Practices for Lesson 4: Configure the Order and Item Creation and Edit Order Pages

Overview

In these practices, you will:

- Use Advanced Create and Edit Page Template to add the Create Order/Item functionality
- Create the Edit Orders page using Simple Create Edit Page Template

Note: These practices focus on configuring the Page Templates and do not include implementing the create/edit back-end integration and functionality.

Practice 4-1: Create the New Order and Line Items Page

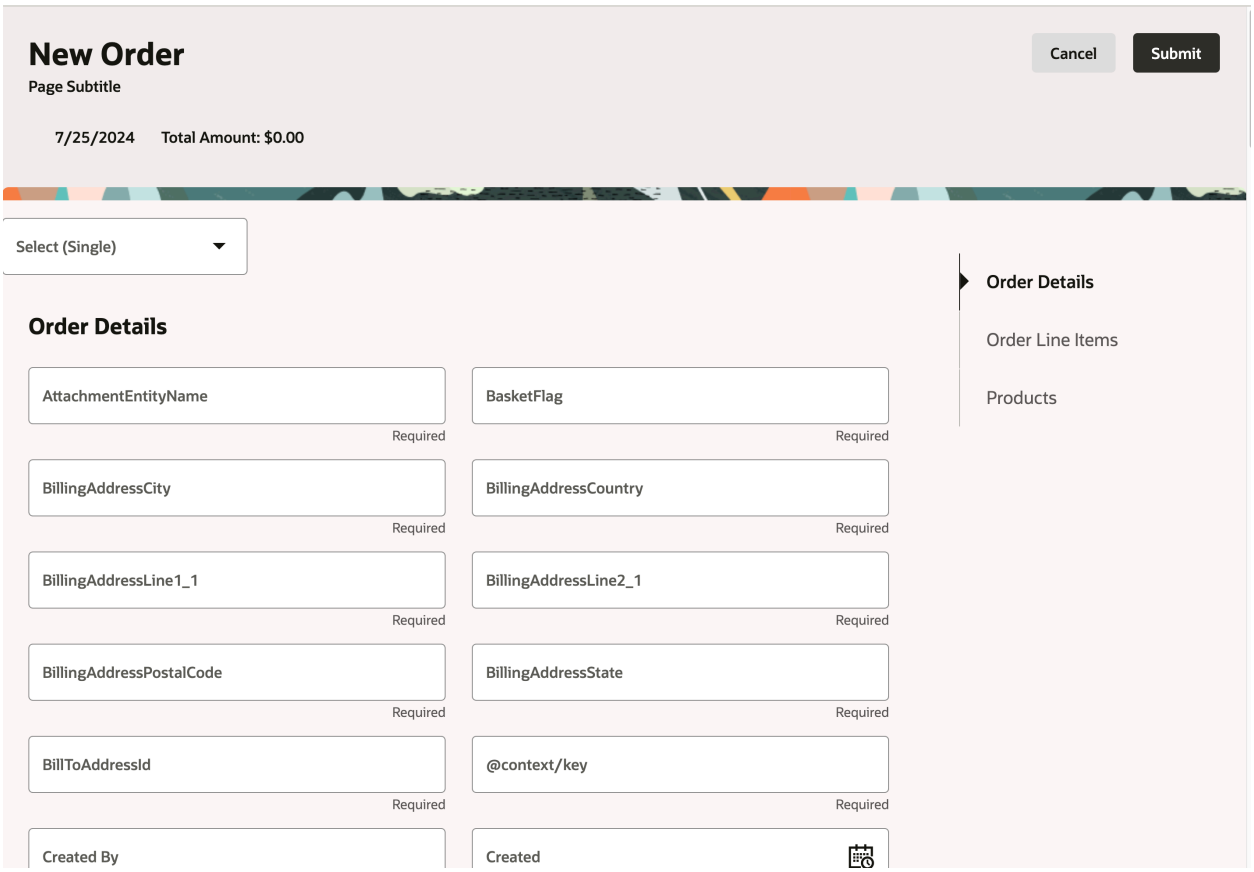
Overview

In this practice, you configure pages to add a new order, add line items, and add products for the line items.

You work with Sections and the Vertical Anchor to navigate around the sections.

You will focus on the layout of the page and working with new components. Some functionality, like inserting rows and saving to the database using the REST service connection and API, are not implemented because these are not relevant for learning the Redwood components and are outside the scope of this course.

Note: Inserting a new row in a table is somewhat complex and is performed through coding. You can see how to do this in the Visual Builder Cookbook, an excellent resource for learning how to develop in Visual Builder: https://vbcookbook.oracle.com/?page=shell&shell=table-add-row

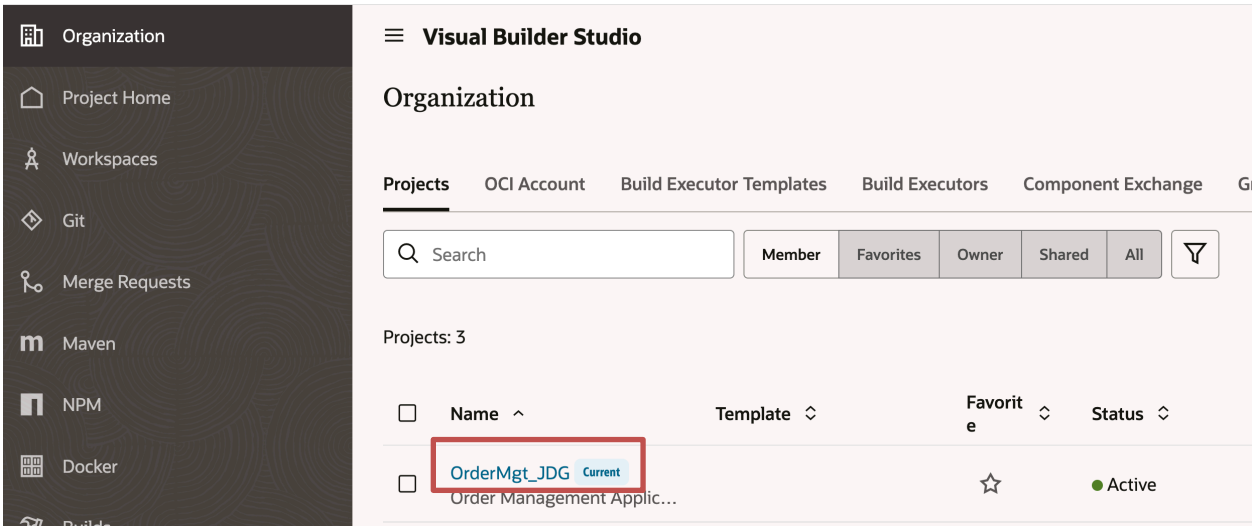


Assumptions

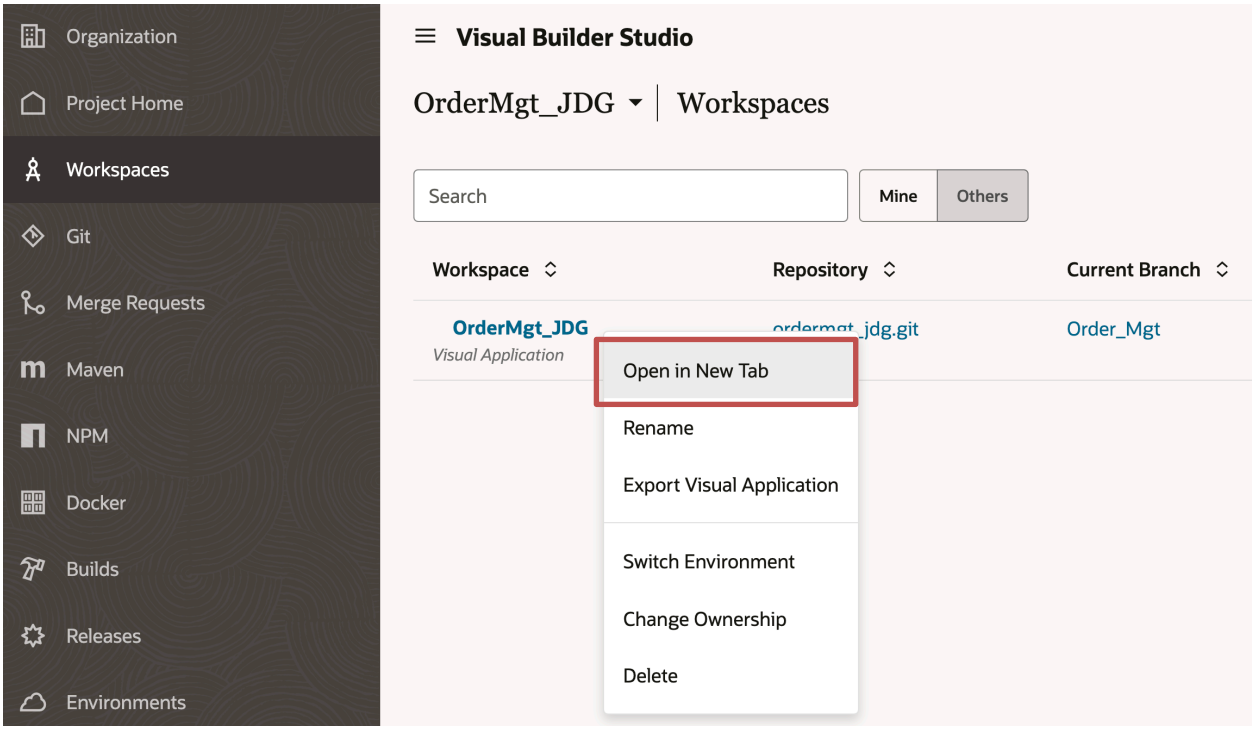
You have successfully completed Practice 3-1.

Tasks

1. Log in to **Visual Builder Studio** and select your project name on the **Projects** tab. Your project name will be **OrderMgt_< your initials >**, where **< your initials >** corresponds to your initials.

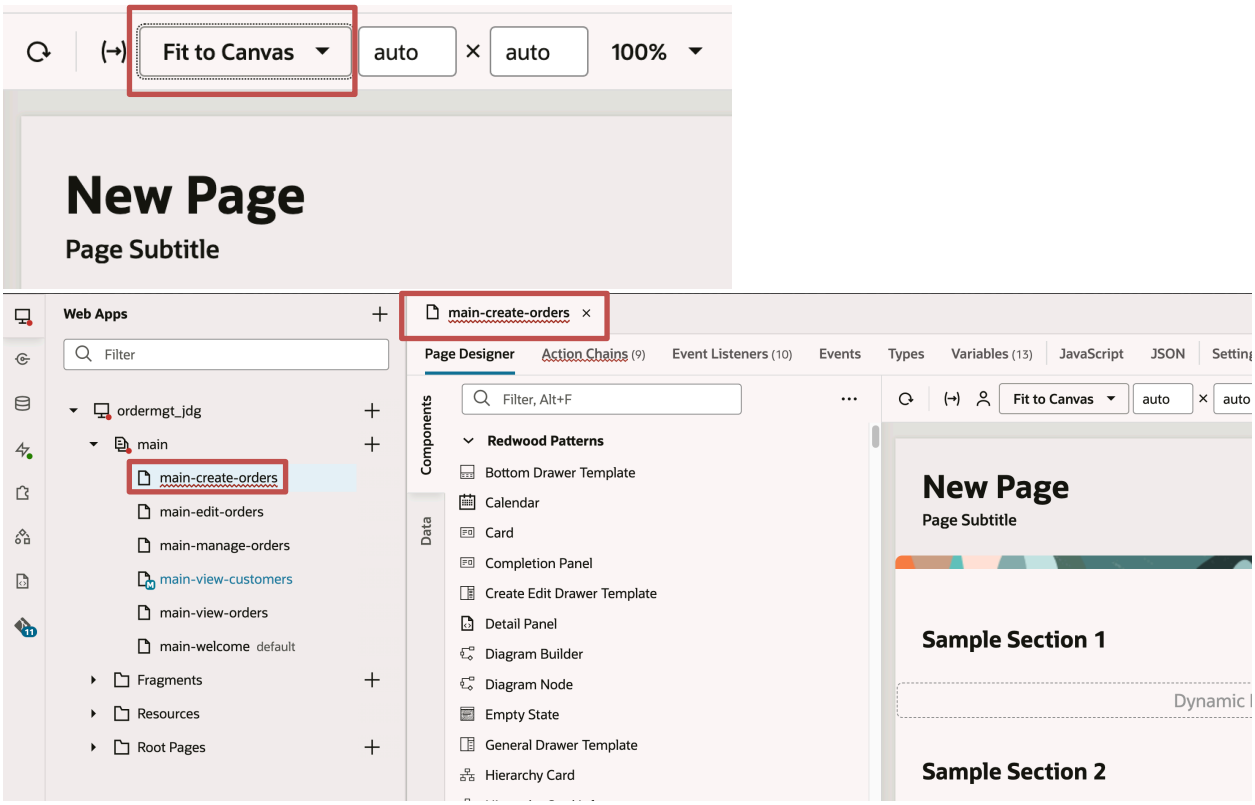


2. The Project Home page opens. Right-click the **OrderMgt_<your initials>** workspace and select **Open in New Tab**.

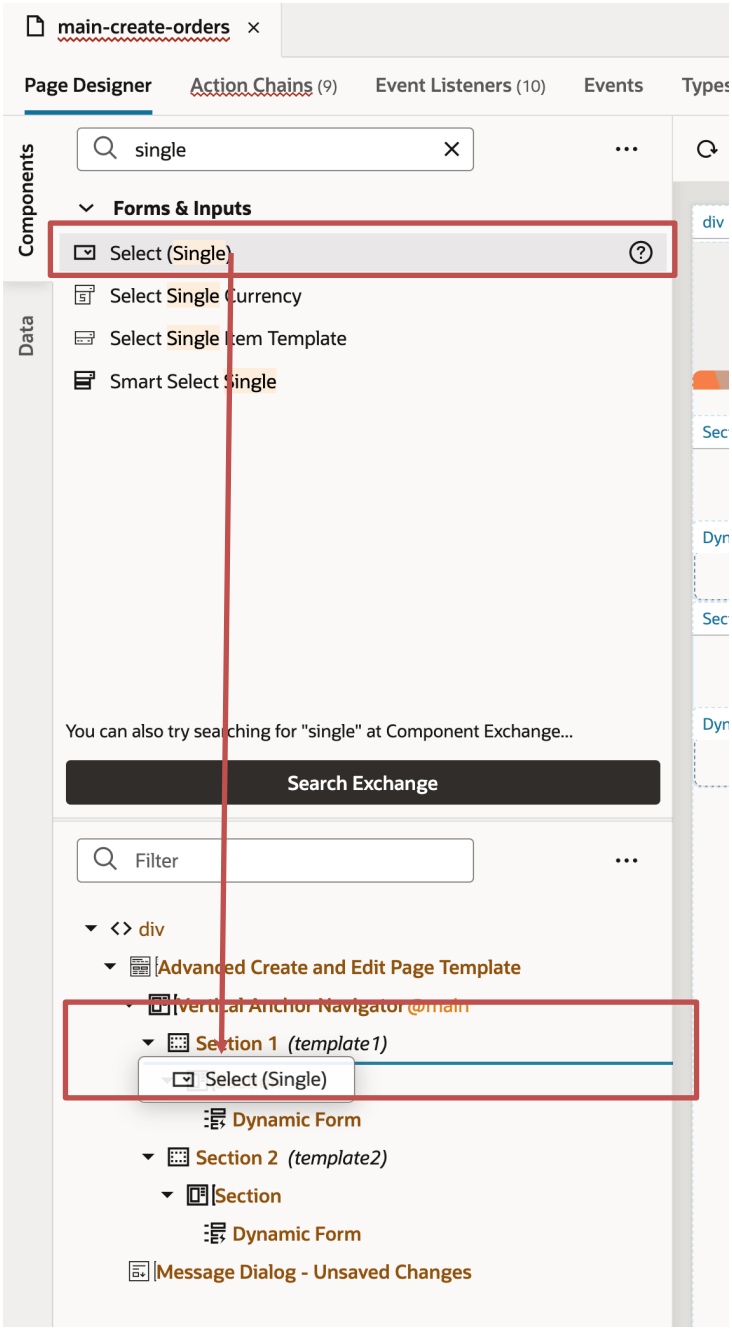


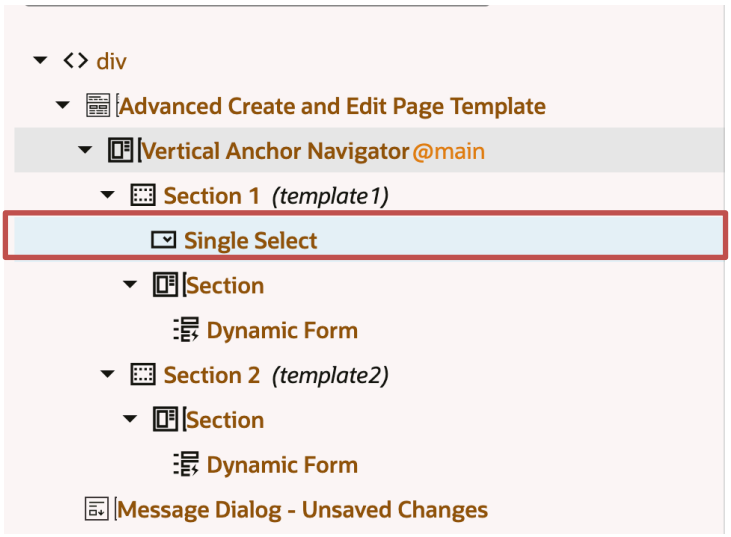
3. Close the **Getting Started** tab if it's open.
4. Select the **Web Apps** tab in the Navigator. Expand **ordermgt_<your initials>**. Expand **main**. Select the **main-create-orders** page. It opens in the Page Designer.

