

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>

New Order

Page Subtitle

7/25/2024 Total Amount: \$0.00

Select (Single)

Order Details

AttachmentEntityName

BillingAddressCity

BillingAddressLine1_1

BillingAddressPostalCode

BillToAddressId

BasketFlag

BillingAddressCountry

BillingAddressLine2_1

BillingAddressState

@context/key

Required

Required

Required

Required

Required

Required

Required

Order Line Items

Products

Created By

Created

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.

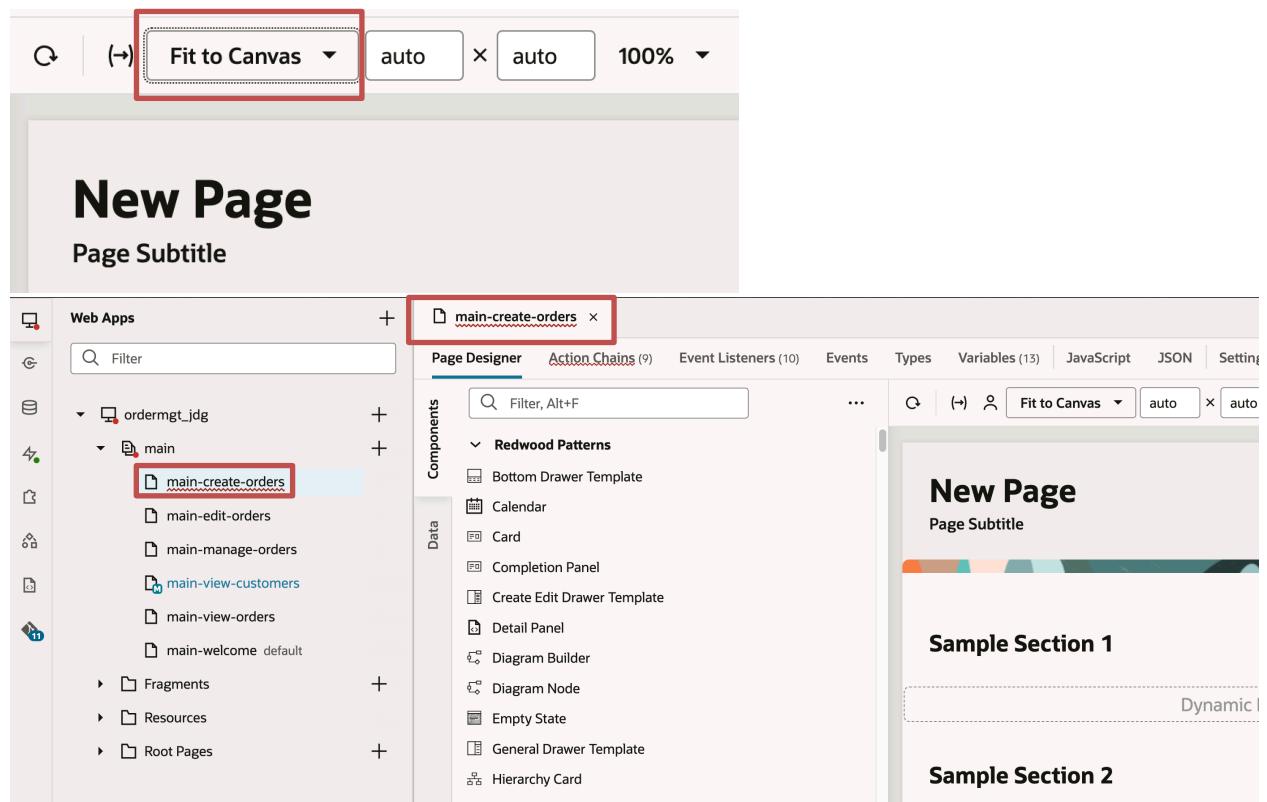
The screenshot shows the Visual Builder Studio interface. On the left is a dark sidebar with icons for Organization, Project Home, Workspaces, Git, Merge Requests, Maven, NPM, and Docker. The main area is titled "Visual Builder Studio" and "Organization". Below it is a navigation bar with tabs: Projects (which is selected), OCI Account, Build Executor Templates, Build Executors, Component Exchange, and Global. A search bar and filter buttons for Member, Favorites, Owner, Shared, All, and a magnifying glass icon are also present. The main content area shows "Projects: 3". A table lists the projects, with "OrderMgt_JDG" highlighted with a red box and labeled as "Current". The table includes columns for Name, Template, Favorite, and Status. "OrderMgt_JDG" is marked as "Active".

