled as a *perspective*), locate your *Book Management* cockpit's only ***Collapsible Container*** widget (in ***centerSlot*** of ***Border Layout***). Within this ***Collapsible Container*** widget, locate the slot named ***bottom***. We now want to add a new ***Editor Area*** widget instance to this slot as follows…

   Pressing the **+** icon of the aforementioned ***bottom*** slot causes a wizard window to pop-up to let you choose what type of widget instance you would like to create and place into this slot. Within the possible widget types, filter the list by typing ***Editor Area*** into the pop-up window's search field – you should find the ***Editor Area*** widget within the ***Data Manipulation*** group. In the final step of the wizard, click on **Add & connect**. A new window pops up in which you can define the socket connections to/from this new widget instance. We will define these connections in the next few steps…

5. Remember from the slides the way widget communication works in the Backoffice Framework: a widget instance sends Java objects (of a specified type) through one of its output sockets to an input socket of another widget instance. These two sockets must essentially be "hard-wired" to each other via a *widget connection* entry in the widget-layout configuration. We *usually* define these connections via the Application Orchestrator, but it can also be done by hand-editing the *-backoffice-widgets.xml file manually.

   Back to our immediate task… The way that a Collection Browser widget is designed to work is that every time one of its list entries is clicked-on by the user, the Collection Browser widget sends out a copy of the selected object (e.g., a `BookModel` object) through its ***selectedItem*** output socket. We want this selected item to be displayed by our new Editor Area widget, so we must send this object to the Editor Area widget's input socket named ***inputObject*** (because Editor Area widgets are designed to render each object as it arrives at its ***inputObject*** input socket). We must define a *widget connection* to hard-wire the Collection Browser's output socket to the Editor area's input socket.

   So do the following: in the ***Choose sockets to connect*** pop-up window, you must first select the ***inputObject*** socket from the left panel. Then, on the right panel, expand the ***type match*** node and look for ***bookstorecustombackoffice-result-list***. (Hint: use a browser-page search, e.g., via Ctrl-f, to automate this search). Click on the ***bookstorecustombackoffice-result-list*** node to expand it, and then in the resulting subtree, click on ***selectedItem*** (i.e., NOT the plural ***selectedItems***). A new connection entry should appear in the bottom area of the popup window.

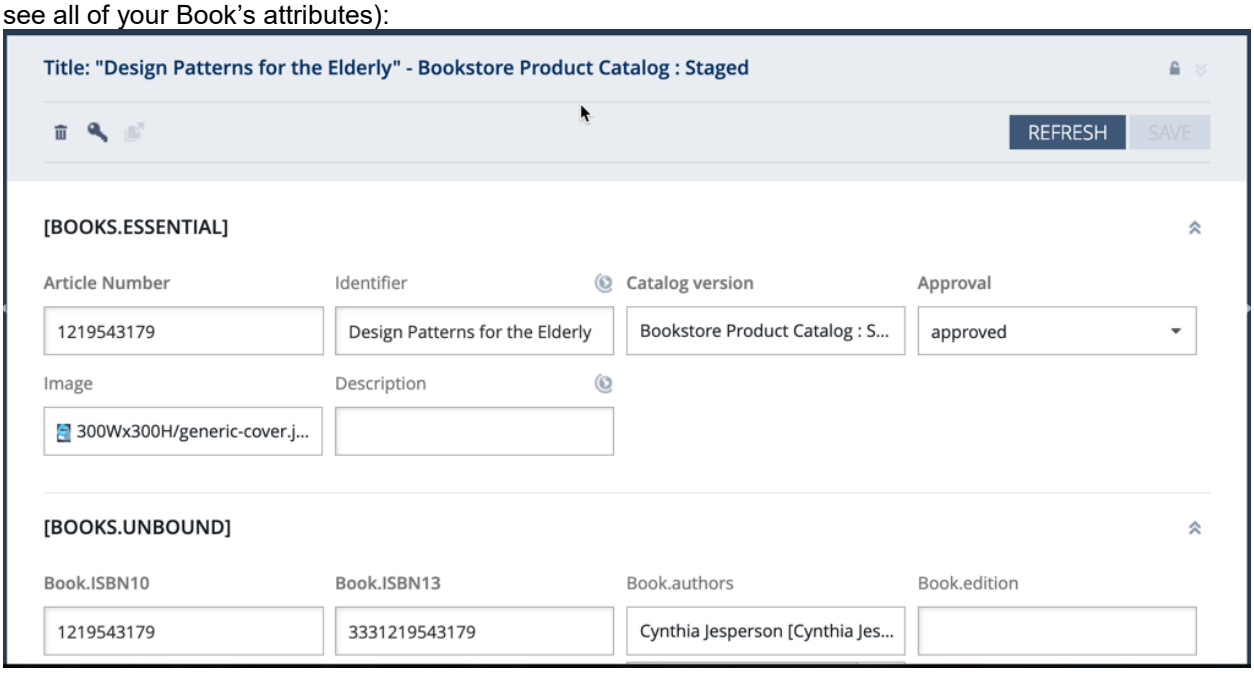
   You can now close this wizard window.

