Product Selector

V2.8

for **Sales Cloud**

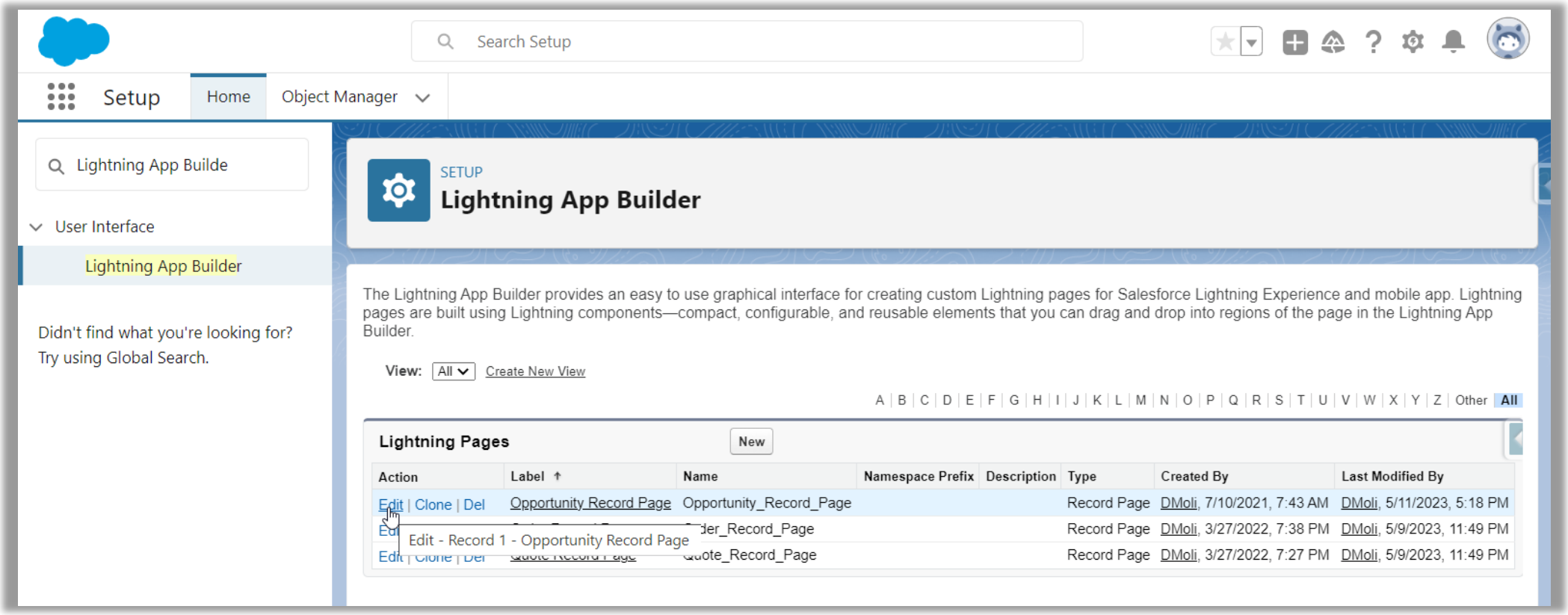
Post Install Instructions

Index

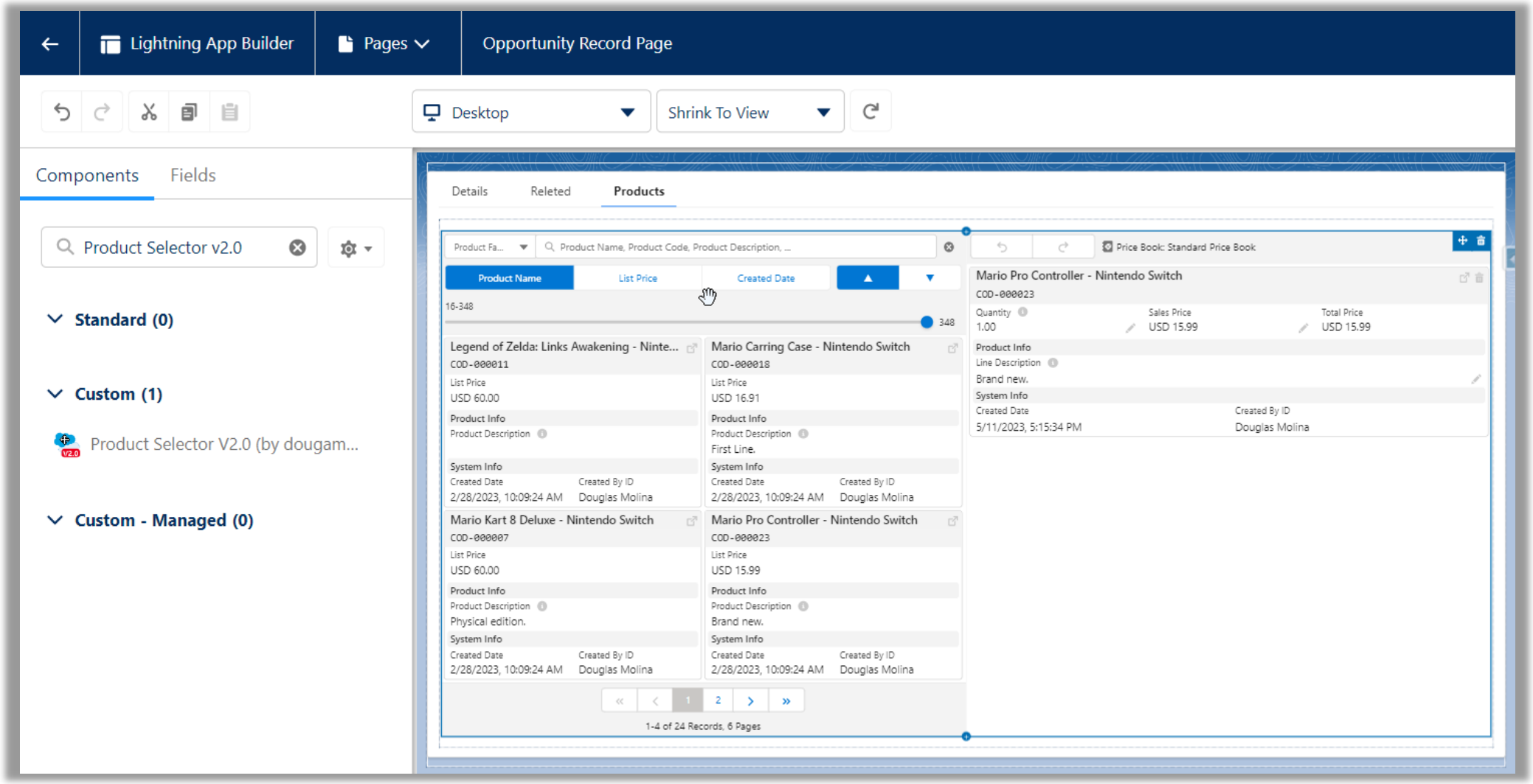
Adding the Component to the Opportunity, Quote or Order Record Page Layout 3
Customizing the Component by Changing its Attribute Values 5
Creating the Product Detail Section 8
Creating the Field Mapping for the Product Import Tool 16
Using the Component Inside Lightning Communities 19

Adding the Component to the Opportunity, Quote or Order Record Page Layout

Go to Setup >> Lightning App Builder and select the Opportunity, Quote or Order Record Page you want to add the component into, and click Edit.