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

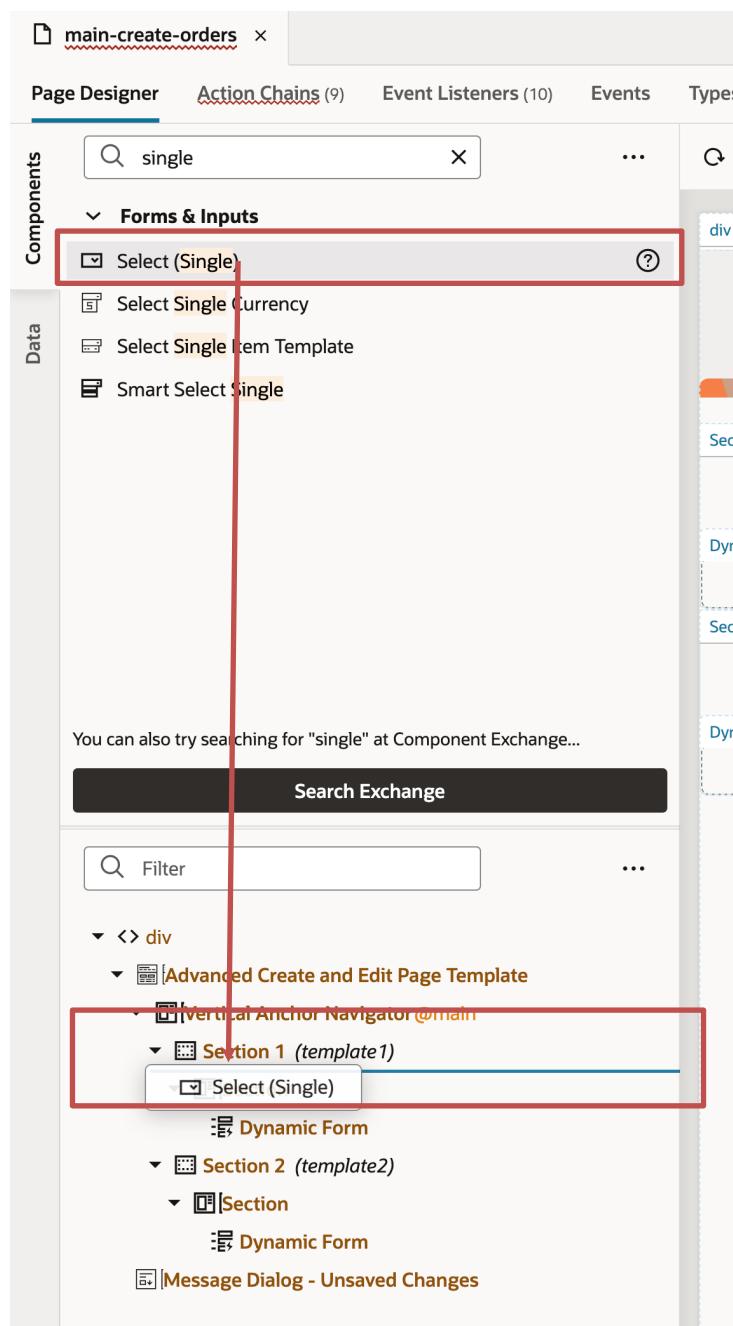
The screenshot shows the Visual Builder Studio interface with the "Workspaces" tab selected. The sidebar on the left is identical to the previous screenshot. The main area shows "OrderMgt_JDG" followed by a dropdown arrow and "Workspaces". A search bar and buttons for "Mine" and "Others" are at the top. Below is a table with columns for Workspace, Repository, and Current Branch. The first row shows "OrderMgt_JDG", "ordermgt_jdg.git", and "Order_Mgt". A context menu is open over the "OrderMgt_JDG" workspace entry, with "Open in New Tab" highlighted by a red box. Other options in the menu include Rename, Export Visual Application, Switch Environment, Change Ownership, and Delete.

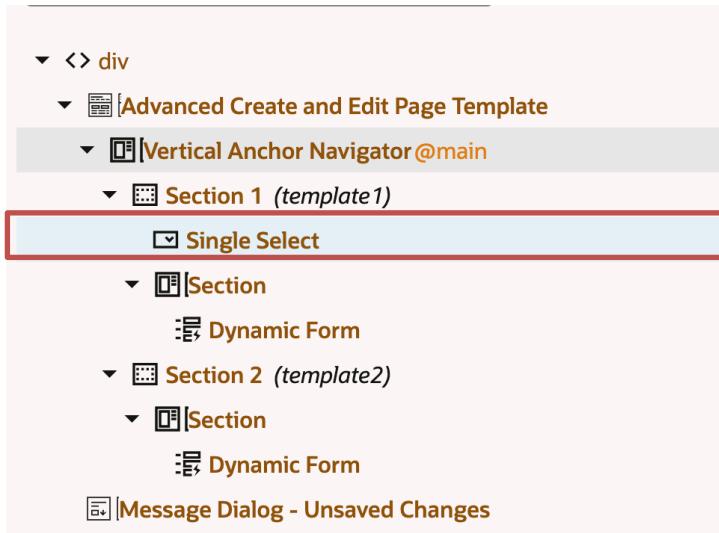
3. Close the **Getting Started** tab if it's open.
4. Select the **Web Apps** tab in the Navigator. Expand **ordermgt_<your initials>**. Expand **main**. Select the **main-create-orders** page. It opens in the Page Designer.

5. If the screen size is still set to Desktop from the previous practice, 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.





The screenshot shows the Oracle APEX Page Designer interface. On the left, the 'Components' sidebar is open, showing a search bar with 'single' and a list under 'Forms & Inputs' containing 'Select (Single)', 'Select Single Currency', 'Select Single Item Template', and 'Smart Select Single'. Below this, a message says 'You can also try searching for "single" at Component Exchange...'. On the right, the main workspace displays a page titled 'New Page' with a subtitle 'Page Subtitle'. The page content includes 'Vertical Anchor Navigator' (highlighted with a red box), 'Single Select' (selected and highlighted with a red box), 'Select (Single)' (highlighted with a red box), 'Sample Section 1' (with a 'Dynamic Form' placeholder), and 'Sample Section 2' (with a 'Dynamic Form' placeholder). A vertical anchor navigator on the left lists components: '<> div', 'Advanced Create and Edit Page Template', 'Vertical Anchor Navigator @main' (highlighted with a red box), 'Section 1 (template1)' (highlighted with a red box), 'Single Select' (selected and highlighted with a red box), 'Section', 'Dynamic Form', 'Section 2 (template2)', 'Section', 'Dynamic Form', and 'Message Dialog - Unsaved Changes'.

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

The screenshot shows the 'Single Select' component's 'Quick Start' tab. The tab navigation bar includes 'General', 'Data', 'Events', 'All', and 'Quick Start' (which is highlighted with a red box). Below the tabs, a large callout box contains the heading 'Add Options' and the sub-instruction 'Map the component to a data source, so we can populate it with data.' A 'Properties' panel is visible on the right side of the interface.

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

Choose the source of your data

▼ **Business Objects**



rraCustomers rraOrderLines rraOrders rraProducts rraSuppliers

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

Dropdown options

Label

A customerName X

Value

customerId X

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.

Styling	
<input type="checkbox"/> Medium (oj-form-control-max-width-md)	<input type="checkbox"/> Medium (oj-form-control-width-md)
<input type="checkbox"/> oj-form-control-text-align-end	<input type="checkbox"/> oj-form-control-text-align-start (oj-form-control-text-align-right)
<input type="checkbox"/> oj-form-control-text-align-start (oj-form-control-text-align-start)	<input checked="" type="checkbox"/> Small (oj-form-control-max-width-sm)
<input type="checkbox"/> Small (oj-form-control-width-sm)	

New Page

Page Subtitle



Sample Section 1

Select (Single) ▾

- Pete Chevis
- Joseph Wilke
- Azalee Goodwater
- Stefany Waninger

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

▼ <> div

▼ Advanced Create and Edit Page Template ⓘ

▼ Vertical Anchor Navigator @main

▼ Section 1 (template1)

Single Select

Advanced Create and Edit Page Template ...