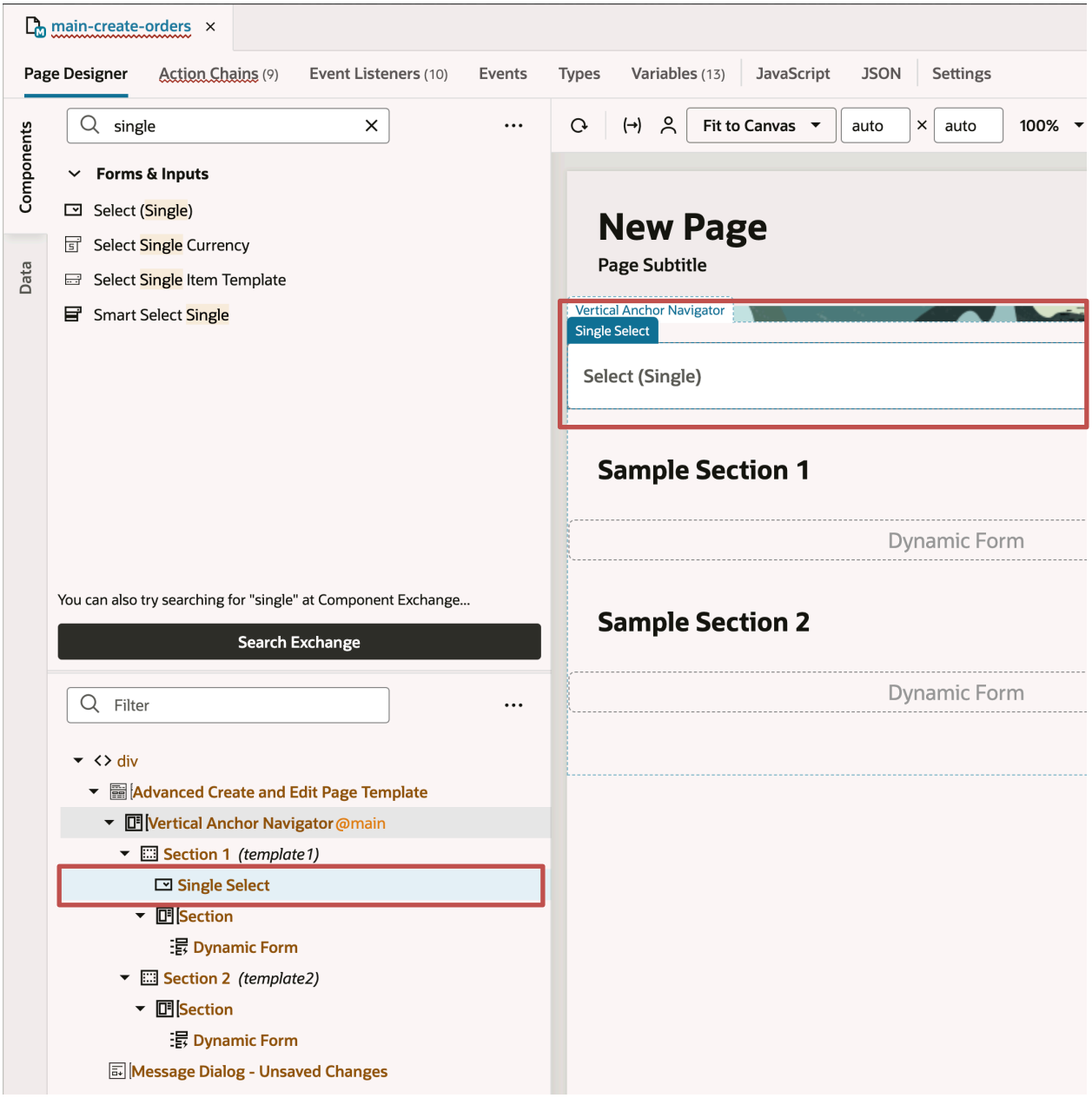
5. If the screen size is still set to Desktop from the previous pracrtice, change it to **Fit to Canvas**.



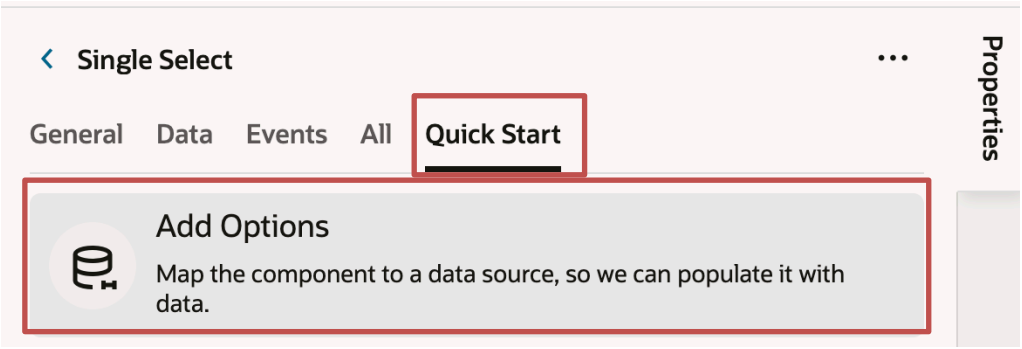
6. Add a **Select (Single)** to **Section 1**, above the **Section** component. **Tip:** Note the green horizontal line indicating where it will be inserted.



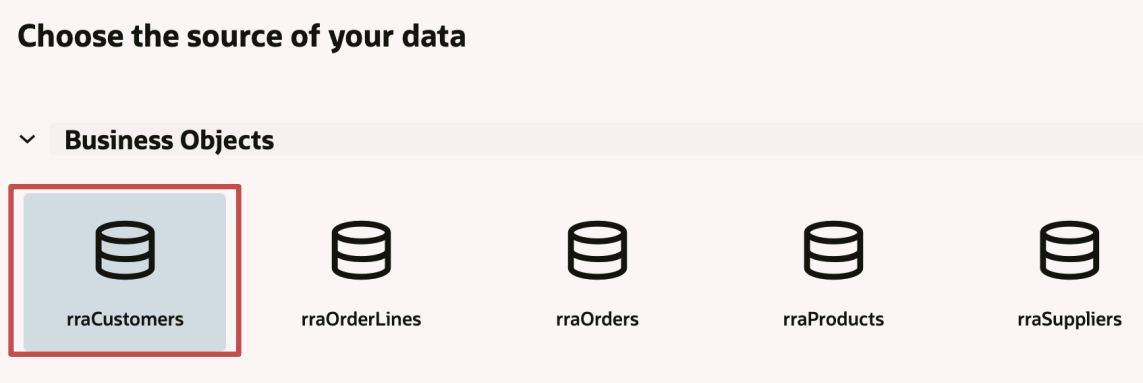




7. With the Single Select selected, click its **Quick Start** tab. Click **Add Options** to define its data.

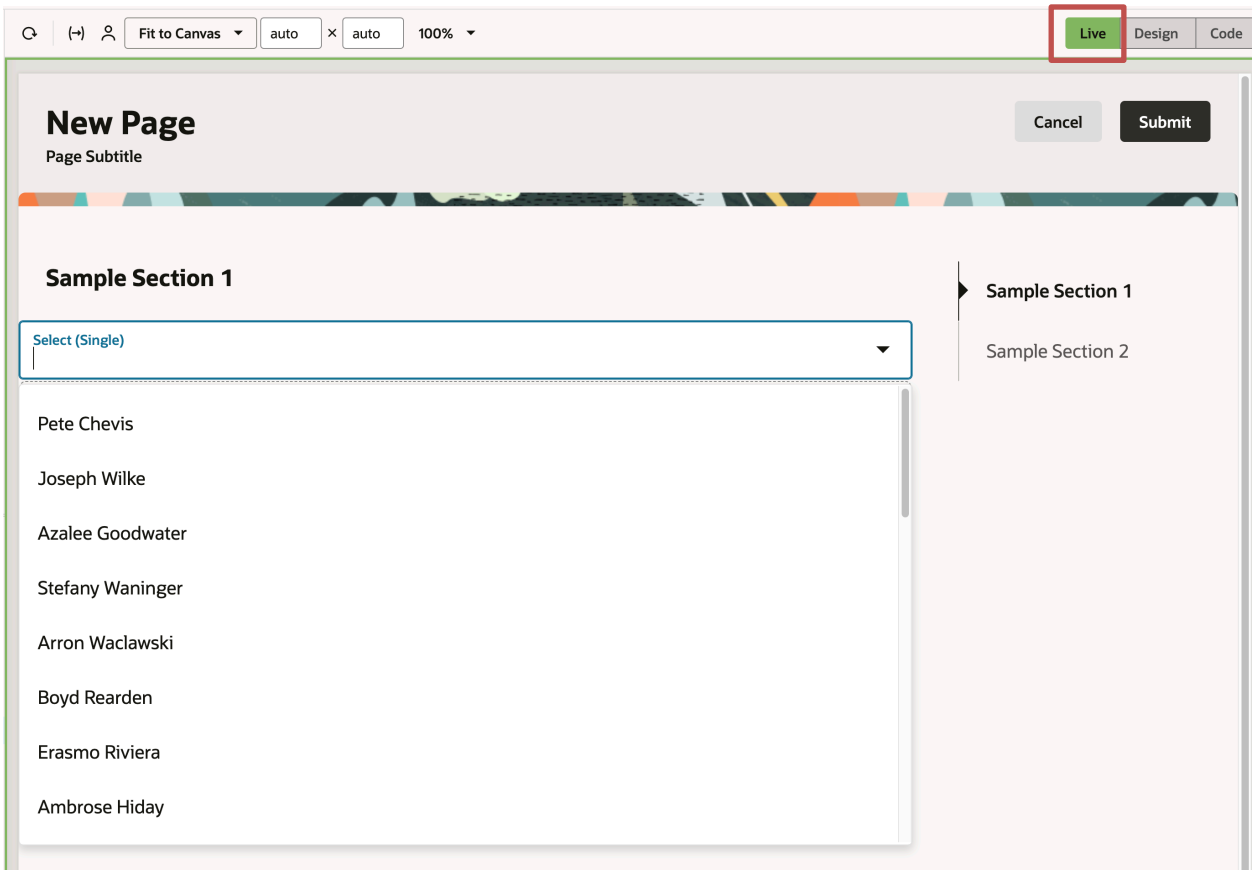


8. Select **rraCustomers** as your data source and click **Next**.

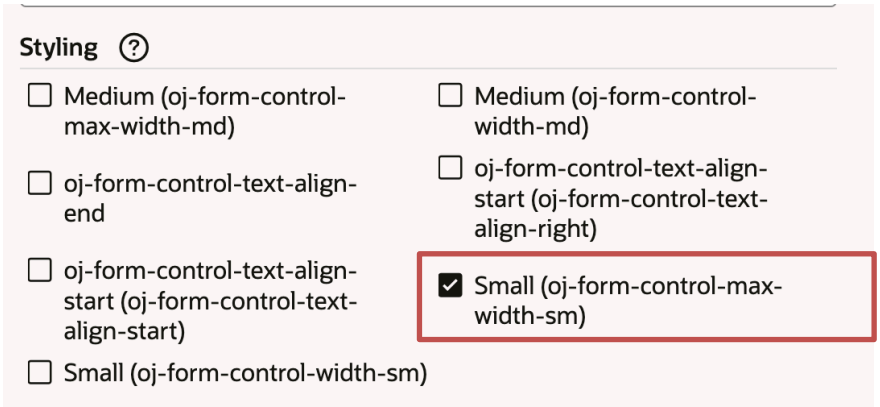


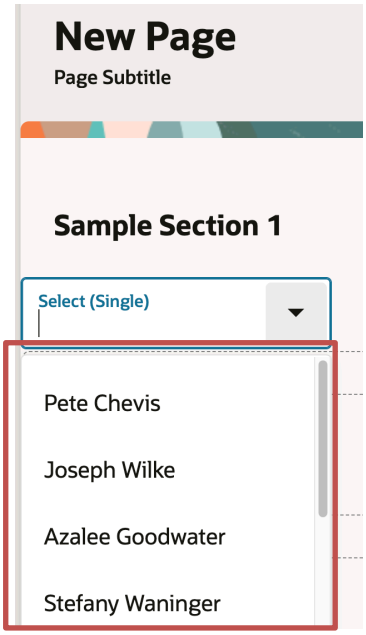
9. Select **customerName** for **Label** and **customerId** for **Value,** and click **Next** and then click **Finish**.

10. Test that it works in **Live** mode. Click the pull-down menu and, after a moment, the list of customers appears. Close the list and return to **Design** mode.

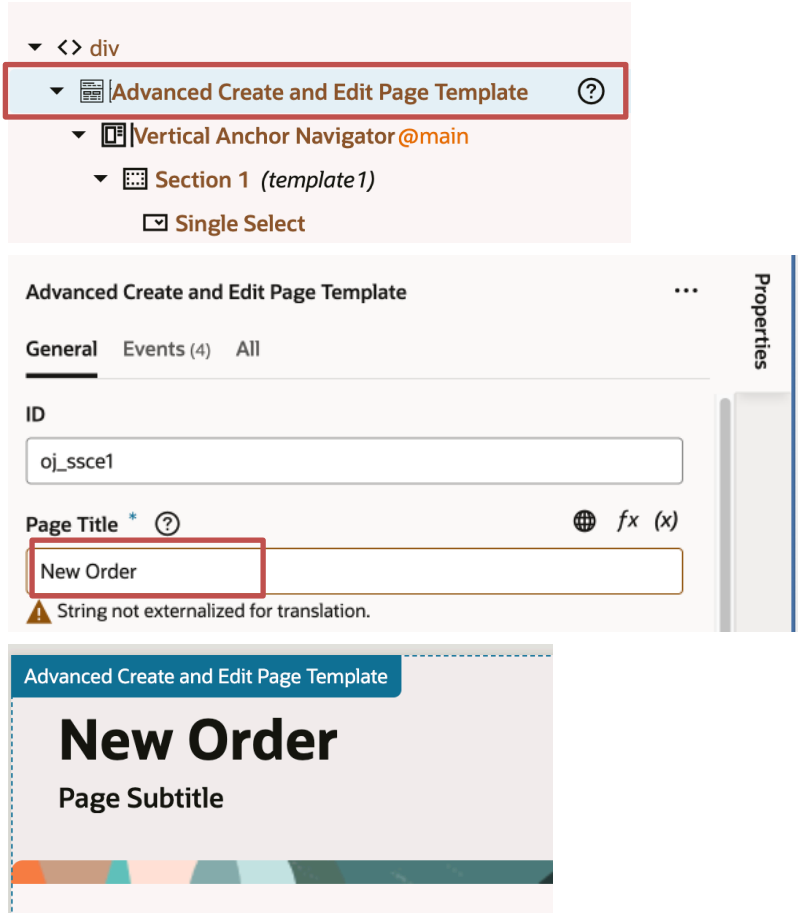


11. Notice the single select is wider than it needs to be. Click the Single Select's **General** tab. Select **Small (oj-form-control-max-width-sm)** for **Styling**. Test the single select again. It now adjusts to fit the contents better. Return to Design mode when done.