6. Let's give the newly instantiated Editor Area widget a more meaningful *Widget ID*. Find the widget and open its settings by clicking on the pencil icon. Then, give it a *Widget ID* of: `bookstorecustombackoffice-editor-area.` While still in *settings*, change the value of *editorAreaConfigCtx* from *editor-area* (the default value specified in this widget type's definition) to *bcb-editor-area* (the *bcb-* prefix is an abbreviation for **b**ookstore**c**ustom**b**ackoffice).

   > NOTE: If you had left the original value, your *Editor Area* widget instance would have taken on the exact same configuration as the Editor Area widget instance in the Backoffice Administration Cockpit. While it's nice that this configuration organizes product attributes into a whole bunch of tabs, all Book-specific attributes are relegated to the catch-all ADMINISTRATION tab (because the *editor-area* configuration is only capable of organizing attributes of *Product*, i.e., the parent type of *Book*). Of course, we want our *Book Administration* cockpit to know all about the *Book* type, so we have provided you with a custom *bcb-editor-area* configuration. We will cover widget configuration in detail in a future module.

7. Exit *Application Orchestrator* by pressing `F4` again.

8. Try selecting a book from the list of books. This time you should be able to see the custom-configured Editor Area widget, just like the following image (scroll down within your Editor Area to see all of your Book's attributes):



9. Now your *Editor Area* widget instance is configured to display *Book* attributes without the use of tabs, but having two *sections* whose labels appear as all-upper-case *localization keys* enclosed in square brackets. These localization keys were specified (actually, in lower-case) inside the *bcb-editor-area* configuration we mentioned earlier. Let's now provide values for these keys. Open the file `bookstorecustombackoffice/resources/bookstorecustombackoffice-backoffice-labels/labels_en.properties`.

   > The `bookstorecustombackoffice-backoffice-labels` folder is a Java *resource bundle* and contains a set of files with the same *base name* (in this case, `labels.properties`) appended by a *Locale*. Hence, English-specific labels reside in `labels_en.properties`, Spanish-(language)-specific labels reside in `labels_es.properties` (Español), etc.

   Add the following two name/value pair entries:

   books.essential=Essential Book Attributes
   books.unbound=Misc. Attributes (unassigned to a section or tab)