General Events (4) All

ID
oj_ssce1

Page Title * ⓘ
New Order 
⚠ String not externalized for translation.

Properties

Advanced Create and Edit Page Template

New Order

Page Subtitle

Decorative horizontal bar with colored segments.

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

The screenshot shows the Oracle ADF Page Designer interface. At the top, there is a header bar with tabs: 'main-create-orders' (with a close button), 'Page Designer' (underlined in blue, indicating it is the active tab), 'Action Chains (9)', 'Event Listeners (10)', 'Events', 'Types (1)', 'Variables (14)' (which is highlighted with a red border), and 'JavaScript'. Below the header, there is a toolbar with icons for 'General', 'Events', and 'Design Time'. The main area is titled 'Variable' with a 'Properties' panel on the right. Under 'General' settings, the 'ID' field contains 'contextualInfo' and the 'Type' field is set to 'Array'. A warning message '⚠ Variable 'contextualInfo' is defined but not used' is displayed below the ID field.

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

This screenshot shows the 'Variable' creation dialog. The 'ID' field is filled with 'contextualInfo'. The 'Type' dropdown is set to 'Array'. A 'Create' button is visible next to the type selection. The 'Properties' panel on the right is currently empty.

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

This screenshot shows the 'Variable' creation dialog for 'customerName'. The 'ID' field contains 'customerName' and the 'Type' dropdown is set to 'String'. A warning message '⚠ Variable 'customerName' is defined but not used' is shown below the ID field. The 'Properties' panel is empty.

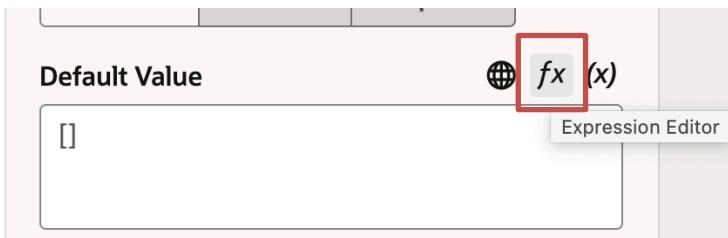
This screenshot shows the 'Variable' creation dialog for 'totalAmount'. The 'ID' field contains 'totalAmount' and the 'Type' dropdown is set to 'String'. A warning message '⚠ Variable 'totalAmount' is defined but not used' is shown below the ID field. The 'Properties' panel is empty.

```

▼ [] contextualInfo
  { } item[i]
    □ contextualInfoLabelVisible           SDP rraCustomersListSDP
    □ contextualInfoSticky
    A saveOption
    A customerName
    A 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)
  }
]

```

Expression Editor

```

1  [
2   {
3     label: 'customer',
4     value: $page.variables.customerName
5   },
6   {
7     label: 'date',
8     value: $flow.functions.formatDate( Date.now() )
9   },
10  {
11    label: 'totalAmount',
12    value: 'Total Amount: ' + $flow.functions.formatCurrency($flow.constants.defaultCurrencyCode, $page.variables.totalAmount)
13  }
14 ]
15

```

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

Advanced Create and Edit Page Template

Vertical Anchor Navigator @main

Section 1 (template1)

Variables

- Page**
- Variables**
 - avatar**
 - contextualInfo**

- 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.

Advanced Create and Edit Page Template

New Order

Page Subtitle

7/25/2024 Total Amount: \$0.00

Select (Single)

Sample Section 1

Dynamic Form

Sample Section 2

General Events (4) All

Filter

Class

ID

oj_ssce1

Advanced Create and Edit Page

Avatar Image

`{{page.variables.avatar}}`

Badge

Contextual Info

`[[$variables.contextualInfo]]`

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**.

Advanced Create and Edit Page Template

Vertical Anchor Navigator @main

Section 1 (template1)

Single Select

Dynamic Form

Dynamic Form

General Data Events (1) All Quick Start

Configure as Create Form

Configure form component as a create form

Configure as Edit Form

Configure form component as an edit form

Configure as Detail Form

Configure form component as a (read-only) detail form

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

Choose the source of your data

▼ **Business Objects**

rraCustomers rraOrderLines **rraOrders** rraProducts rraSuppliers

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

Label *

`createOrder`

ID *

`createOrder`

Description

Use Simple Layout

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.

1 2

Locate Data Select Rule Set

<p>Label *</p> <input type="text" value="createOrder"/> <p>Description</p> <input type="text"/>	<p>ID *</p> <input type="text" value="createOrder"/>
---------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------

Fields

Suggested Fields

- A attachmentEntityName
- A basketFlag
- A billingAddressCity
- A billingAddressCountry
- A billingAddressLine11
- A billingAddressLine21
- # billingAddressPostalCode
- A billingAddressState

Select fields to display [Use template instead ↶](#)

- A attachmentEntityName
- A basketFlag
- A billingAddressCity
- A billingAddressCountry
- A billingAddressLine11

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.