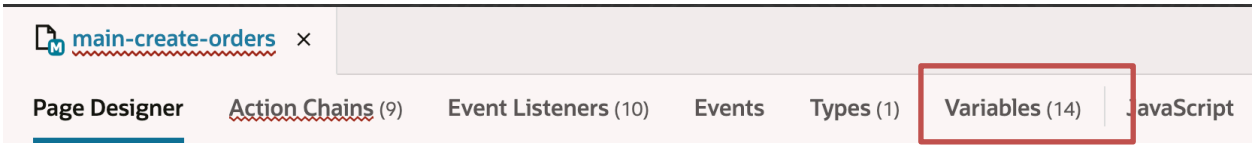




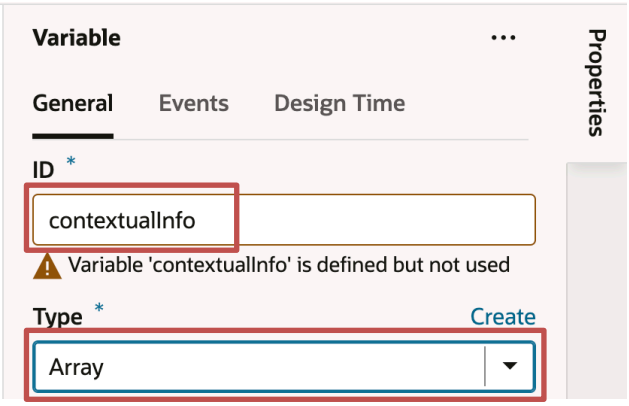
12. Select the **Advanced Create and Edit Page Template**. Set its Page Title to `New Order`.



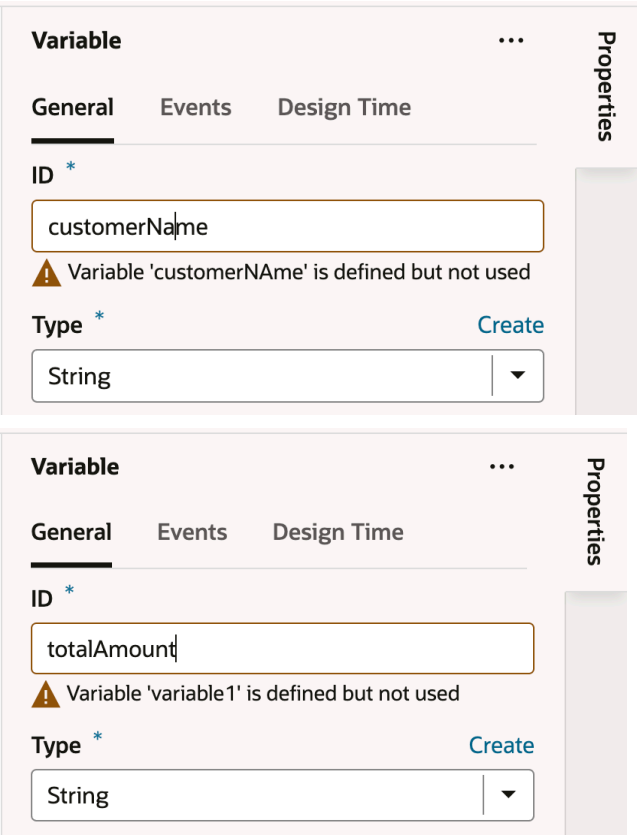
13. Create variables to display contextual information. Click the **Variables** tab.

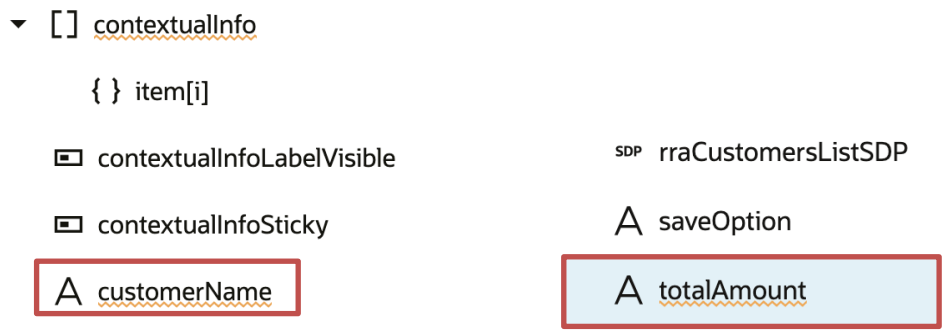


14. Create a new variable named `contextualInfo` of type **Array**.

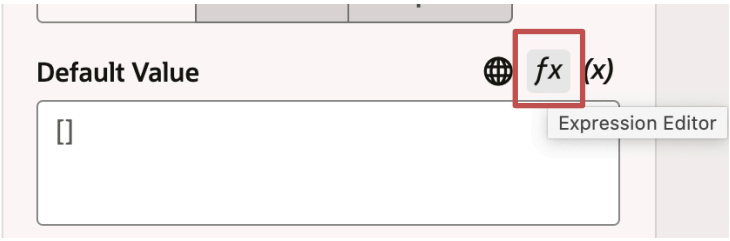


15. Create two variables of type String to hold the contextual values: `customerName` and `totalAmount`.





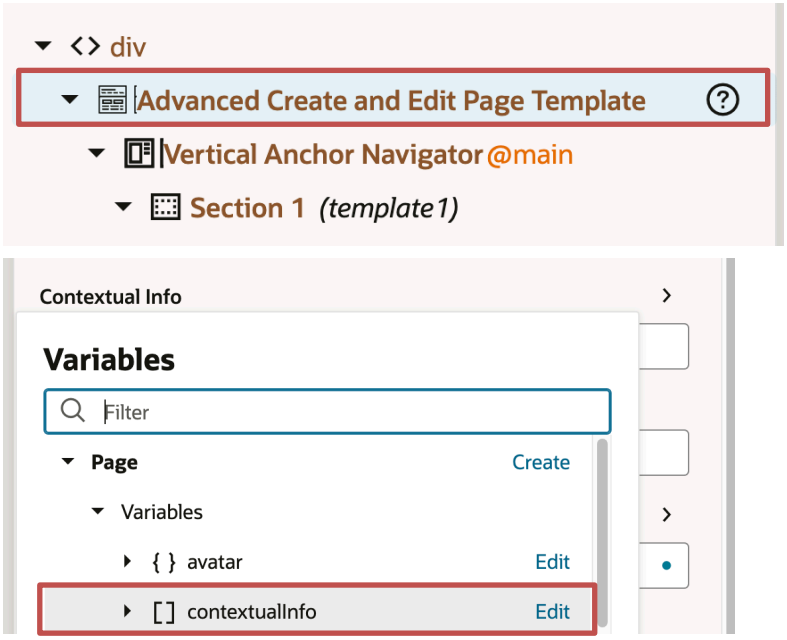
16. Select **contextualInfo** and open the Expression Editor for its Default Value property by clicking the **fx** button.



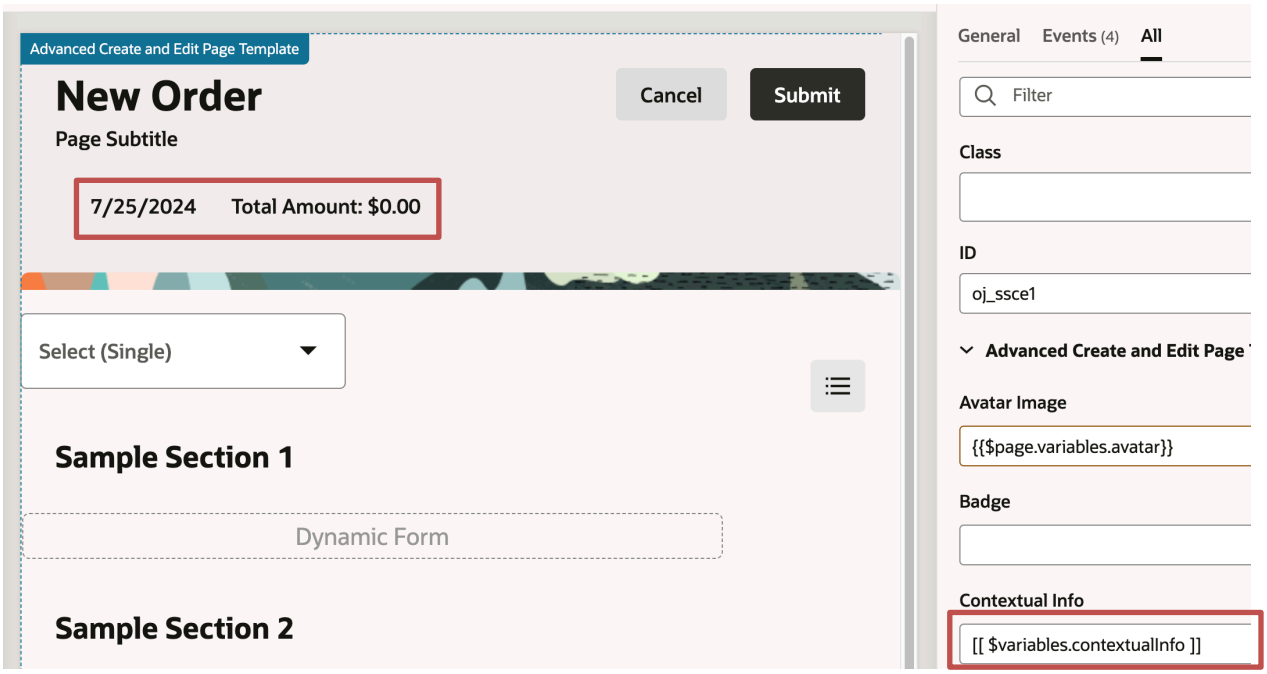
17. Paste this code into the editor. Click **Save** when done. The code can be found in your code text file.

```
[
    {
        label: 'customer',
        value: $page.variables.customerName
    },
    {
        label: 'date',
        value: $flow.functions.formatDate( Date.now())
    },
    {
        label: 'totalAmount',
        value: 'Total Amount: ' +
$flow.functions.formatCurrency($flow.constants.defaultCurrencyCod
e, $page.variables.totalAmount)
    }
]
```

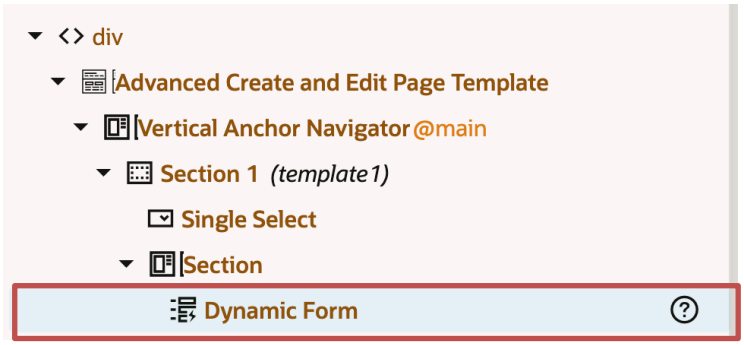
18. Click the **Page Designer** tab. With A**dvanced Create and Edit Page Template** selected, click its **All** tab. Choose the **contextualInfo** variable for the **Contextual Info** property.

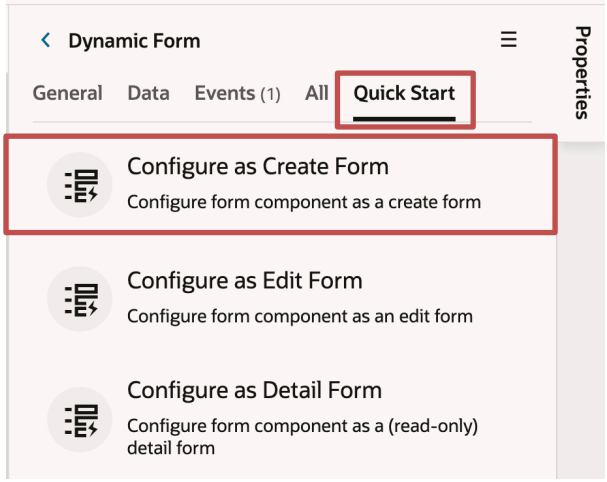


19. The contextual info displays in the banner. As a new order is built on the page, the **totalAmount** variable would be updated and the banner would reflect the change in value.

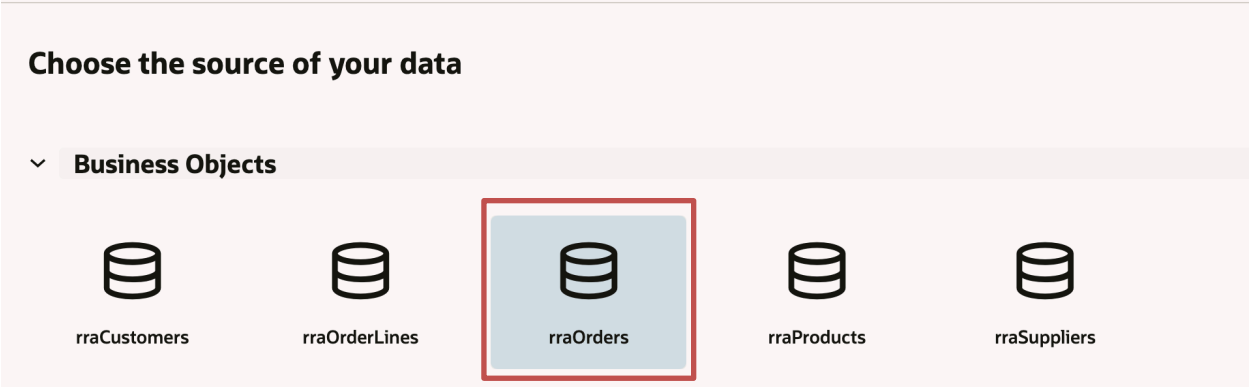


20. Select the **first Dynamic Form** to create the form for the new order. Select its **Quick Start** tab and click **Configure as Create Form**.