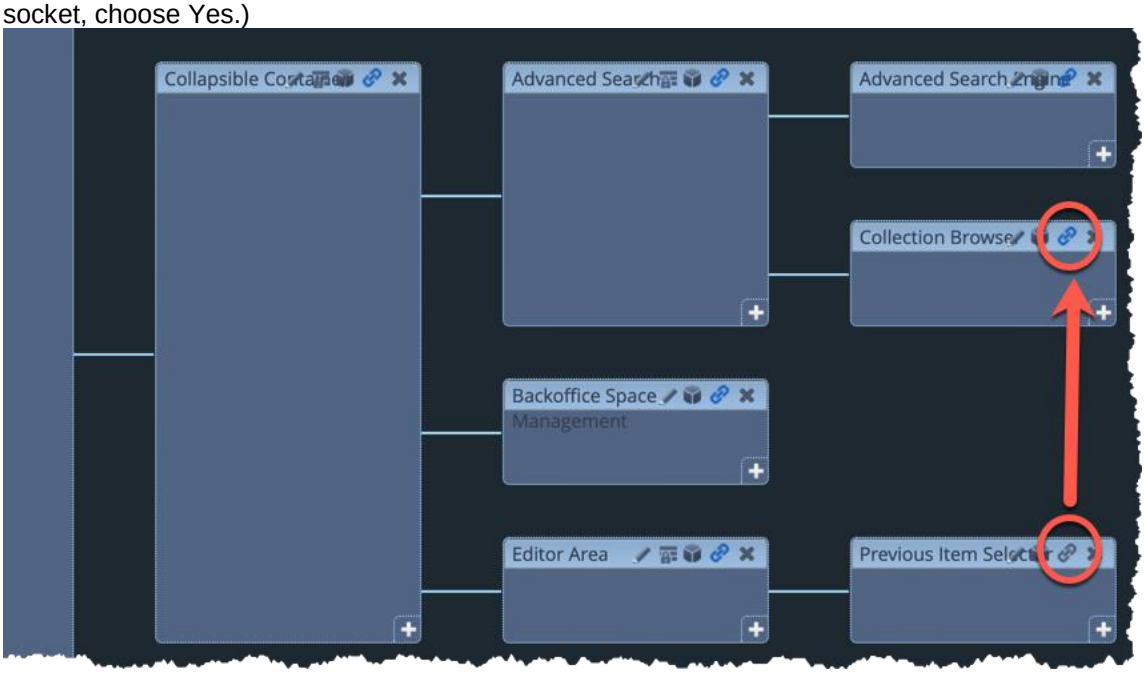
   You won't see the effects of your new `.properties` file entries until you restart the Commerce Platform, which we will save for later.

10. Let's add the Next Item Selector and Previous Item Selector widgets to the Editor Area to enable navigation through the list of books from within the Editor Area. Because of a limitation in the way header-area slots are displayed, it is better to perform this task in the Symbolic Widgets view of Application Orchestrator.

    Enter the Application Orchestrator and switch to the Symbolic Widgets view. You need the Editor Area widget instance with the ID, `bookstorecustombackoffice-editor-area`, but since instance-specific Widget IDs are not displayed on the browser page, you can only (Ctrl-F) search the text *Editor Area* (i.e., the widget's type, which ***is*** displayed) and, one at a time, click on each *Editor Area* widget's pencil icon to view its instance ID. There are only 8 occurrences of this text, so it's not so bad – for me, it was the fourth occurrence.

On this widget instance, to add a child widget to one of its slots, press the **+** button; a *Chose slot id* window should pop up. Choose `previousItemSelectorSlot`. In the new *Chose widget* wizard popup window, choose the *Previous Item Selector* widget to instantiate in this slot of the Editor Area widget. Rather than clicking on *ADD & CONNECT* and having to peruse sockets from every widget in the application, we will follow a more targeted approach, so click on *ADD & CLOSE*.

Using the technique where you click-and-drag the chain-link icon of the source widget to the chain-link icon of the target widget, connect the *Previous Item Selector* widget's output socket (named `previousItemSelectorInvocation`) to the identically named input socket of your cockpit's Collection Browser widget instance (if you are asked to auto-set the <T> type on a socket, choose Yes.)