▼ <> div

 ▼ Advanced Create and Edit Page Template

 ▼ Vertical Anchor Navigator @main

 ▼ **Section 1 (template1)**

Single Select

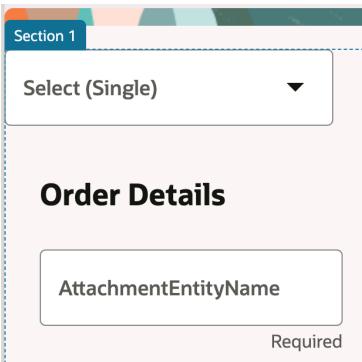
 < Section 1

ID
template1

Title
Section 1

Description
Sample Section 1

Label Hint
Order Details

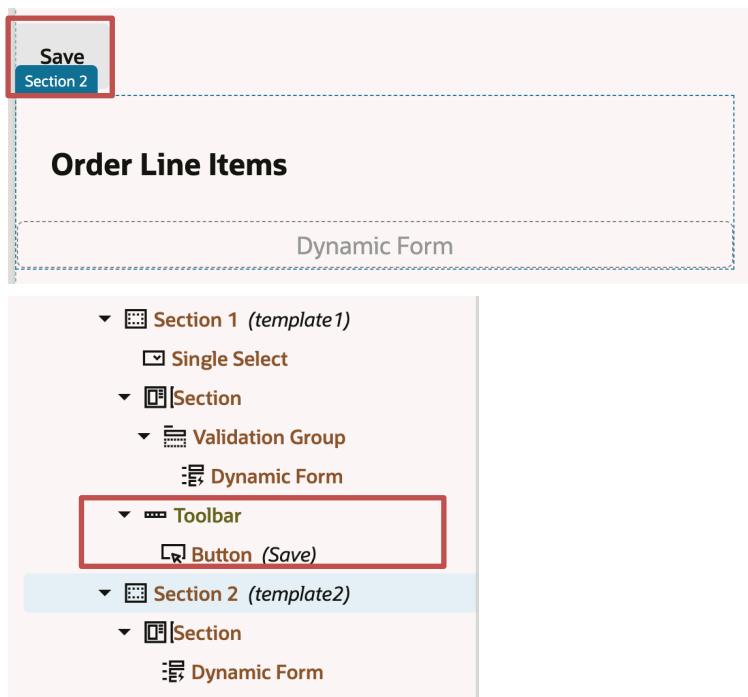


25. Select **Section 2** and change its **Label Hint** property to Order Line Items.

```
< > div
  Advanced Create and Edit Page Template
    Vertical Anchor Navigator @main
      Section 1 (template1)
        Single Select
      Section
        Validation Group
          Dynamic Form
      Toolbar
      Button (Save)
    Section 2 (template2)
      Section
        Dynamic Form
      Message Dialog - Unsaved Changes
```

Section 2	
ID	template2
Title	Section 2
Description	Sample Section 2
Label Hint	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.

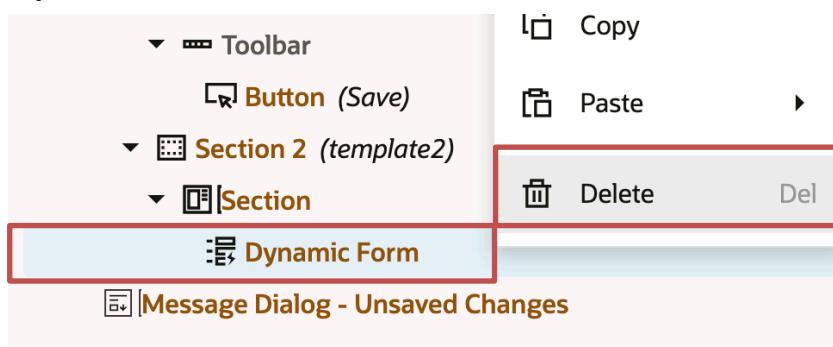


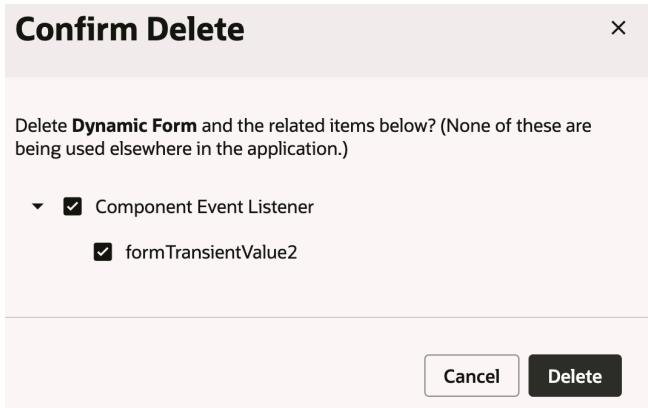
This is the default behavior, but you could combine the save of the order (master record) with its line items (detail records) and use the **Submit** button to save and commit both and then navigate back to the parent page (main-manage-orders), or use **Save** to save the order in case you still wanted 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.

main-create-orders

Page Designer Action Chains (10) Event Listeners (10) Events

Components Data

rraOrders

- Get Many
- Create
- Get One
- Update
- customerIdObject
- orderLinesCollection

Get Many

GET /rraOrders/{rraOrders_id}/child/orderLinesCollection

Business Object
getall_rraOrders-orderLinesCollection getMany
Get all

rraSuppliers

Filter

Advanced Create and Edit Page Template

Vertical Anchor Navigator @main

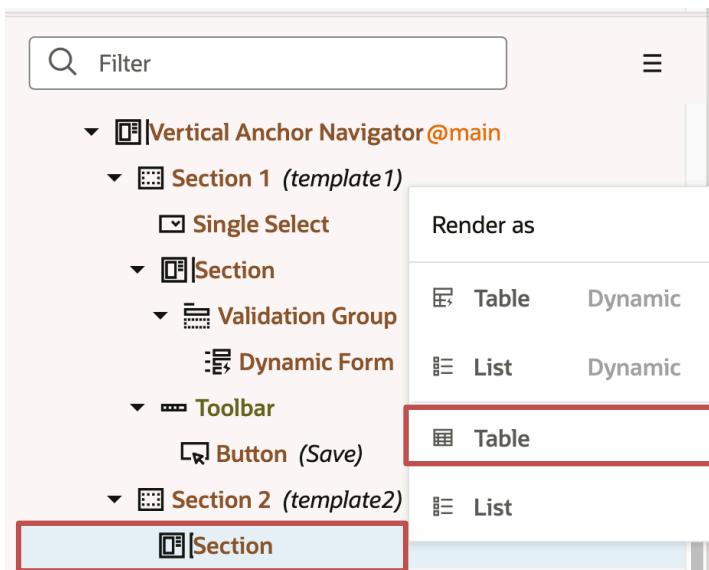
Section 1 (template1)

- Single Select
- Section
- Validation Group
- Dynamic Form
- Toolbar
- Button (Save)

Section 2 (template2)

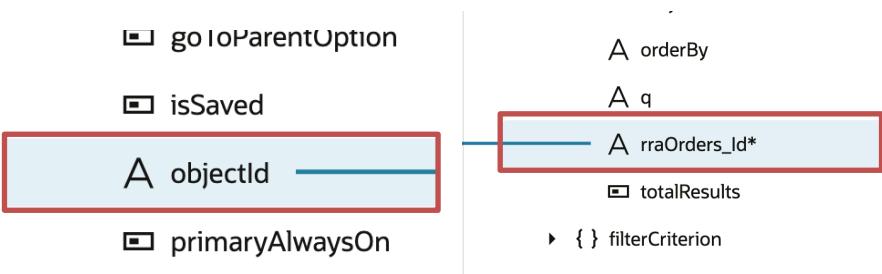
Section

Message Dialog - Unsaved Changes



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.