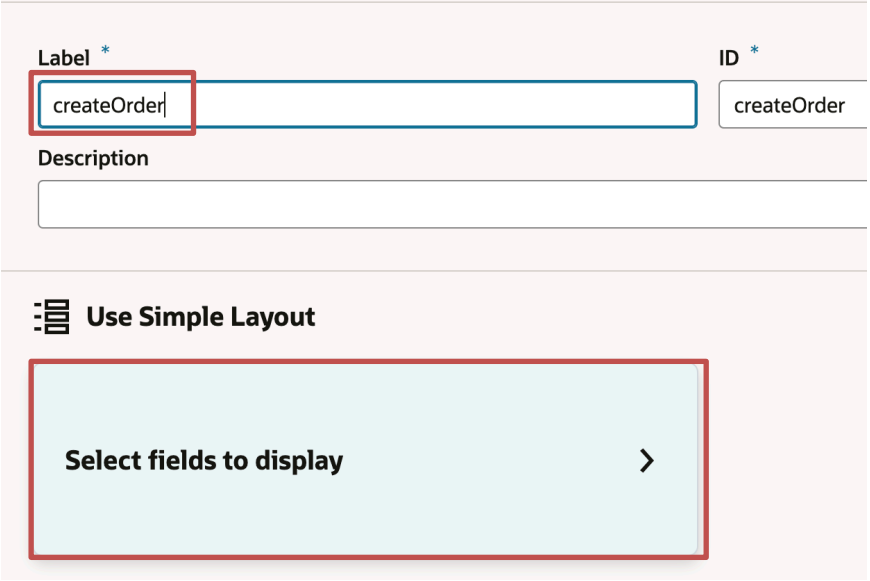




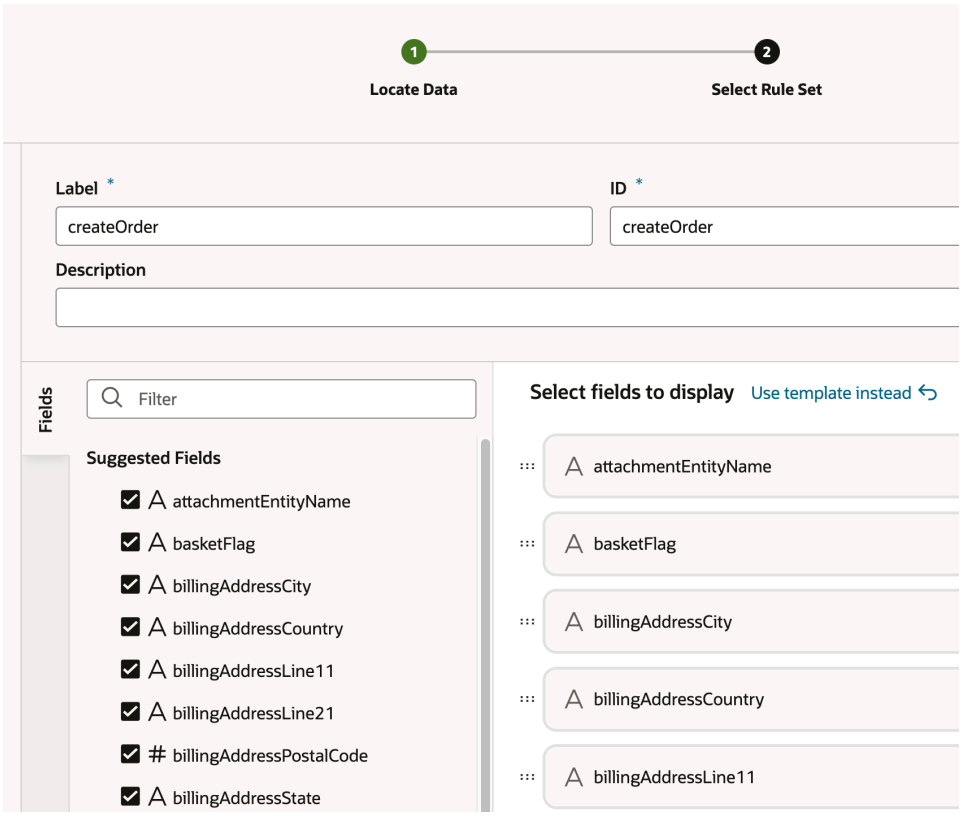
21. Select **rraOrders** for its data source and click **Next**.



22. Enter `createOrder` for the **Label**. Click **Select fields to display >**.

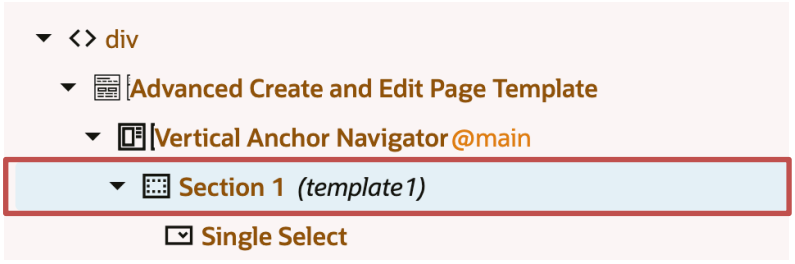


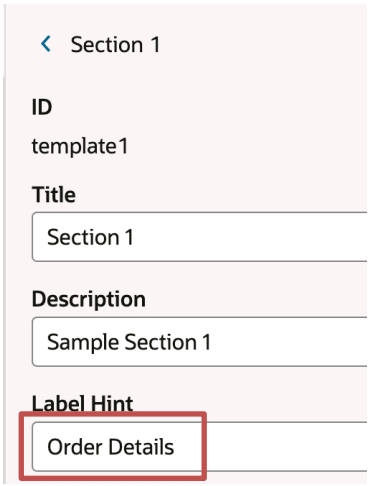
23. Click **Finish** to accept all the fields.
    Note: In this example, you accept all the fields for simplicity, but in a real application, you would choose what would be entered, and you would want to reference the selected Customer Number and possibly use default values for other fields. You are not creating a working create operation; you are doing this to see the resulting form on the page.

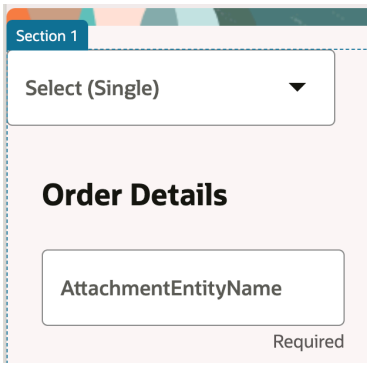


You are returned to Page Designer. After a moment, the form fields display.

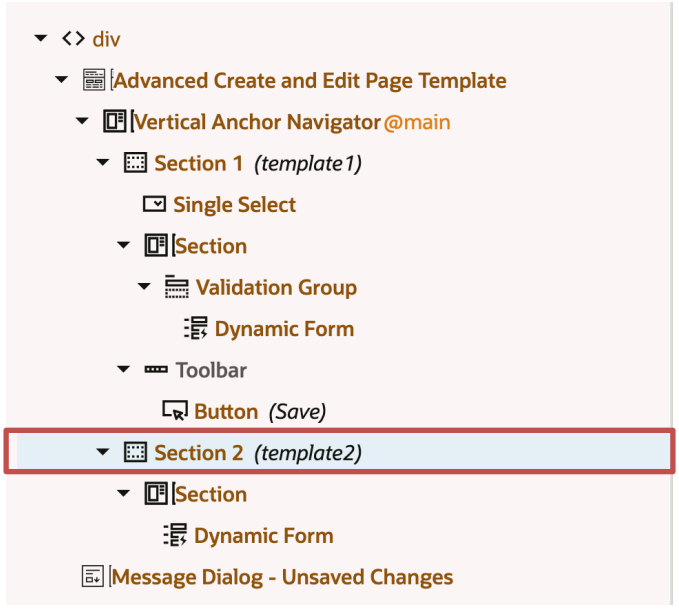
24. Select the **Section 1** element and change its **Label Hint** to `Order Details`.

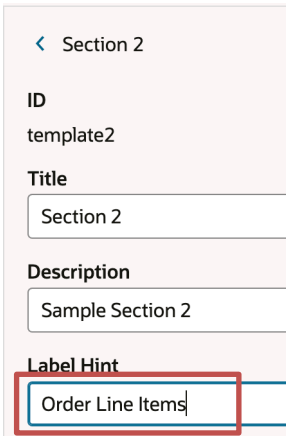




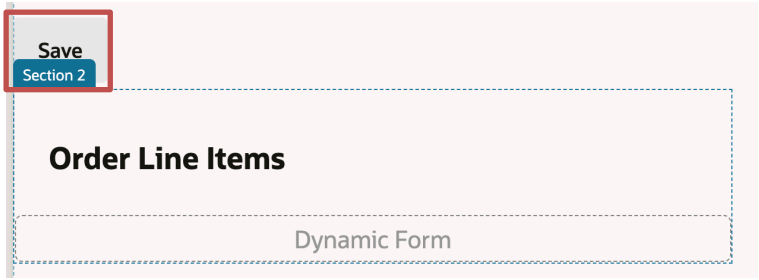


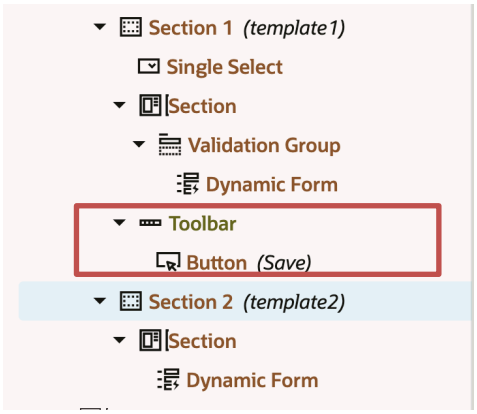
25. Select **Section 2** and change its **Label Hint** property to `Order Line Items`.





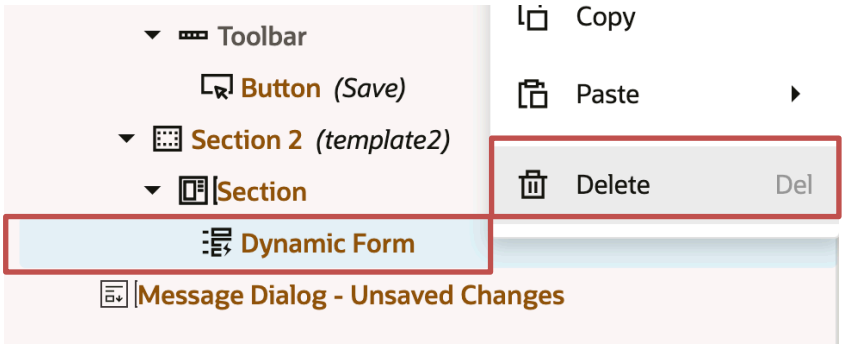
Notice a toolbar and **Save** button were added for the **Section 1** form to save the master order data first and then save the detail rows.

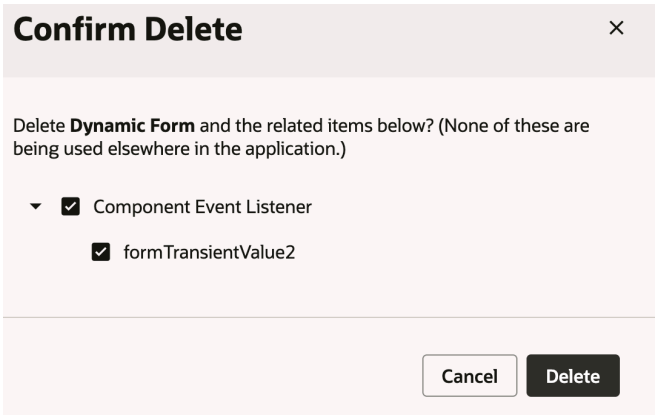




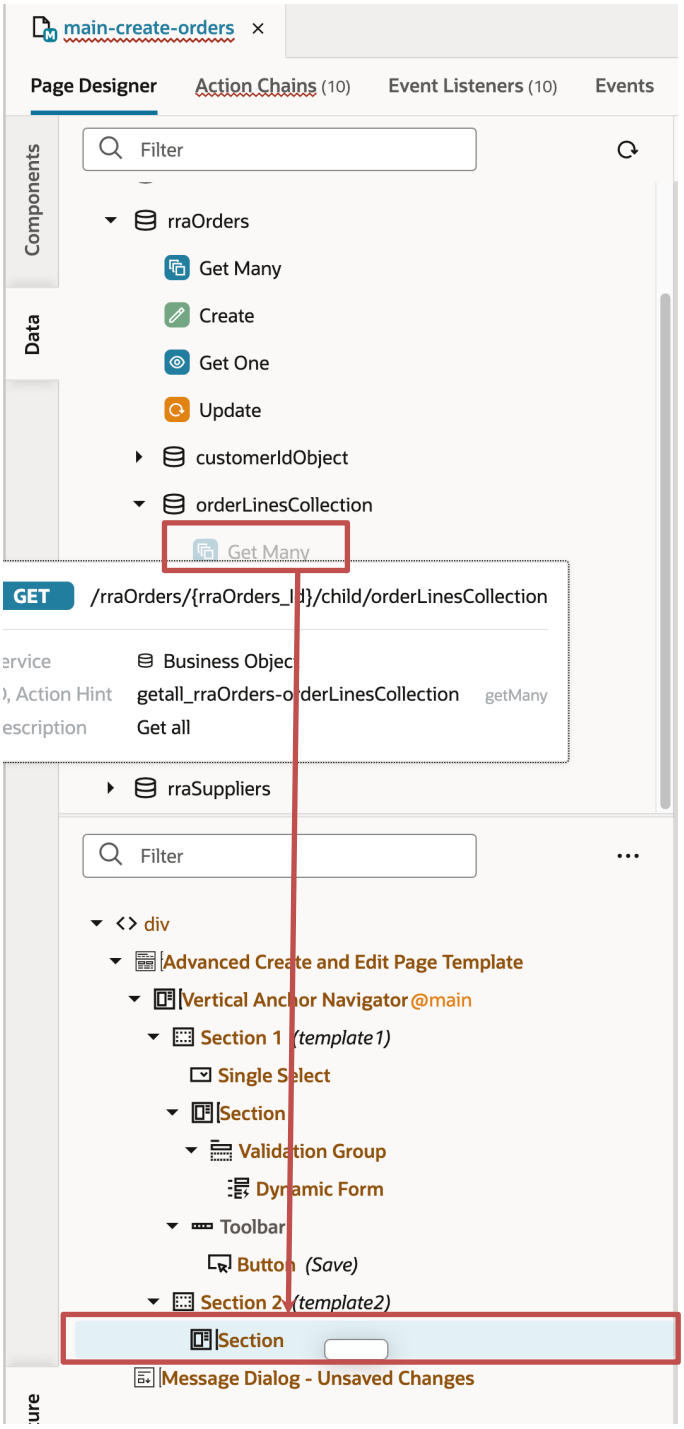
This is the default behavior, but you could combine the save of the order (master record) with its line items (detail records) and use the **Submit** button to save and commit both and then navigate back to the parent page (main-manage-orders), or use **Save** to save the order in case you still wanted to make changes before submitting and committing. You could then add line items and use the **Submit** button to submit the added line items and order. You could also delete the Save button entirely and let the Submit button save the master record and details records and navigate back to the parent page. The Submit action chain is coded to call the Save action chain and then navigate back to the parent (root) page. This is implemented with action chains and implementing it is beyond the scope of this course. But notice the flexibility built into the page template.
Optional: Feel free to explore the event listeners and action chains to see how this is implemented.

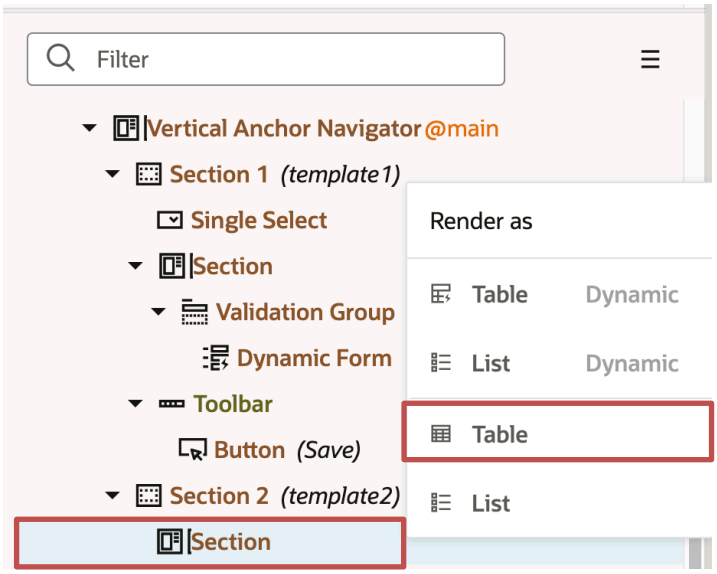
26. Add a table to add line items to the order. Replace the **Dynamic Form** in **Section 2** with a table. Select the **Dynamic Form** in **Section 2** and delete it. Confirm the deletion of the **Component Event Listener** and **formTransientValue2**.