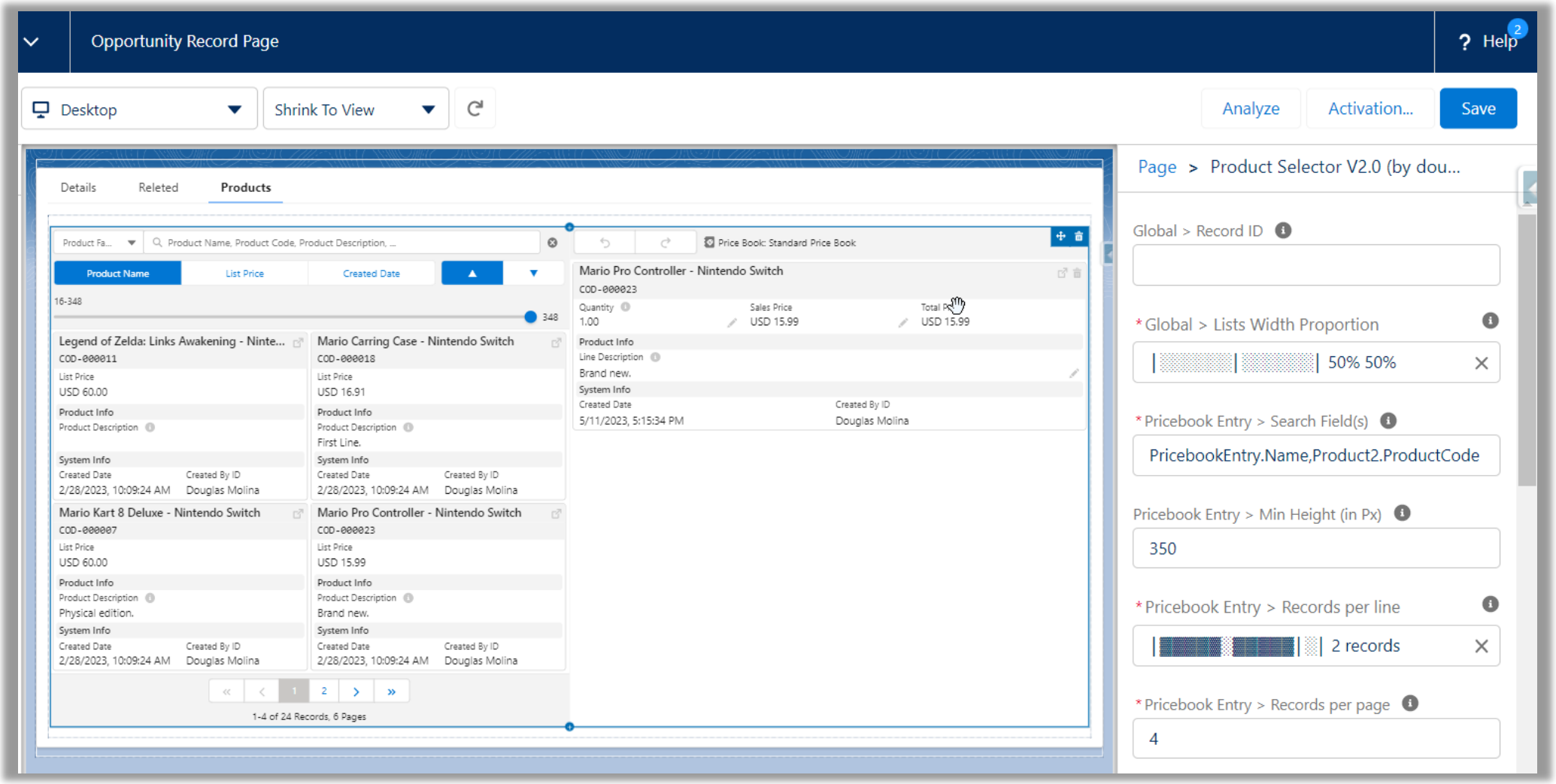


Once there, chose a place on the page of a reasonable size, on the Components Search field search for **Product Selector v2.8**, and drag and drop it to the chosen place. The component will show up in its default configuration.