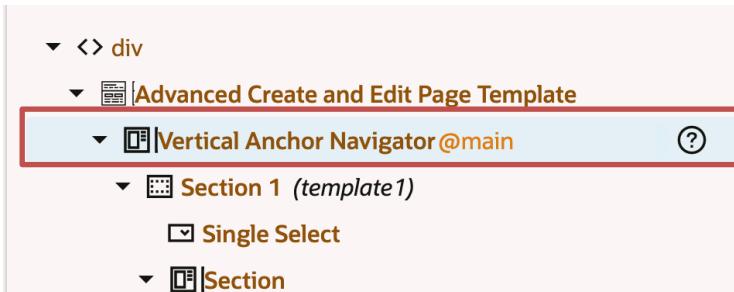
Order Line Items

ProductNumber	ProductName	ListPrice	Quantity	Amount
---------------	-------------	-----------	----------	--------

No data to display.

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**.

The screenshot shows the Oracle ADF Faces application's configuration interface. At the top, there is a navigation bar with tabs: 'Vertical Anchor Navigator', '...', 'General' (which is selected), 'Sections (2)', 'Events', and 'All'. On the right side, there is a vertical sidebar labeled 'Properties'.

In the main area, there is a section titled 'Section Fragment' with a dropdown menu. Below it is a 'Display Logic' section. Under 'Display Logic', there is a tree structure for 'Case 1'. The 'Condition' for 'Case 1' is set to 'Always Show'. Under 'Sections' for this case, 'Section 1' and 'Section 2' are listed. To the right of this tree, there is a red box highlighting a plus sign (+) button.

A modal dialog is open over the main interface, specifically for 'Case 1'. It contains the same 'Case 1' configuration with 'Always Show' condition and sections 'Section 1' and 'Section 2'. Below this, there is a button labeled '+ New Section' which is also highlighted with a red box. The background of the main interface is dimmed.

The screenshot shows two overlapping windows. The top window is titled "Create Section" and contains fields for "Title" (ProductSelector), "ID" (productSelector), and "Description". The "Title" field is highlighted with a red border. The bottom window is titled "Case" and shows a hierarchical structure under "Case 1". It includes sections for "Condition" (Always Show), "Sections" (Section 1, Section 2, ProductSelector), and a "+" button for adding more sections.

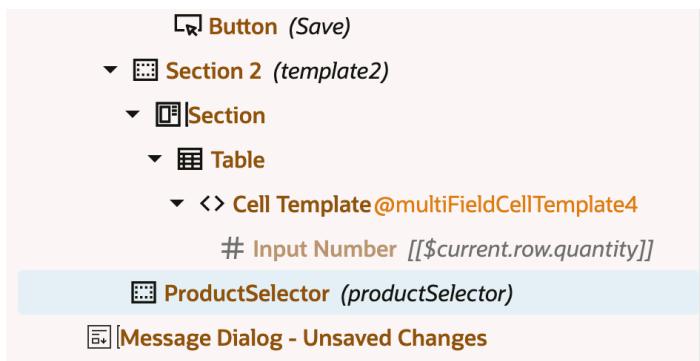
Tip: Did you notice the fragment you created earlier: **shippingaddress** was available to add as a section? This is another good use of fragments for leveraging repeatable code.

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

The screenshot shows the "Properties" panel for the "ProductSelector" section. It displays the following properties:

- ID: productSelector
- Title: ProductSelector
- Description: (empty)
- Label Hint: 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.

main-create-orders x

Page Designer Action Chains (10) Event Listeners (10) Events

Components Filter

Business Objects

- rraCustomers
- rraOrderLines
- rraOrders
- rraProducts

rraProducts

Get Many

GET /rraProducts

Service Business Object
ID, Action Hint getall_rraProducts getMany
Description Get all

Vertical Anchor Navigator @main

Section 1 (template1)

- Single Select
- Section

 - Validation Group
 - Dynamic Form
 - Toolbar
 - Button (Save)

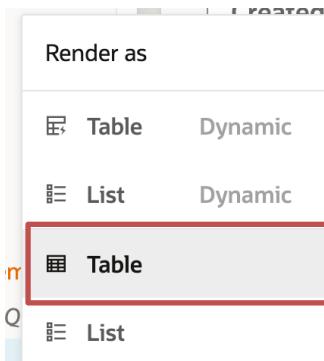
Section 2 (template2)

- Section
- Table
- Cell Template @multiFieldCellTemplate4

Input Number [[\${current.row.quantity}]]

ProductSelector (productSelector)

Message Dialog - Unsaved Changes



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.

The screenshot shows two views of the Oracle APEX application. The top view is the 'Design' mode, where a 'Jump to' menu is open, showing options like 'Order Details', 'Order Line Items', and 'Products'. The bottom view is the 'Live' mode, showing the 'New Order' page. In the 'Live' mode view, the 'Order Details' section is highlighted with a red box, and a similar 'Jump to' menu is visible on the right side of the page, mirroring the one in the Design mode.

**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.

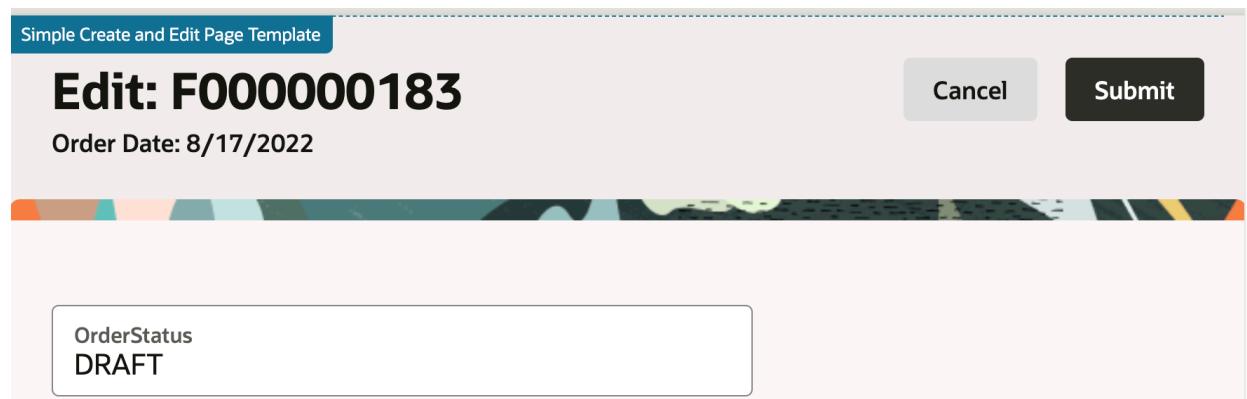
This completes this practice.