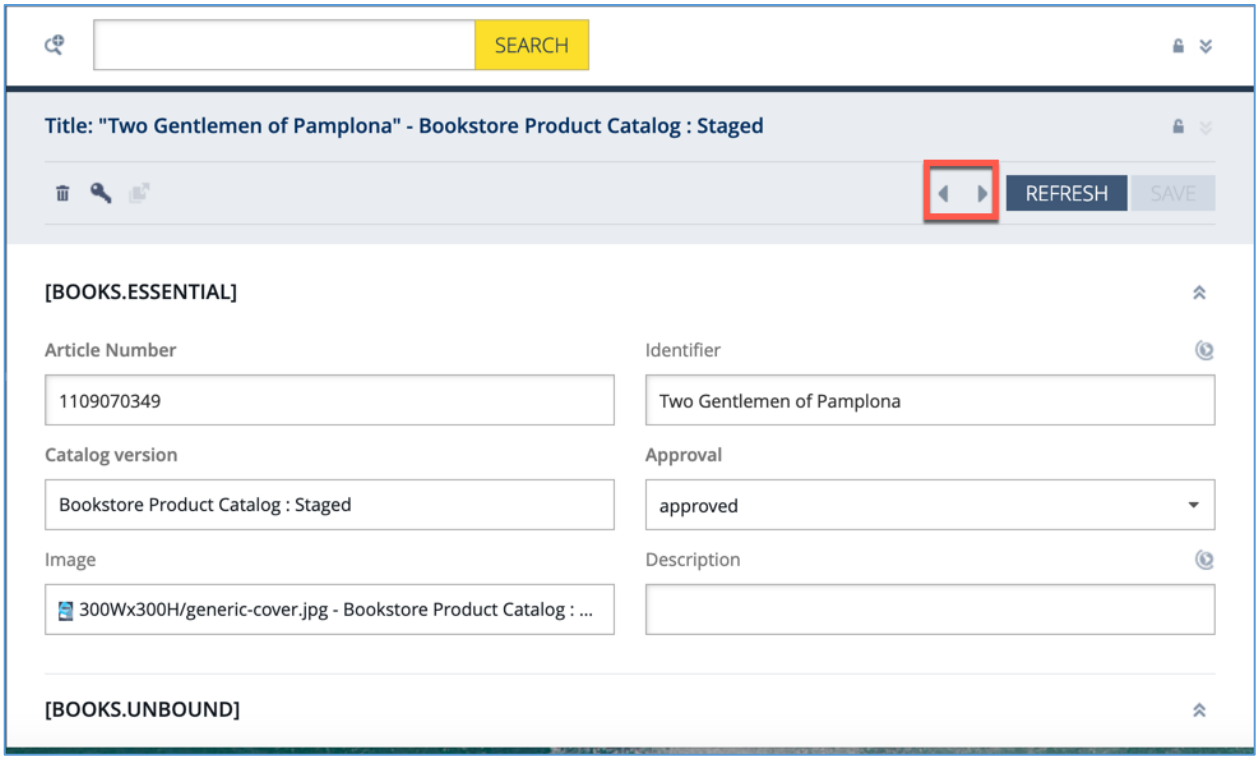


Now that you have this widget connection, if a user clicks on the Previous Item Navigation Arrow button in the Editor Area, it will cause the Collection Browser to select the next Book item in its list and send a matching `BookModel` object to the Collection Browser's ***selectedItem*** output socket.

In order for the Previous Item Navigation Arrow to know whether there IS or ISN'T a previous item in the list (so that it knows whether or not it should be greyed-out), you will also need to connect the `previousItemSelectorContext` output socket of the *Collection Browser* widget instance back to the identically named input socket of the *Previous Item Selector* widget instance (if you are asked to auto-set the <T> type on a socket, choose Yes.)

Do a similar set of tasks with `nextItemSelectorSlot` and the *Next Item Selector* widget.

Exit the Application Orchestrator mode, click on a book from the list to force Backoffice to re-render the editor area and its navigation arrows. You should be able to see a set of active *previous* and *next* arrow icons in the editor area next to the Refresh button, as it is shown in the image below. Clicking on the buttons will cause a different book to display in the Editor Area. (Remember, your label values won't apply until you do a restart).



11. So far, most changes made are only available until a reset of the configuration (e.g., logout and re-login, according to our current configuration), and then all changes are gone. To preserve these changes, we should place them in `bookstorecustombackoffice-backoffice-widgets.xml`.

    The required XML has already been placed in the file in the form of two comment-out blocks of XML (placed in this file for you by the preparation script): one for adding the new widget instances to existing widgets' slots, and the other for establishing the widget-instance connections.