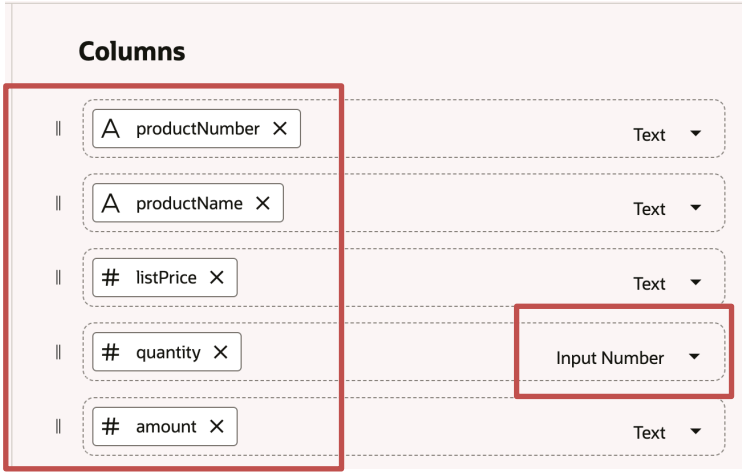


27. Take a moment to study the documentation for the **Advanced Create and Edit Page Template**. Click the **Components** tab, click **Installed** and click **Advanced Create and Edit Page Template** - not the pattern. Notice how objectId is used for the master record and is passed to the detail record. Also, the page can be used to retrieve a parent (master) record and then add children to it and submit the added children. This functionality is built in but is beyond the scope of this course to use.
28. Click the **Data** tab, expand **Business Objects > rraOrders > orderLinesCollection** and drag the **Get Many** to **Section** under **Section 2**. Choose **Table** to Render as. You can use a table to add multiple order lines and then commit them as a batch or use **Create** to add and save one line item at a time. Implementing this functionality is beyond the scope of this course.

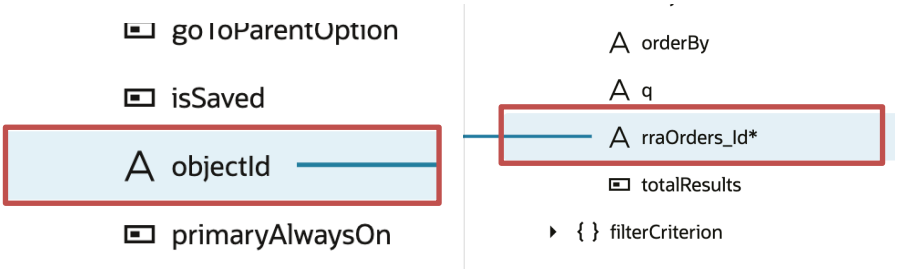




29. From the Endpoint Structure, select the following fields in order: **productNumber**, **productName**, **listPrice**, **quantity,** and **amount**. Change the component type for **quantity** to **Input Number**. Click **Next**.

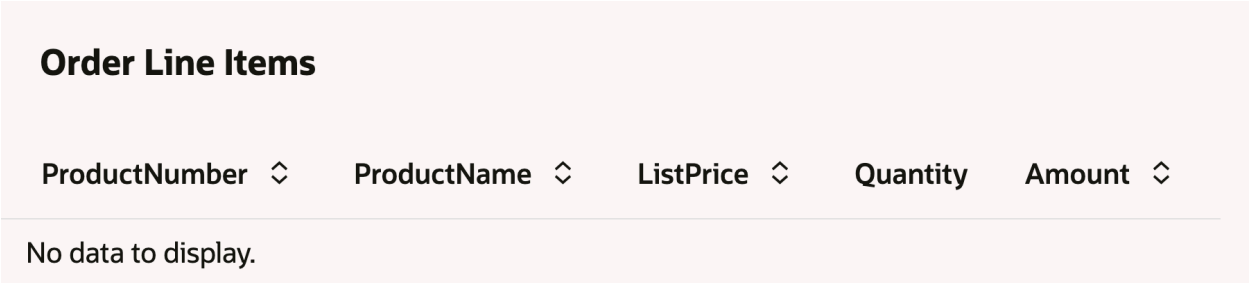


30. Drag **objectId** to **Target > getall_rra-rraOrders-orderLinesCollection > rraOrders_Id**. ObjectId is the value from the parent record that is needed for the detail record as a ”foreign key.” Click **Finish**.



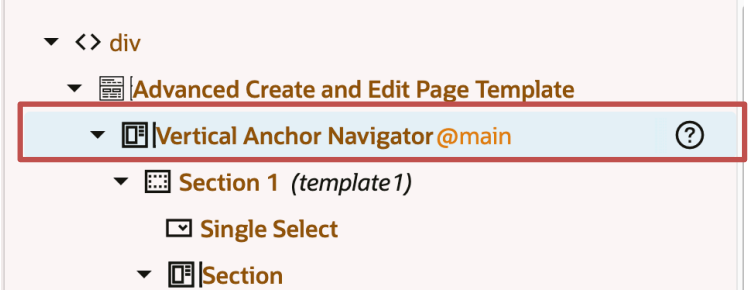
You see the Table display. You don’t see any rows, as no order lines have been created yet. Normally, you would code this to allow you to use buffer variables to store the values for the new row and then insert it into the table. The technique to do this is

presented in the Visual Builder Cookbook that was mentioned at the beginning of this practice. You will not be implementing that functionality here because it is not relevant to Redwood.

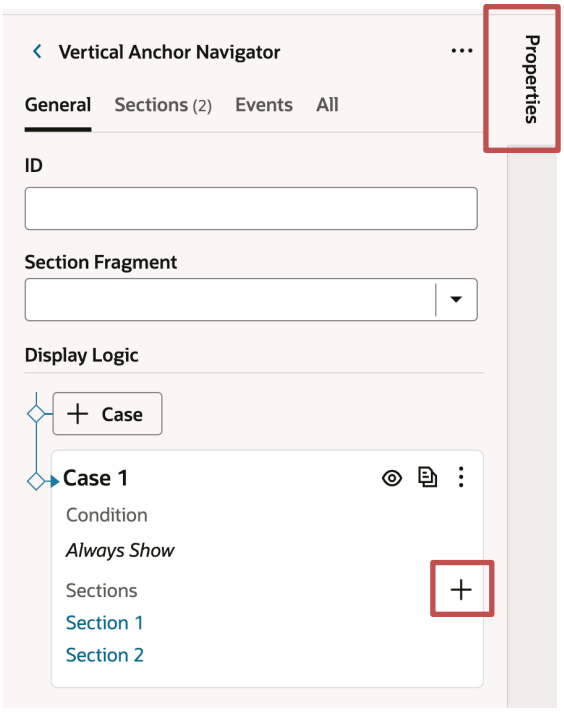


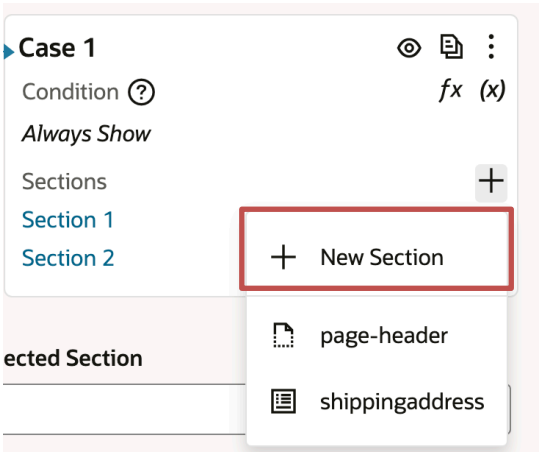
To add a line item, you need to select the desired product, have its Product Number and List Price added to the new row, and then enter the quantity. The Amount would be calculated in a JavaScript or API function based on the quantity and list price. While you won't be doing all that in this practice, you will add a Table to display the Products. You first add a new Section to hold the table of products. You want the new section to be accessible via the right-hand side Vertical Anchor Navigator. You will add a new section using this component.

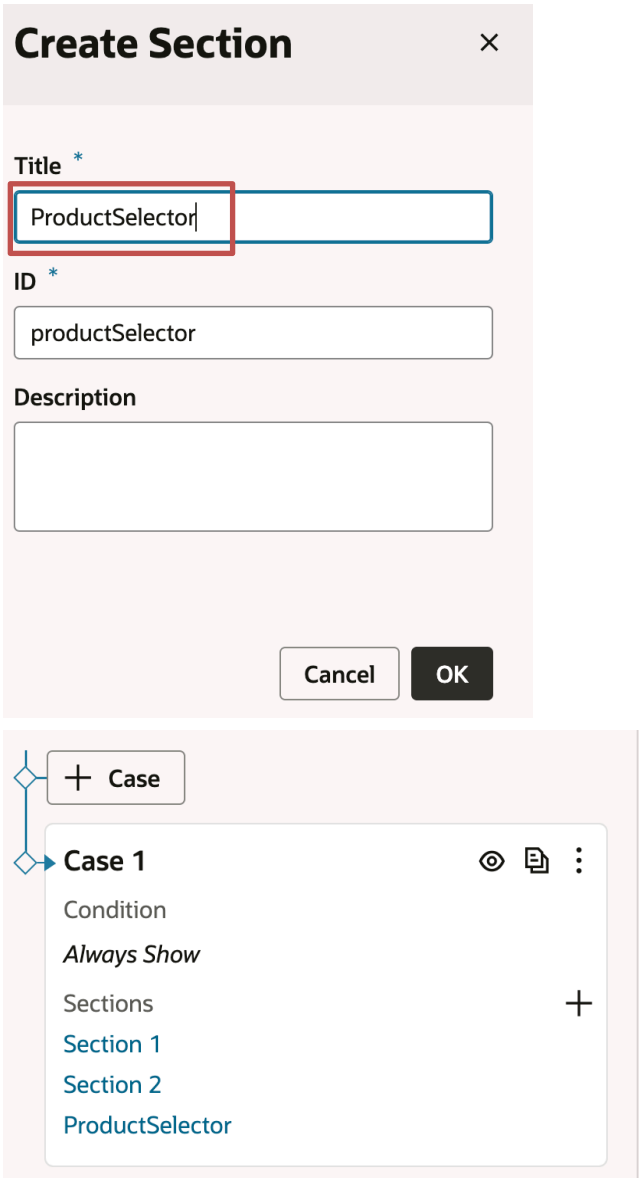
31. Select the **Vertical Anchor Navigator**.



32. In its **Property** pane, click the **+** to add a new **Section**. Name it `ProductSelector` and click **OK**.

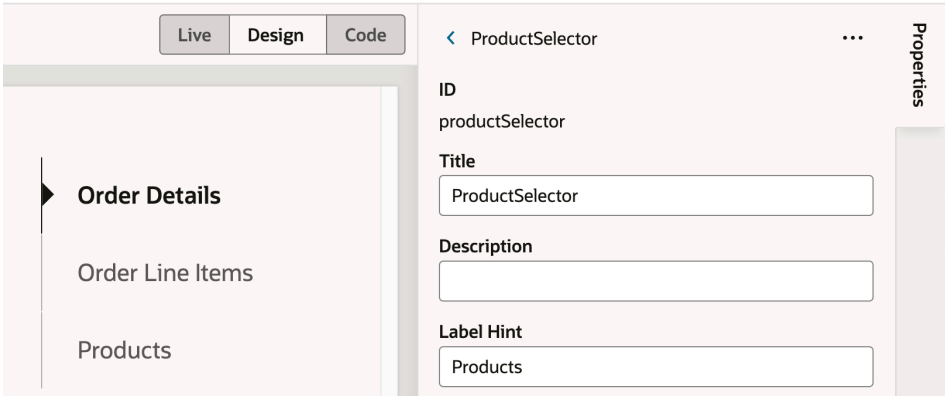




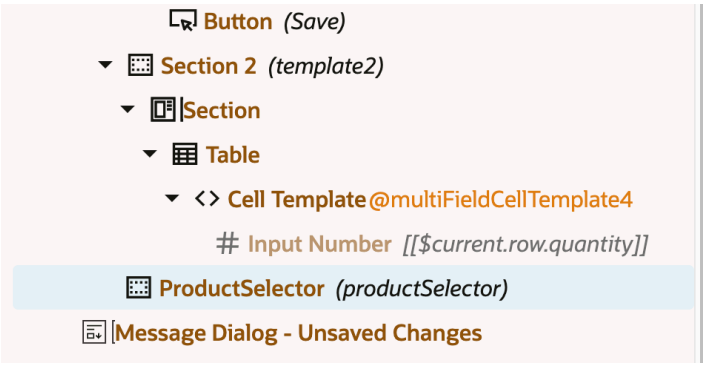


Tip: Did you notice the fragment you created earlier: **shippingaddress** was available to add as a section? This is another good use of fragments for leveraging repeatable code.

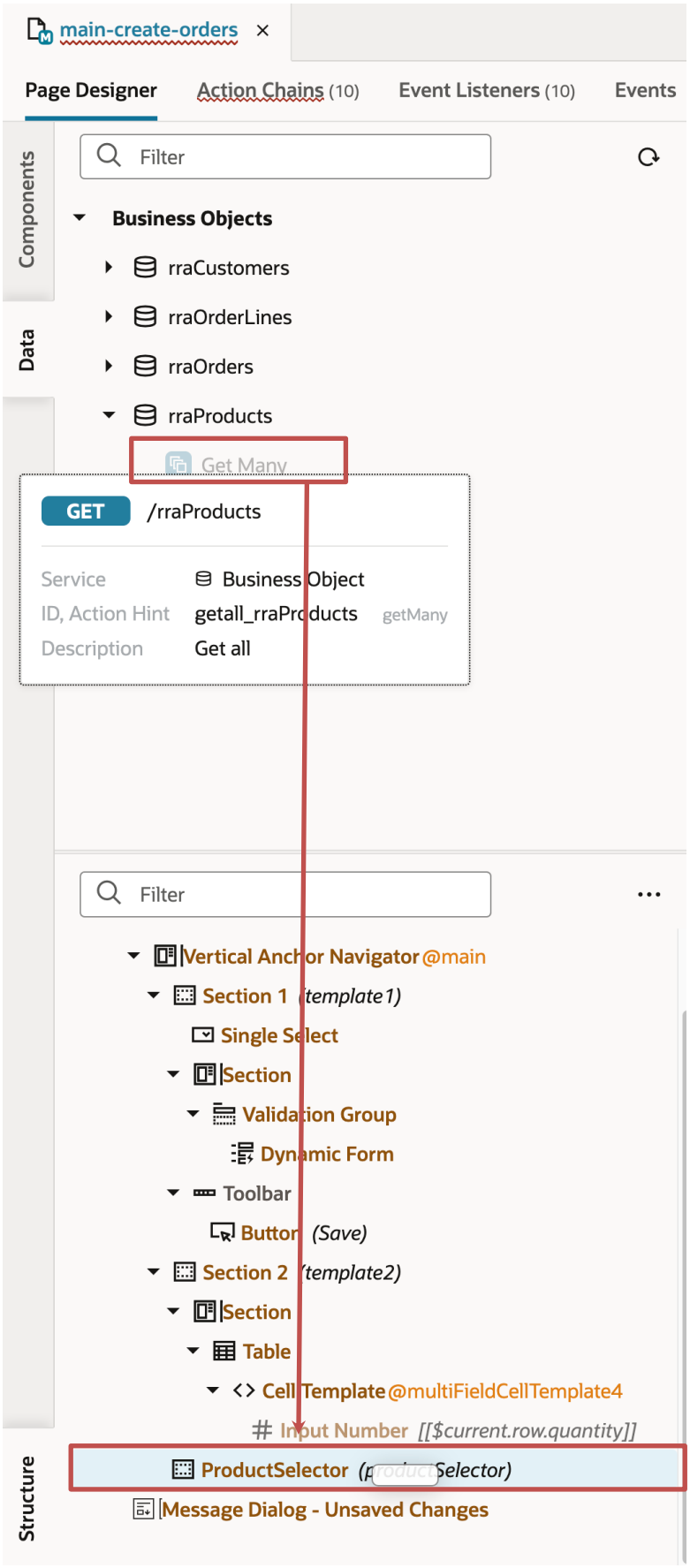
33. Select the new **ProductSelector** section and set its **Label Hint** to `Products`.