## 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.

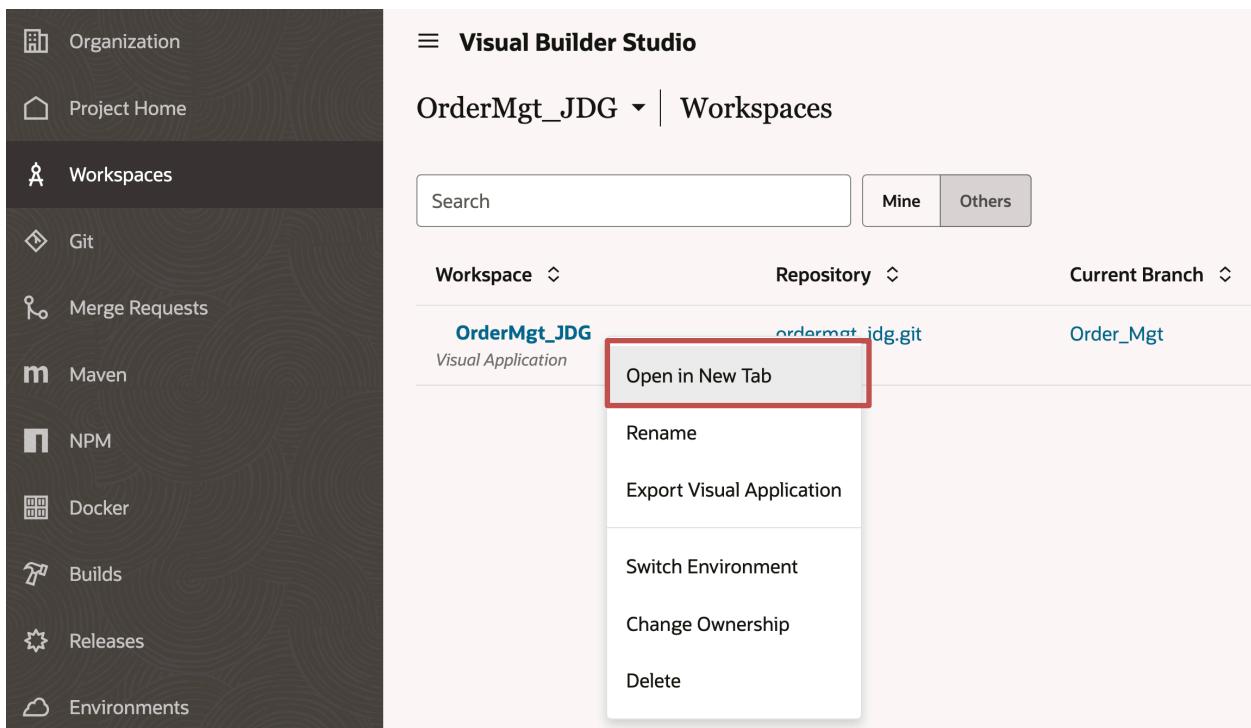
### Tasks

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

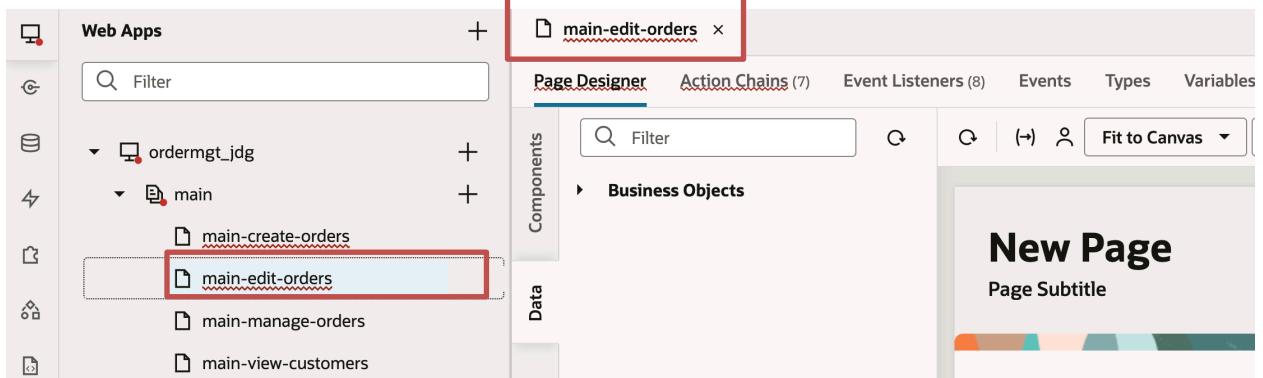
1. Log in to **Visual Builder Studio** and select your project name on the **Projects** tab. Your project name will be **OrderMgt\_< your initials >**, where **< your initials >** corresponds to your initials.

The screenshot shows the Visual Builder Studio interface. On the left is a sidebar with icons for Organization, Project Home, Workspaces, Git, Merge Requests, Maven, NPM, and Docker. The main area is titled "Visual Builder Studio" and "Organization". It has tabs for "Projects", "OCI Account", "Build Executor Templates", "Build Executors", "Component Exchange", and "Grid". Below the tabs is a search bar and filters for "Member", "Favorites", "Owner", "Shared", and "All". A "Projects: 3" section lists three projects: "OrderMgt\_JDG" (highlighted with a red box), "Order Management Applic...", and another partially visible project. The "OrderMgt\_JDG" row includes columns for "Name" (with a dropdown arrow), "Template" (with a dropdown arrow), "Favorite" (with a dropdown arrow), and "Status" (with a dropdown arrow). The "OrderMgt\_JDG" project is marked as "Current" and "Active".

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.