    Via your Eclipse IDE, open `bookstorecustombackoffice/resources/bookstorecustombackoffice-backoffice-widgets.xml` and uncomment the two commented-out blocks.

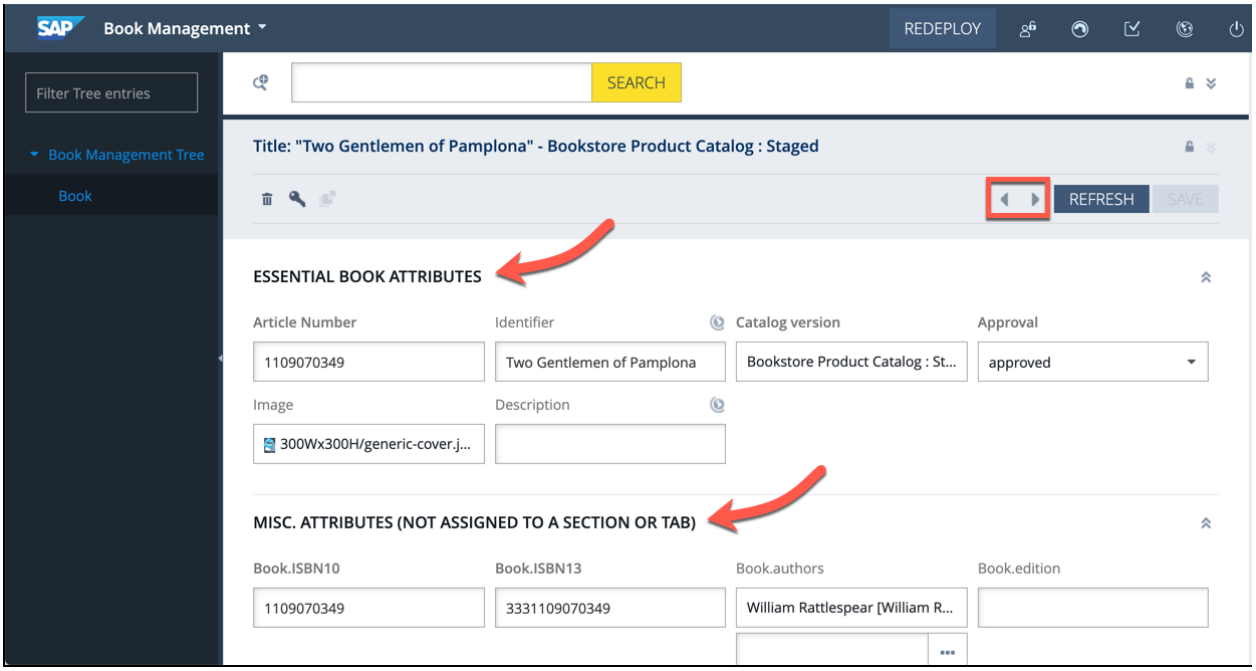
    Don't forget to save the file before closing it.

    Remember, the usual way to accomplish this task would have been to open `widgets.xml` from the Application Orchestrator's *stack menu*, copying its contents into a stand-alone text editor, locating the new `<widget>` and `<widget-connection>` entries, and copying-and-pasting them into the appropriate places in your extension's `*-backoffice-widgets.xml` file manually.

12. [OPTIONAL] If you really want to verify that you've not broken anything by editing (and saving) your `*-backoffice-widgets.xml` file, and if you want to verify that your label entries worked, you will need to perform a restart followed by a "widets.xml reset everything", BUT before you do

that, read on… A restart will force you to log back in and trigger a reset anyway, so I like to do the following to keep everything nice and orderly: log out of Backoffice, stop and restart the Commerce Suite, then log back in (which will trigger a configuration reset, saving you from having to do that manually), and re-verify all of the above functionality, i.e., that the Editor Area widget and (optionally) its *previous* and *next* navigation arrows work properly and that your label values are now visible in your Book Management cockpit's *Editor Area* widget instance.

If you've done everything correctly, your cockpit should look like the screenshot below:



Congratulations – you are now ready to proceed to the next exercise!