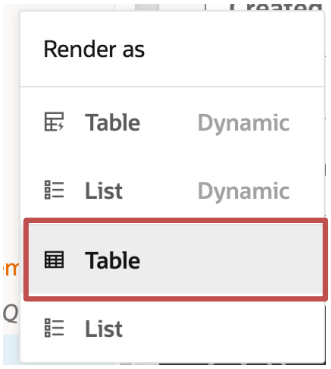


Notice you don’t see the new section because it is too low on the page and is hidden. When you test the page, you will see how it is displayed.

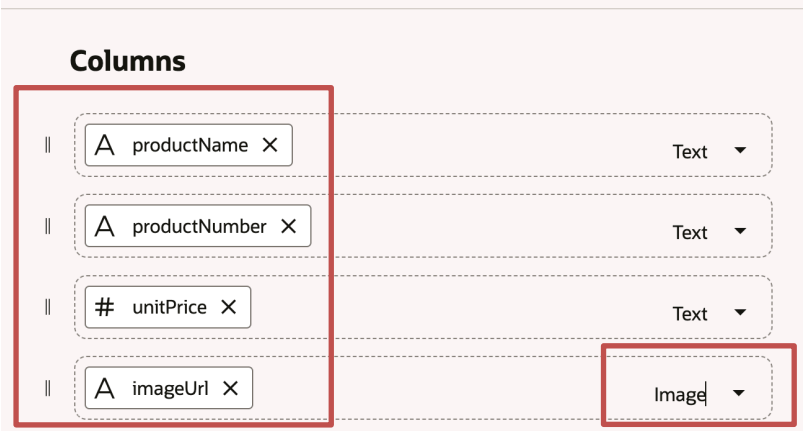


34. Click the **Data** tab and expand **Business Objects > rraProducts** and drag **Get Many** into the new **ProductSelector** Section. Render as a Table when prompted.



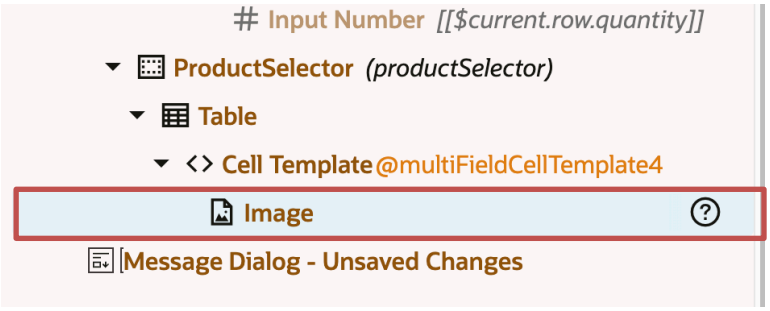


35. Select the following fields in order: **productNumber**, **productName**, **unitPrice**, **imageUrl**. Set the component type for **ImageUrl** to **Image**. Click **Next** and **Finish**.



36. Select the **Image** component, click its **Data** tab, and change the **Source URL** to:

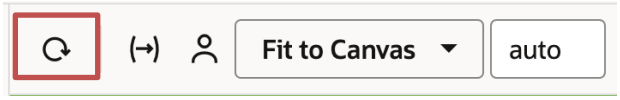
```
[[  $application.path + 'resources/' +$current.row.imageUrl ]]
```

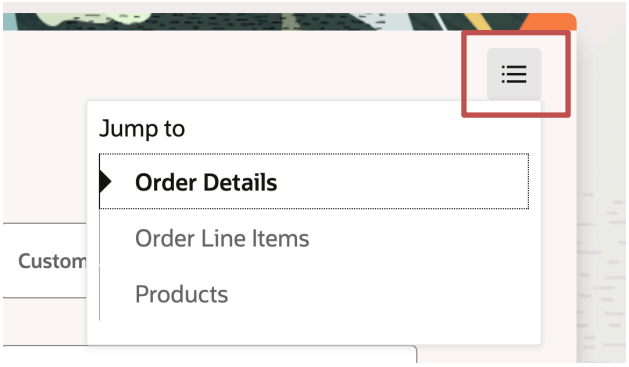


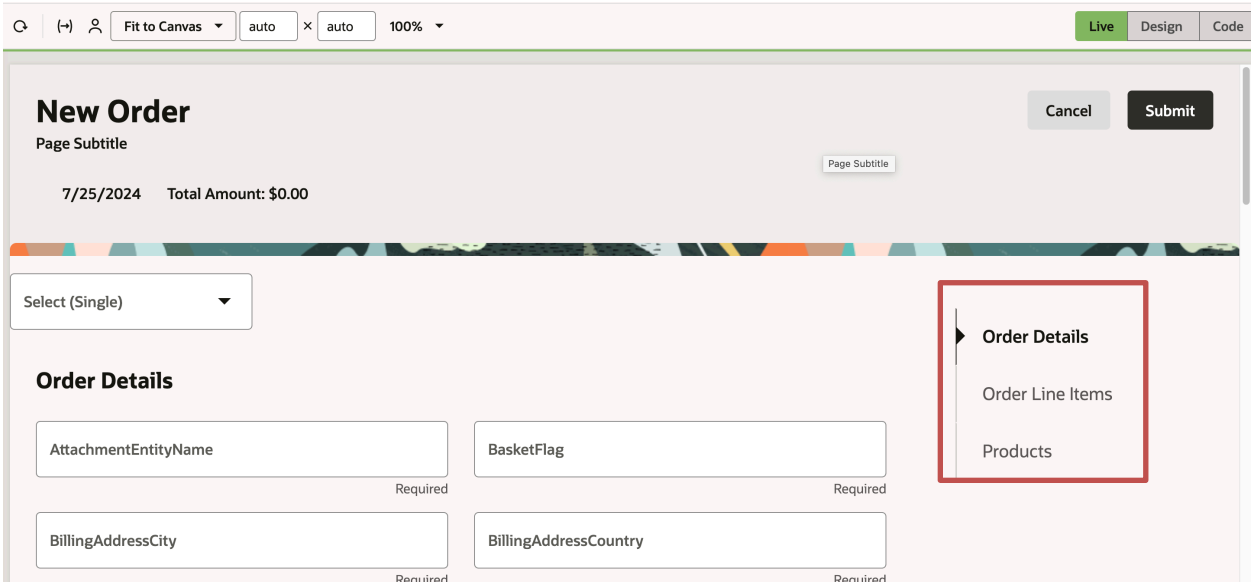
37. Test this in **Live** Mode. You should be able to click the **Products** link on the right and the page will scroll to the list of products. You can also use the navigator to jump to the **Order Details** or **Order Line Items** sections.

    Hints:

    - If the page does not scroll to the different sections, click the **Reload Design Preview** icon.
    - If you don't see **Jump to**, click the Actions icon or make the page wider.







    Note: Notice how the page is responsive and displays the Anchor component or an action button as needed.

    In a real app, you would add a row selection to the Products table to select a row in the table and use its values to populate the corresponding fields in a new row in the Order Lines table.

38. Return to Design mode.
39. Close all tabs.

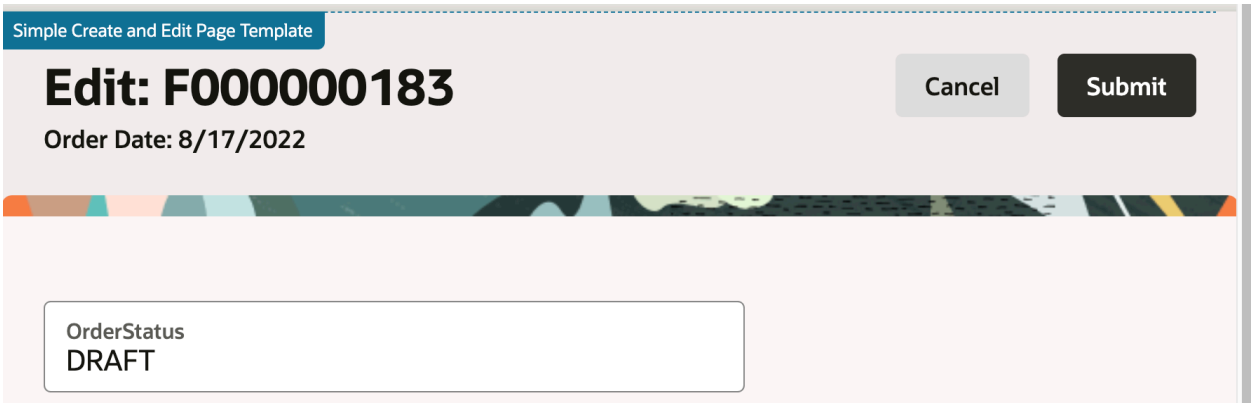
This completes this practice.

Practice 4-2: Create the Edit Orders Page

Overview

In this practice, you create the Edit Orders page, which accepts a passed order number, retrieves the order, and presents fields for edit and save. You will use the Simple Create and Edit Page Template as a basis for the Edit page.

Note the edit functionality backend is not implemented, so you're working with the frontend design only.



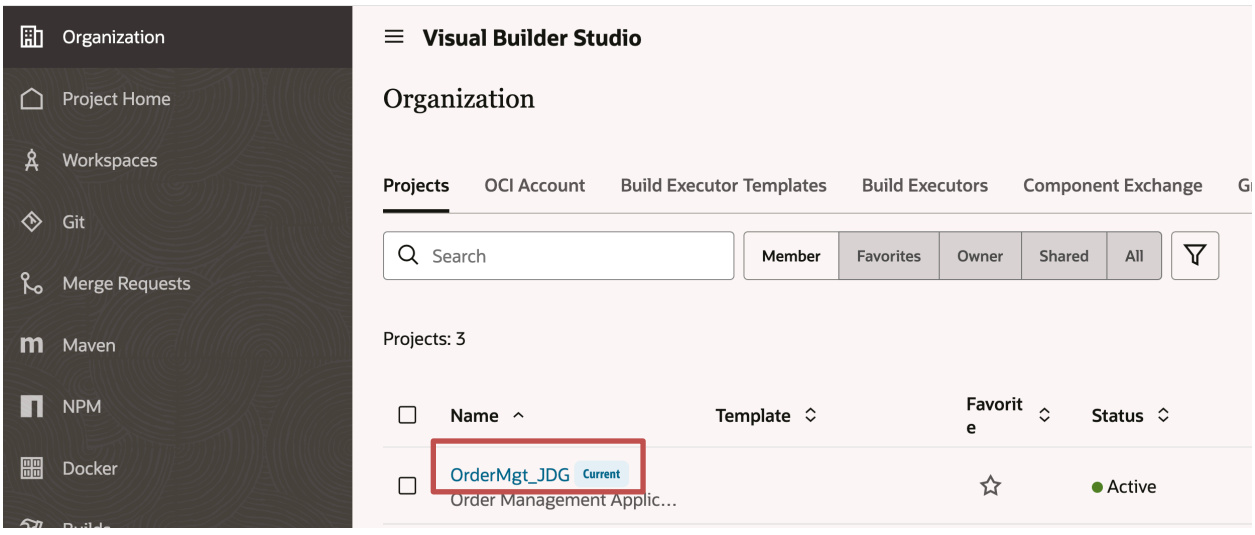
Assumptions

You have successfully completed practice 3–1.

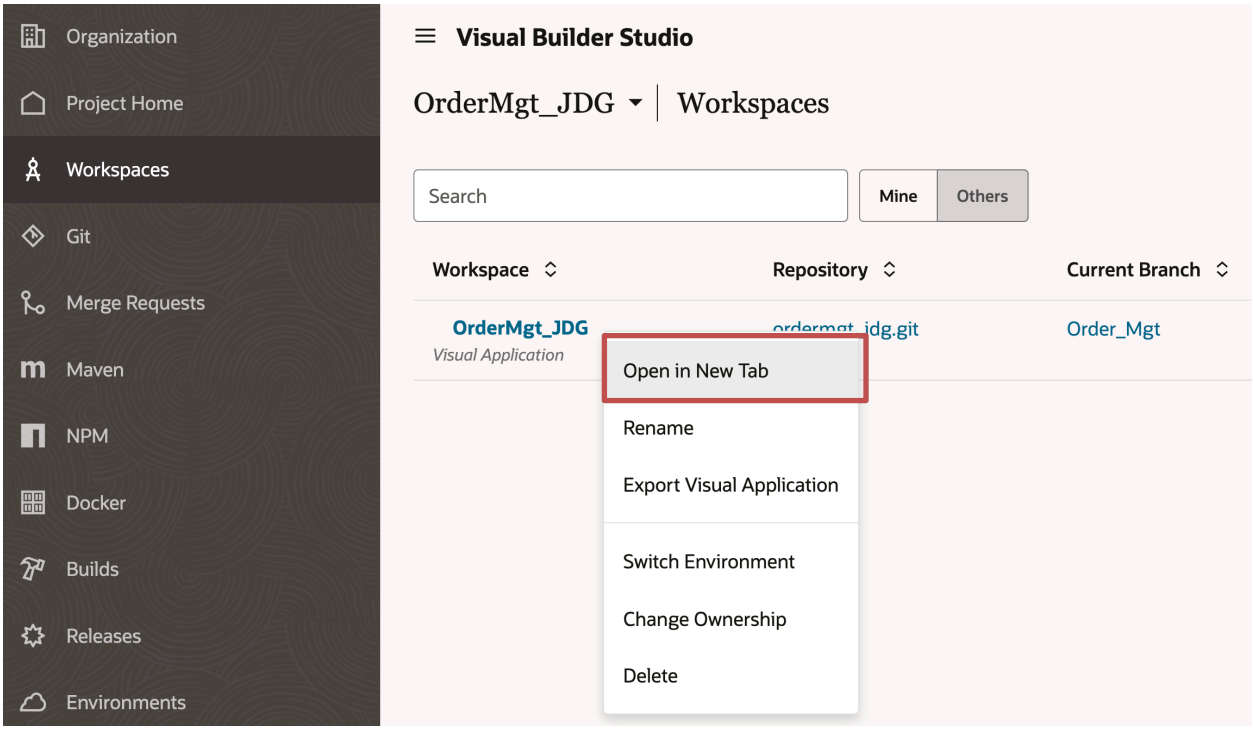
Tasks

If Visual Builder Studio Designer is still open from the last lesson, skip to step 4.

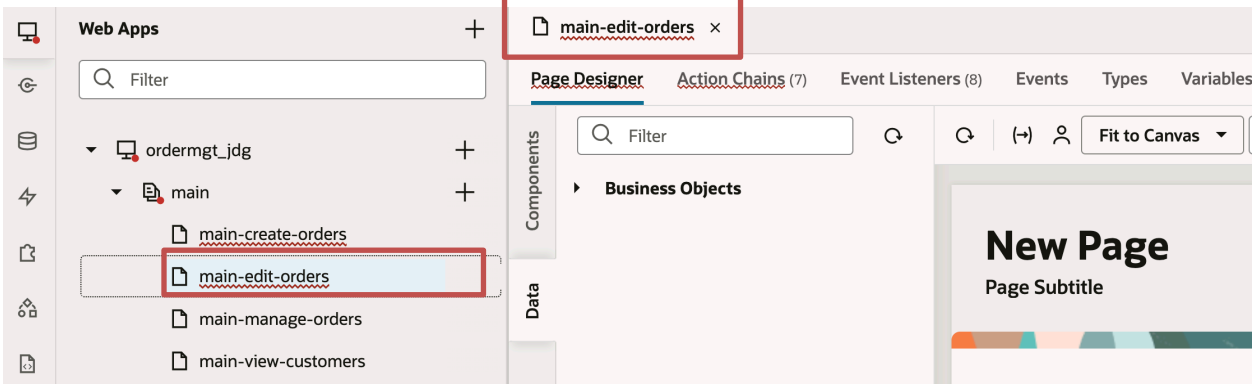
1. Log in to **Visual Builder Studio** and select your project name on the **Projects** tab. Your project name will be **OrderMgt_< your initials >**, where **< your initials >** corresponds to your initials.