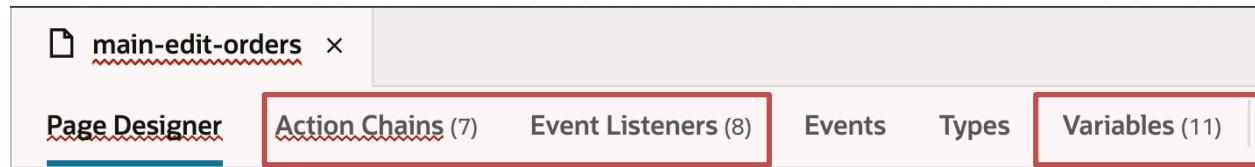


Similar to the Advanced Create and Edit Page Template, this has a dynamic form for its main section. This is where you choose the fields that will be included for edit. Notice the page comes with Cancel and Submit buttons and will have a Save button as well. The intent is that the Save button would allow 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**.

A screenshot of the Oracle ADF Dynamic Form configuration dialog. The title bar says 'Dynamic Form'. Below it, there are tabs: 'General', 'Data', 'Events (1)', 'All', and 'Quick Start' (which is underlined in black). Under 'Quick Start', there are three options: 'Configure as Create Form' (with a note 'Configure form component as a create form'), 'Configure as Edit Form' (with a note 'Configure form component as an edit form' and this entire section is highlighted with a red box), and 'Configure as Detail Form' (with a note 'Configure form component as a (read-only) detail 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.

## Choose the source of your data

### Business Objects



rraCustomers



rraOrderLines



rraOrders



rraProducts



rraSuppliers

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

Label \*

 ID \*

Description

Use Simple Layout

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**.

# objectVersionNumber

Field

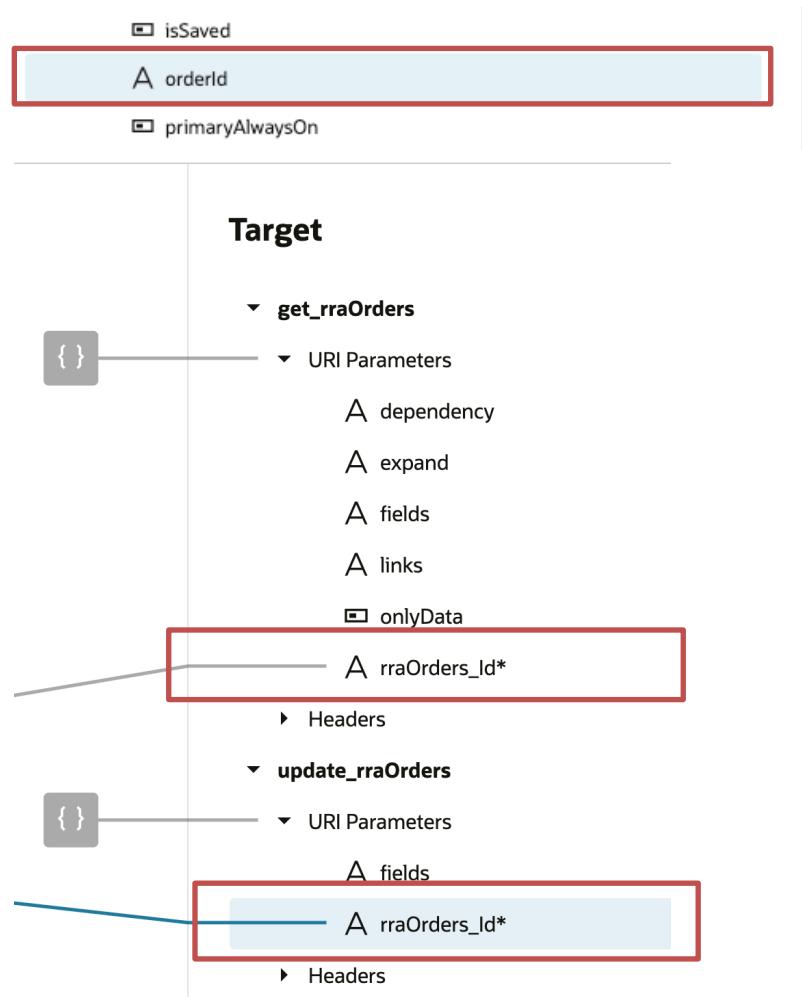
# objectVersionNumber1

Select fields to display [Use template instead](#) ↶

- ... A orderStatus
- ... A orderNumber
- ... A orderDate

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.



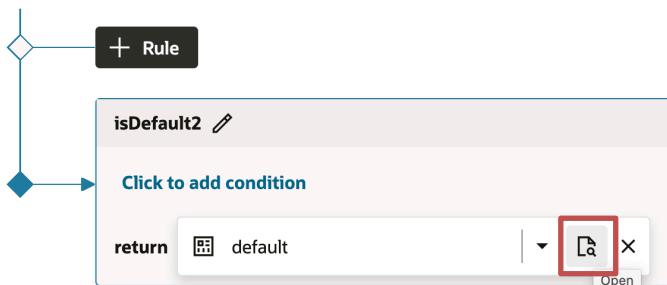
- The three fields you selected appear on the form and you see an error. The 500 means it cannot load the data because it does not have a valid orderId. Enter **Live** mode and click the **X** to dismiss the message and return to Design mode. Click the **General** tab for the dynamic form to make some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

The screenshot shows the Oracle Forms interface. At the top, there are tabs for 'Live', 'Design', and 'Code'. The 'Live' tab is active. Below the tabs, a message box displays an error: 'Could not load data: status 500'. In the center, a window titled 'New Page' is shown with a 'Page Subtitle'. On the right, there are 'Cancel' and 'Submit' buttons. A red box highlights the close button ('X') in the top right corner of the message box. Below the window, the 'Dynamic Form' properties panel is visible. It has tabs for 'General', 'Data', 'Events (1)', 'All', and 'Quick Start', with 'General' selected. The 'General' tab includes fields for 'ID' (containing 'oj-dynamic-form--346859652-1') and 'RuleSet' (containing 'editOrder'). To the right of the 'RuleSet' field is a 'Rule Sets' dropdown. A red box highlights the 'Go to Rule Set' link under the 'editOrder' section. A vertical 'Properties' sidebar is on the right.

**Note:** Your ID will be different.

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

#### Display Logic ?



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.

The screenshot shows the Oracle Forms builder interface. On the left, the 'Select fields to display' panel lists three fields: 'orderStatus', 'Field' (selected), and 'orderDate'. On the right, the 'Properties' panel shows the 'Show Field' property set to 'Always'. A red box highlights the 'orderNumber' field in the list.