2. The Project Home page opens. Right-click the **OrderMgt_<your initials>** workspace and select **Open in New Tab**.



3. Close the **Getting Started** tab if it's open.
4. Select the **Web App** tab in the Navigator. Expand **ordermgt_<your initials>**. Expand **main**. Select the **main-edit-orders** page. It opens in the Page Designer.

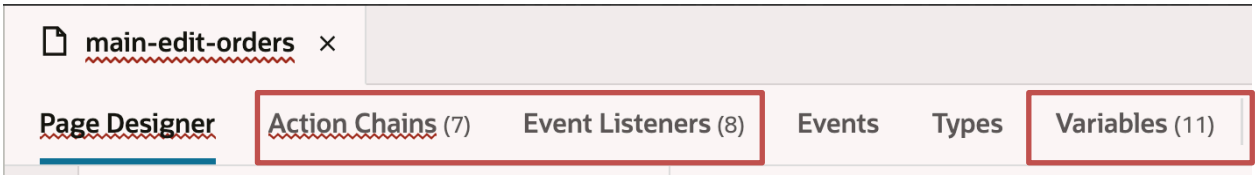


Similar to the Advanced Create and Edit Page Template, this has a dynamic form for its main section. This is where you choose the fields that will be included for edit. Notice the page comes with Cancel and Submit buttons and will have a Save button as well. The intent is that the Save button would allow the user to make changes to the editable fields and save without navigating off the page back to the parent or root page. The Submit button calls the Save action chain and then navigates back to the parent or root page. If you desire, you can have Submit be the only button and perform save and navigation at once and not have a separate Save button.

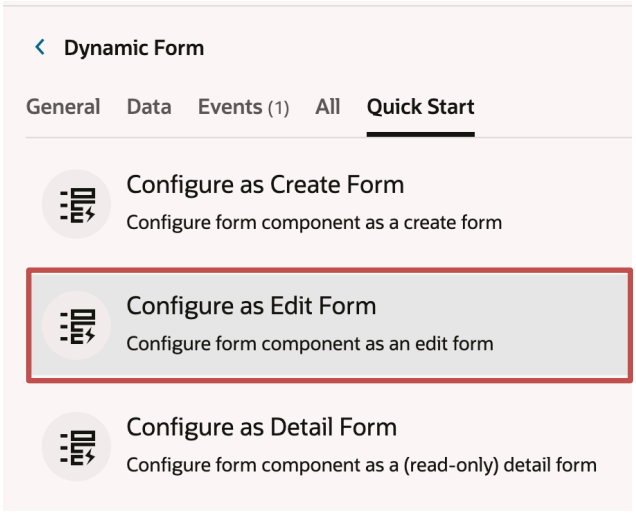
For example, in this application, the business rule is that orders cannot be edited once they're created. New orders are set to DRAFT order status, and so appear on the welcome page, and the edit will simply change the DRAFT order status to PROCESSING.

So, in this case, you would remove the Save button, but you won't do so in order to see a typical edit page.

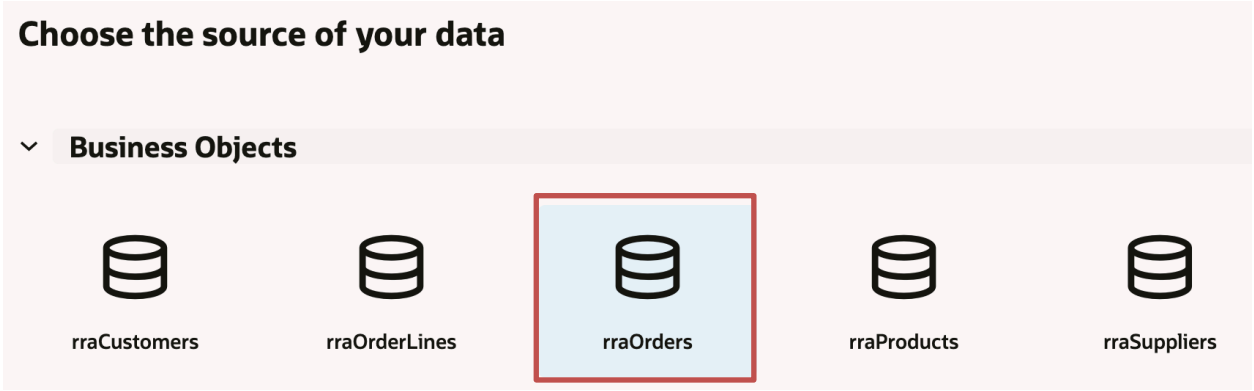
Notice this page template comes with predefined actions, event listeners, and variables, and they will all be used to support the edit or create operations, depending on what is selected.



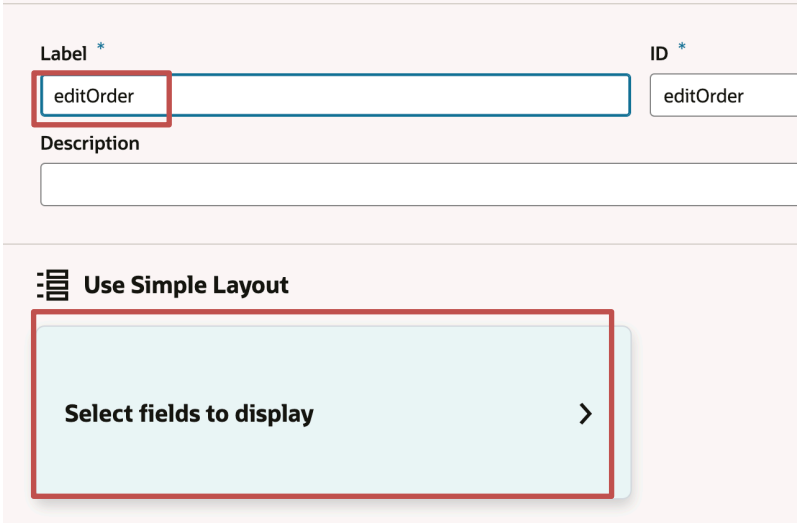
5. Select the **Dynamic Form** in the **Structure** pane, click its **Quick Start** tab, and select **Configure as Edit Form**.



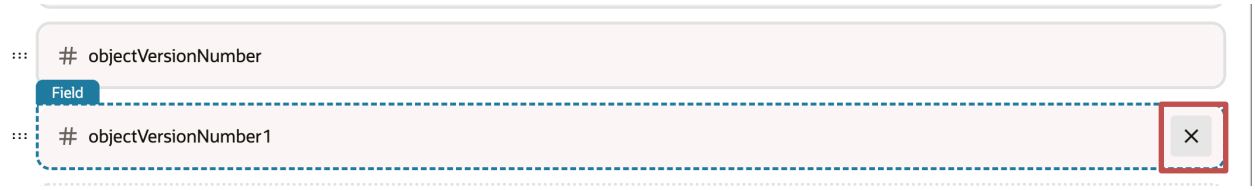
6. Select **rraOrders** as the source of your data and click **Next.** On the **Select Save Data** step, click **Next** again. Notice you can choose a different endpoint to save the data to. In this case, you're using the same data source for both locating the data and saving it.

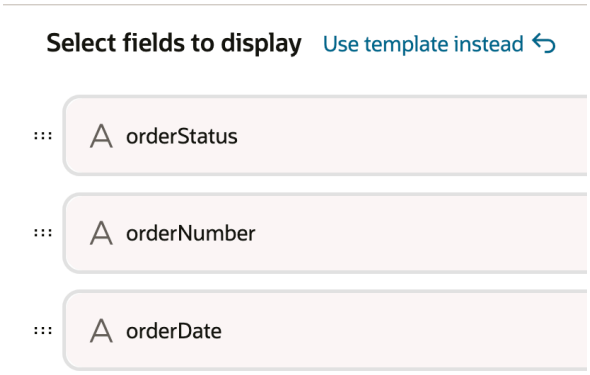


7. Enter `editOrder` for the **Label** and then click **Select fields to display >**.



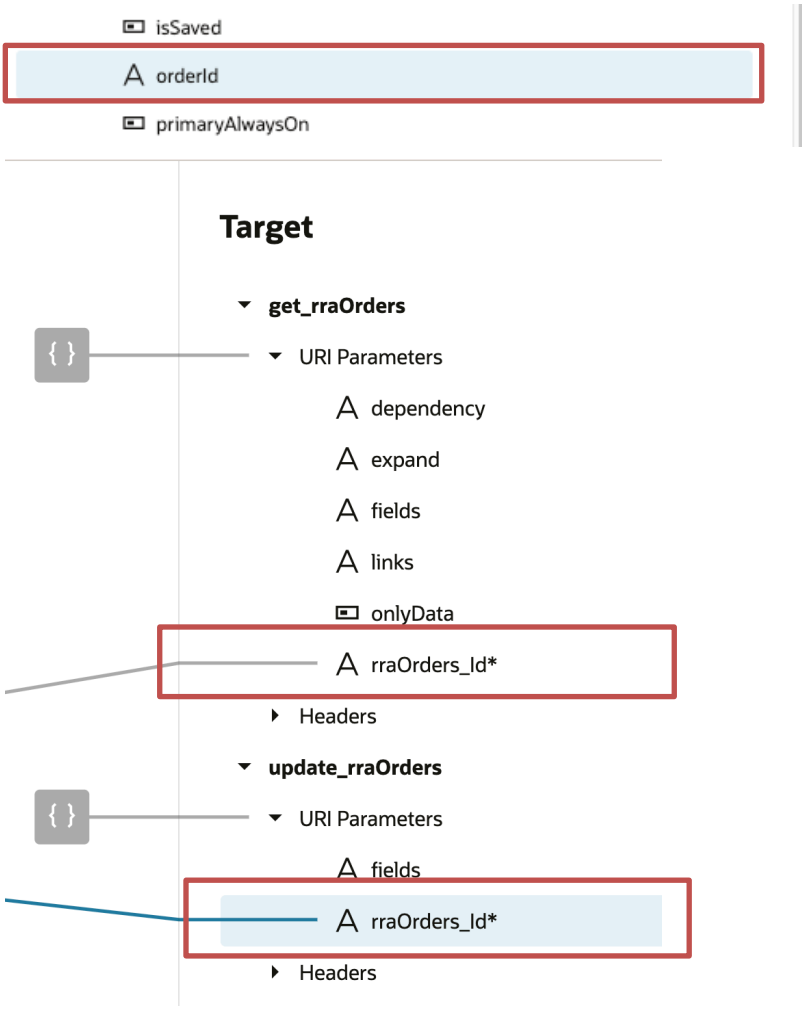
8. Remove all fields except **orderStatus**. Click the **X** at the end of the field to remove that field. Then, from the list of fields to the left, select **orderNumber** and **orderDate**. Click **Next**.



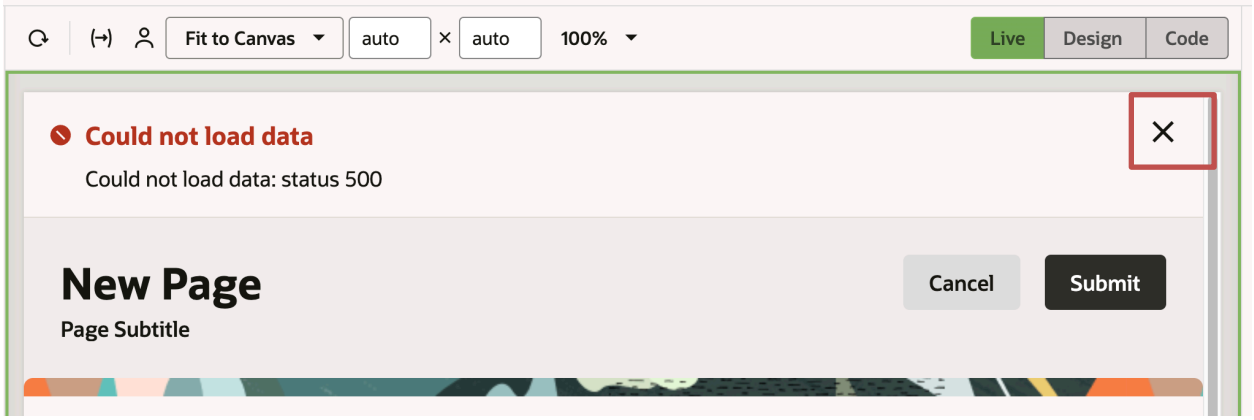


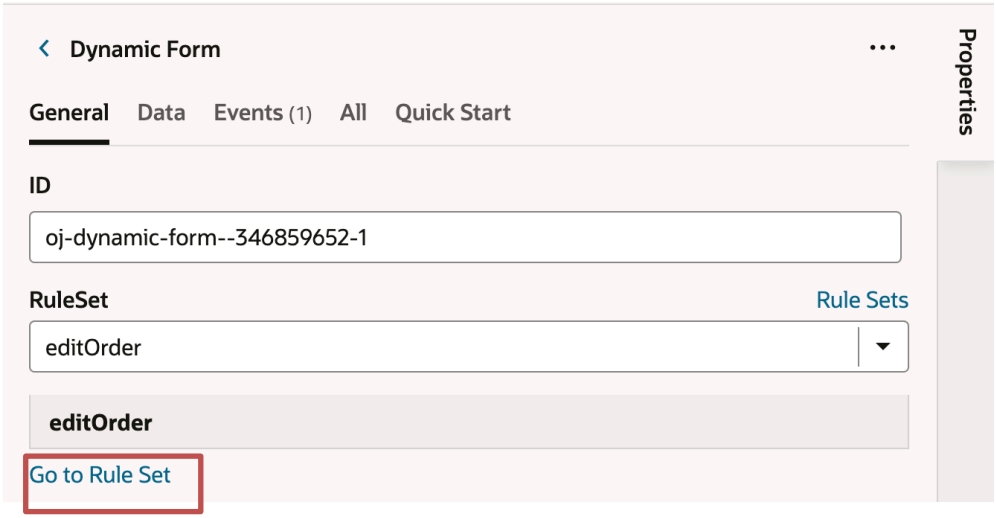
9. Map **orderId** to**Target> get_rraOrders > rraOrders_Id** (used to retrieve the record for editing) and **Target > update_rraOrders > rraOrders_Id** (used to perform the update) fields. Click **Finish**.

   Hint: If you have difficulty dragging the **orderId** to the **rraOrders_Id** fields, try making the page narrower.



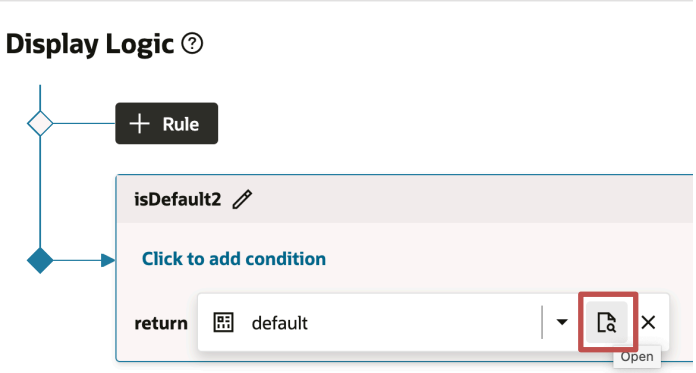
10. The three fields you selected appear on the form and you see an error. The 500 means it cannot load the data because it does not have a valid orderId. Enter **Live** mode and click the **X** to dismiss the message and return to Design mode. Click the **General** tab for the dynamic form to make some changes to the form fields. Click the **Go to Rule Set** link to enter the **Rule Set** and layout.





Note: Your ID will be different.