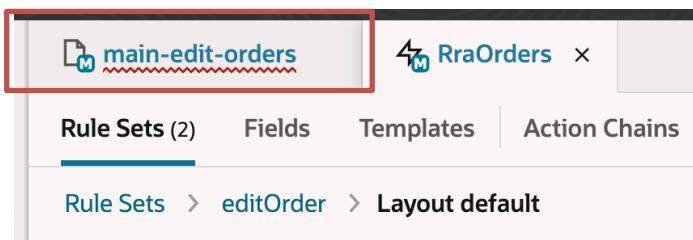
13. Change the operator to **==** and the attribute to **false** and click **Save**. Because true is not equal to false, the field will not appear on the form, but it is still available to reference in another part of the form.

The screenshot shows the 'Edit Show Field Condition' dialog box. It contains a single condition: 'if true == false'. The 'true' and 'false' fields are highlighted with a red box.

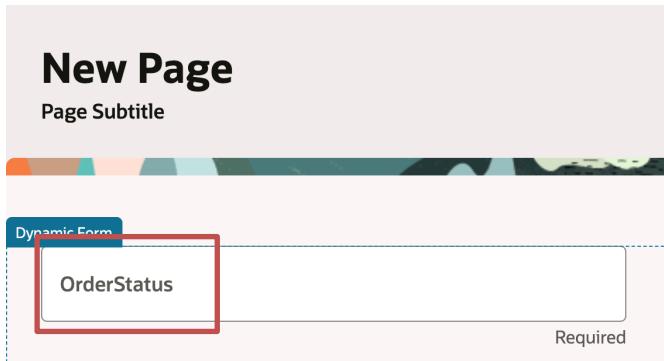
14. Repeat these steps with **orderDate**.

The screenshots show the Oracle Forms builder interface. The top screenshot shows the 'orderDate' field selected with its 'Show Field' property set to 'Always'. The bottom screenshot shows the 'orderDate' field selected with its 'Show Field' property set to a condition: 'if true strictly equals false'.

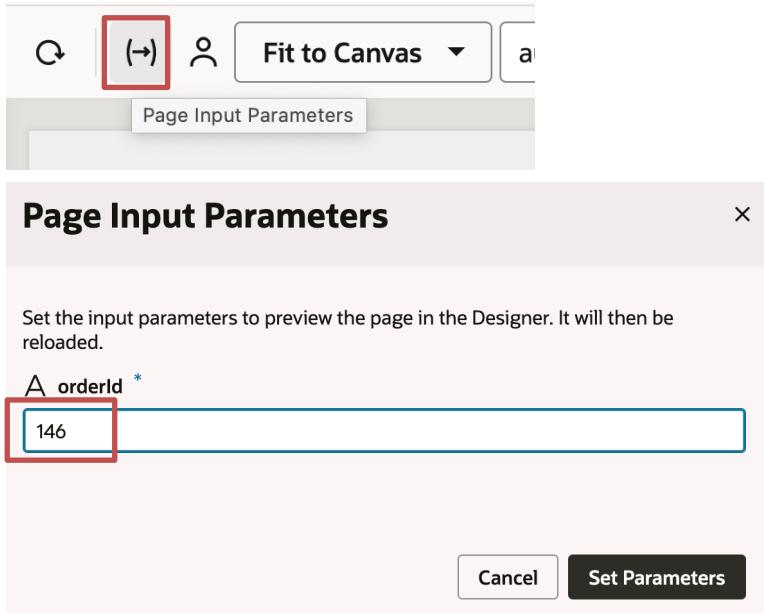
15. Click the **main-edit-orders** tab to return to the page. Dismiss the 500 error as you did before.



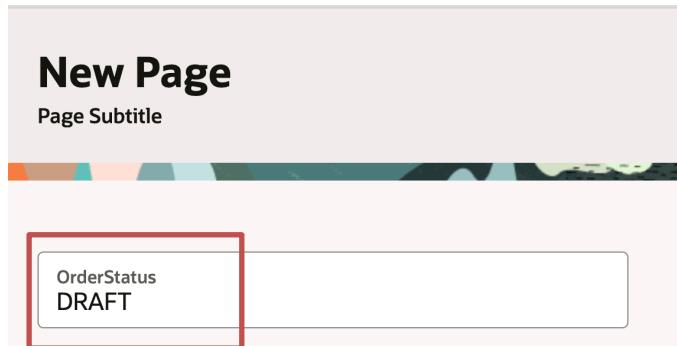
Notice only the OrderStatus field now appears.



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]]
```

| Simple Create and Edit Page Template |                                                                                                                                                     | ... | Properties |
|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----|------------|
| <b>General</b>                       | Events (4)                                                                                                                                          | All |            |
| <b>ID</b>                            | <input type="text" value="oj_ssce1"/>                                                                                                               |     |            |
| <b>Page Title *</b>                  | <input style="border: 2px solid red;" type="text" value="[[ 'Edit: ' + \$variables.rraOrders.orderNumber ]]"/> <span>(?)</span> <span>fx (x)</span> |     |            |

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

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

Page Title \*

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

Page Subtitle

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

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

# Edit: F000000146

Order Date: 8/13/2022

OrderStatus  
DRAFT

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**.

main-edit-orders RraOrders main-manage-orders

Page Designer Action Chains (7) Event Listeners (5) Events Types (4)

- Smart Search Page Template
  - Input Text @search {{\$variables.orderSearchString}}
  - Collection Container
    - List View
      - Row Template @itemTemplate (itemTemplate)
        - List Item Template

**List View**

General Data **Events (1)** All Quick Start

first-selected-item → **ListViewFirstSelectedItemChangeChain**

**Event** Select

first-selected-item

**Action Chain** Select

**ListViewFirstSelectedItemChangeChain**

**Input Parameters** +

\* rowData \*  
{{ \$event.detail.value.data }}

\* rowKey \*  
{{ \$event.detail.value.key }}

**toMainEditOrders = Navigate To Page main-edit-orders**

Parameters

A orderId: ''

**Page \*** Create

main-edit-orders

Go to Page

**Parameters**

A orderId \*  
[[ rowKey ]]

**toMainEditOrders = Navigate To Page main-edit-orders**

Parameters

A orderId: 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.