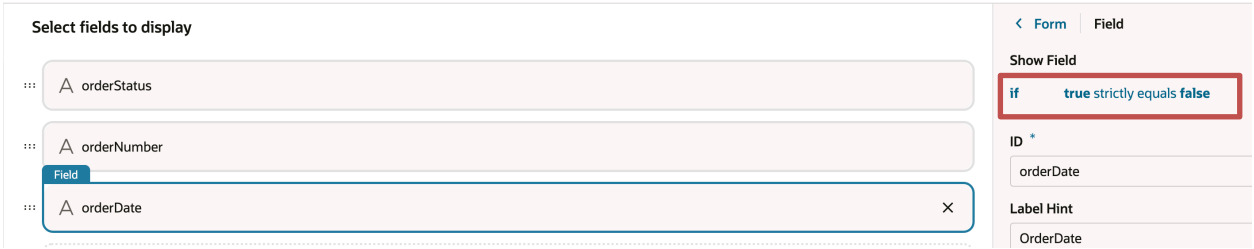
11. Click the **Open** icon for the layout.



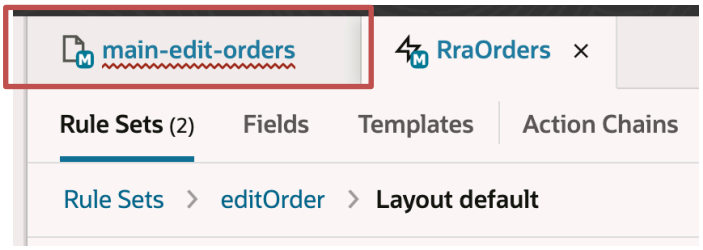
12. You included the **orderNumber** and **orderDate** fields to use them to display in the form header but you don't want them to display as part of the form itself. Select each one, one at a time, and set the rule for when the field will appear. Click **orderNumber** first. Click the **Always** link under the **Show Field** property.

13. Change the operator to **===** and the attribute to `false` and click **Save**. Because true is not equal to false, the field will not appear on the form, but it is still available to reference in another part of the form.

14. Repeat these steps with **orderDate**.

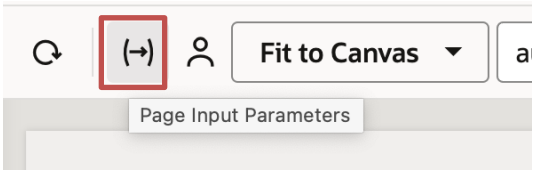


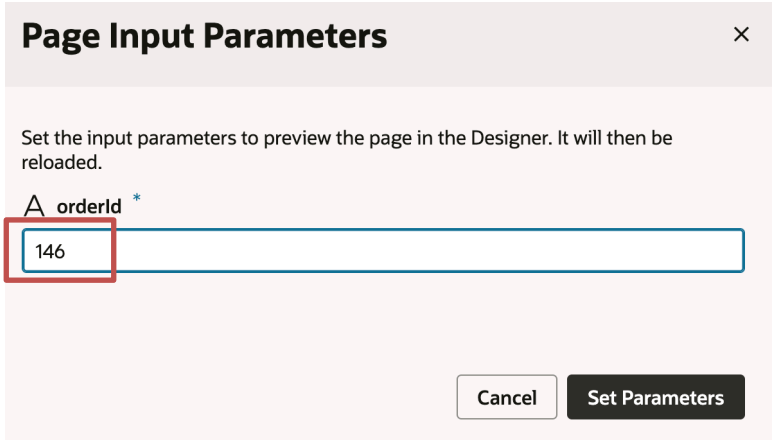
15. Click the **main-edit-orders** tab to return to the page. Dismiss the 500 error as you did before.



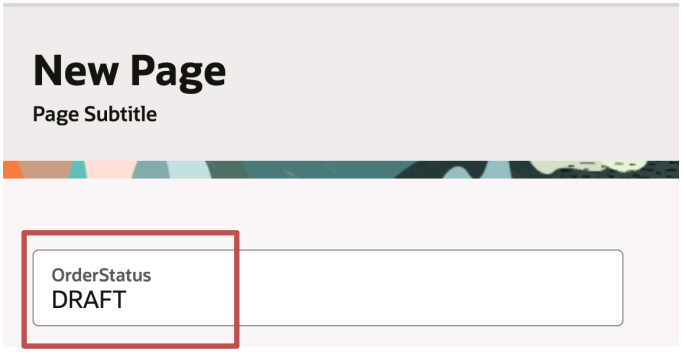
Notice only the OrderStatus field now appears.

16. Click the icon to enter Page Parameters. Enter a value of `146` for **orderId**.



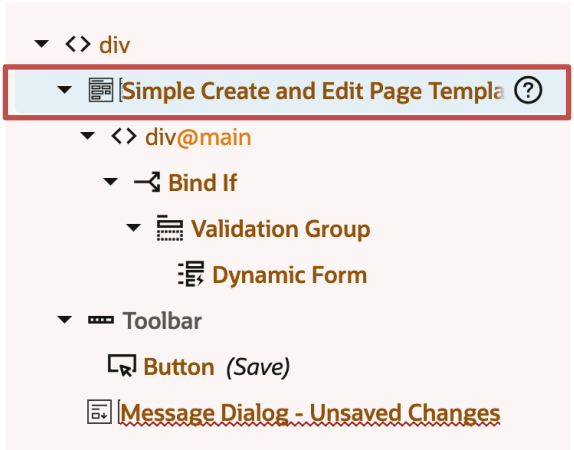


17. The input parameter causes the orderId to refresh the page and retrieve the record and you'll see the value **DRAFT** appear in the **Order Status**.

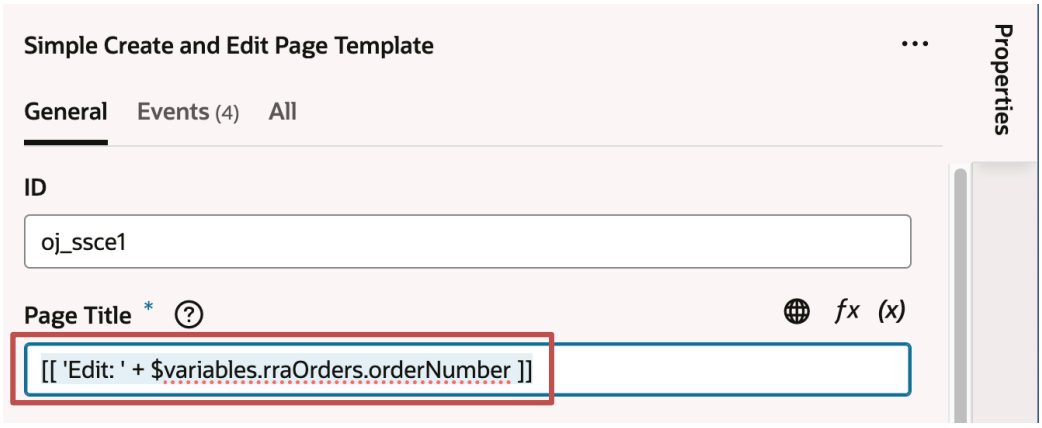


The important functionality of doing the update for an edit is performed in action chains that are pre-created. How those work is beyond the scope of his course. You'll finish by configuring the page title and subtitle to display the order number and order date.

18. Select the **Simple Create and Edit Page Template**, and then enter the following code for the **Page Title**.

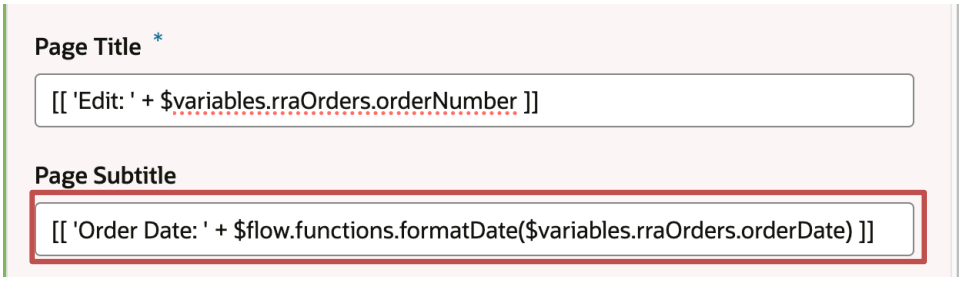



```
[[ 'Edit: ' + $variables.rraOrders.orderNumber ]]
```

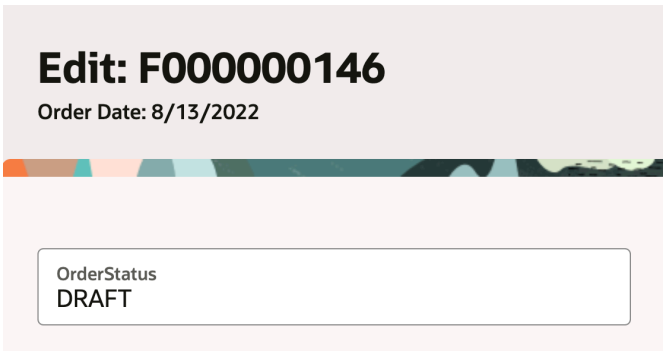


19. Enter the following code for **Subtitle**:

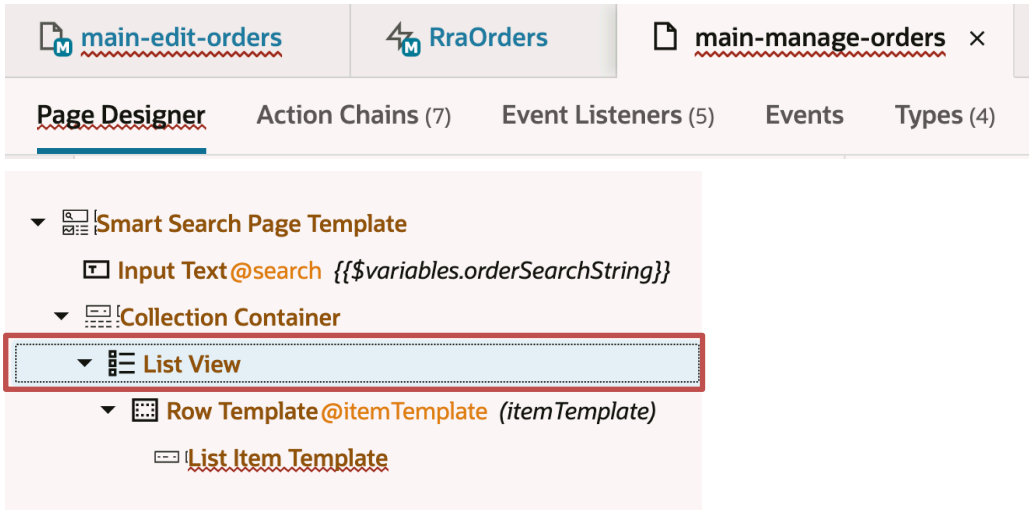
```
[[ 'Order Date: ' +
$flow.functions.formatDate($variables.rraOrders.orderDate) ]]
```

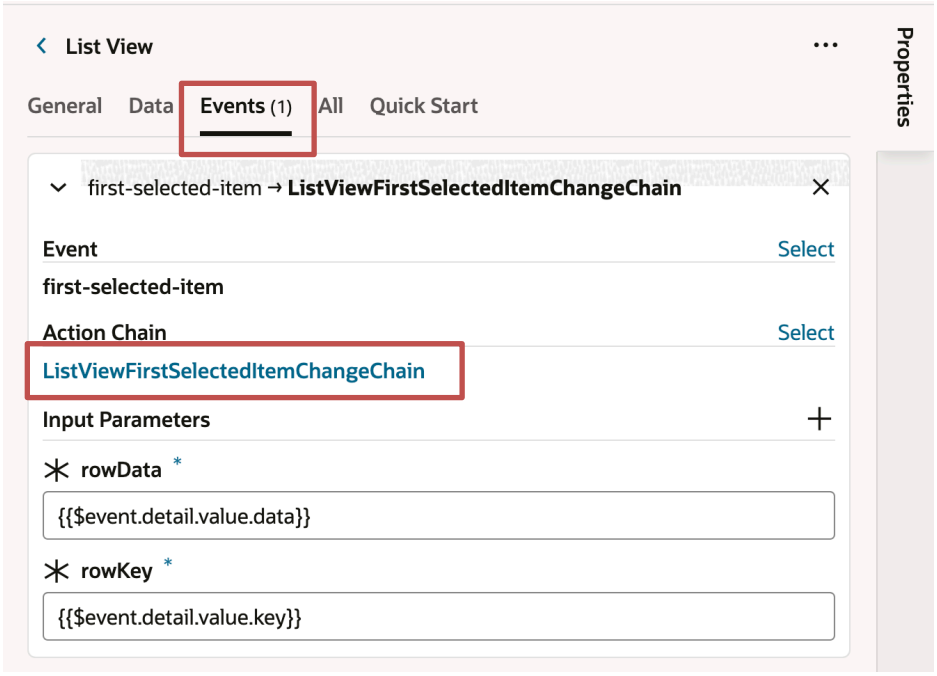


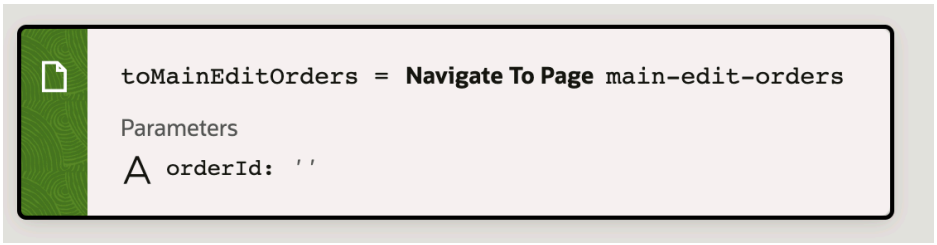
The Page Title and Subtitle change to reflect the data from the retrieved record.

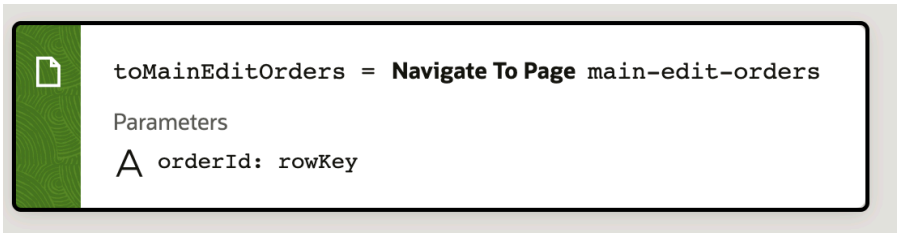


20. Lastly, open the **main-manage-orders** page, select the List View and click the link for the **ListViewFirstSelectedItemChangeChain** event under the **Events** tab. Select the **toMainEditOrders** action and map the **orderId** parameter to the ***rowKey**.









21. Return to the **Page Designer** to test that this works in **Live** mode. Select a record from the list of orders and the app will navigate to the **main-edit-orders** page and display the correct information.

    Note: In the Redwood reference application, the Advanced Create and Edit Page Template was hard-coded to support both edits and creates on the same page, depending on whether an orderId value was passed into the page or the page was entered without an orderId. If an orderId is passed in, then it is assumed to be an edit operation and the edit sections of the page are displayed. If no orderId is passed to the page, it is assumed to be a create operation and the edit sections are hidden and create sections are displayed.

    This is accomplished using many Bind If controls and it makes the page more complicated. By default, the Advanced Create and Edit Page Template and the Simple Create and Edit Page Template do not support using the same page for both create and edit operations. It is recommended to use a separate page for each operation as you've done here.

    Optional: If you're curious, you may wish to examine the action chains and events that implement the Create and Edit page functionality. Examine the events and action chains for the Save button and Primary Action – the Submit button. Select the Events tab for the Simple Create and Edit Page Template and view the events and action chains that were created automatically as part of the page template, like the spCancelChain, spSaveChain, goToParent, and spSubmit chains. One of the benefits of using a page template is it comes preconfigured with events, buttons, and other features.
22. Close all open tabs.

This completes this practice.