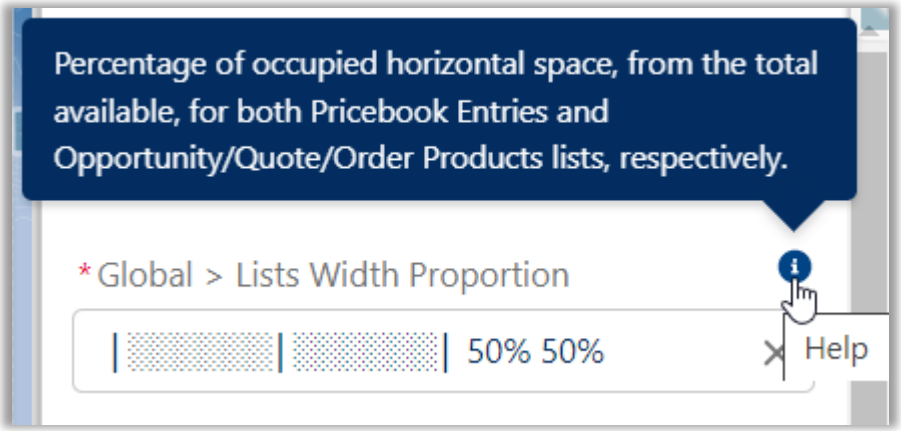


Customizing the Component by Changing its Attribute Values

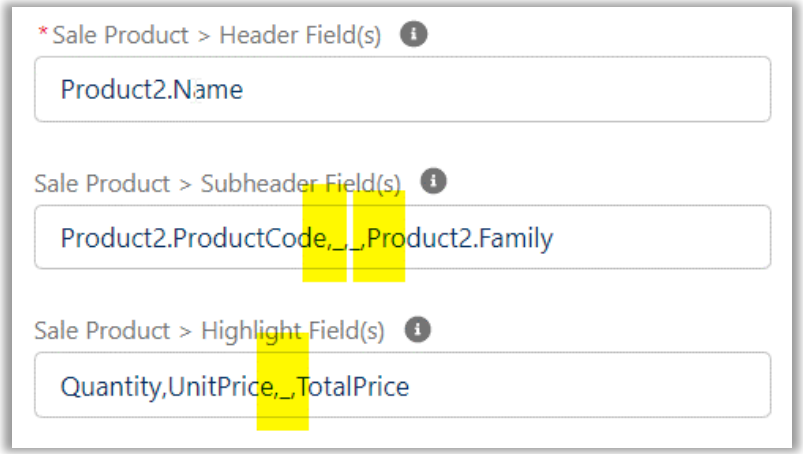
You can customize the component by changing its attribute values, located on the Attributes panel of the App Builder.

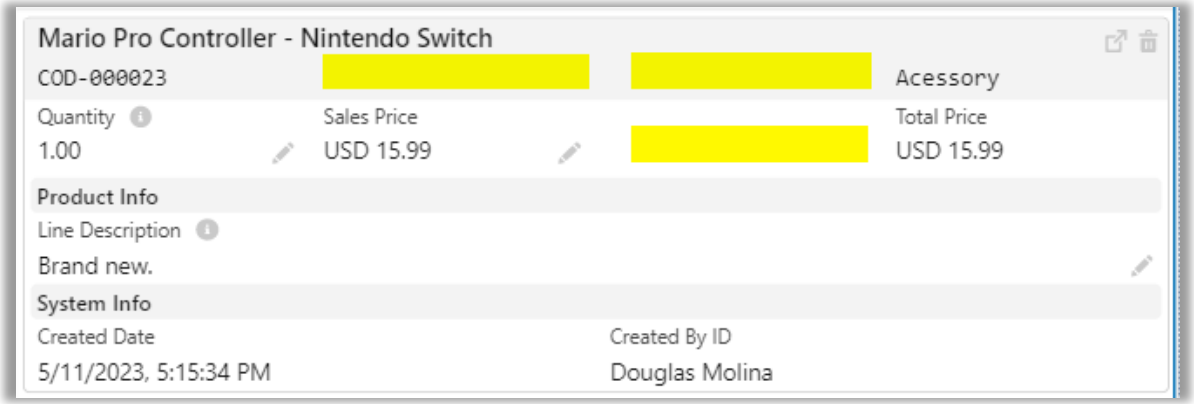


Every attribute brings an explanation about itself. To see it, hover the mouse cursor over the “i” icon.

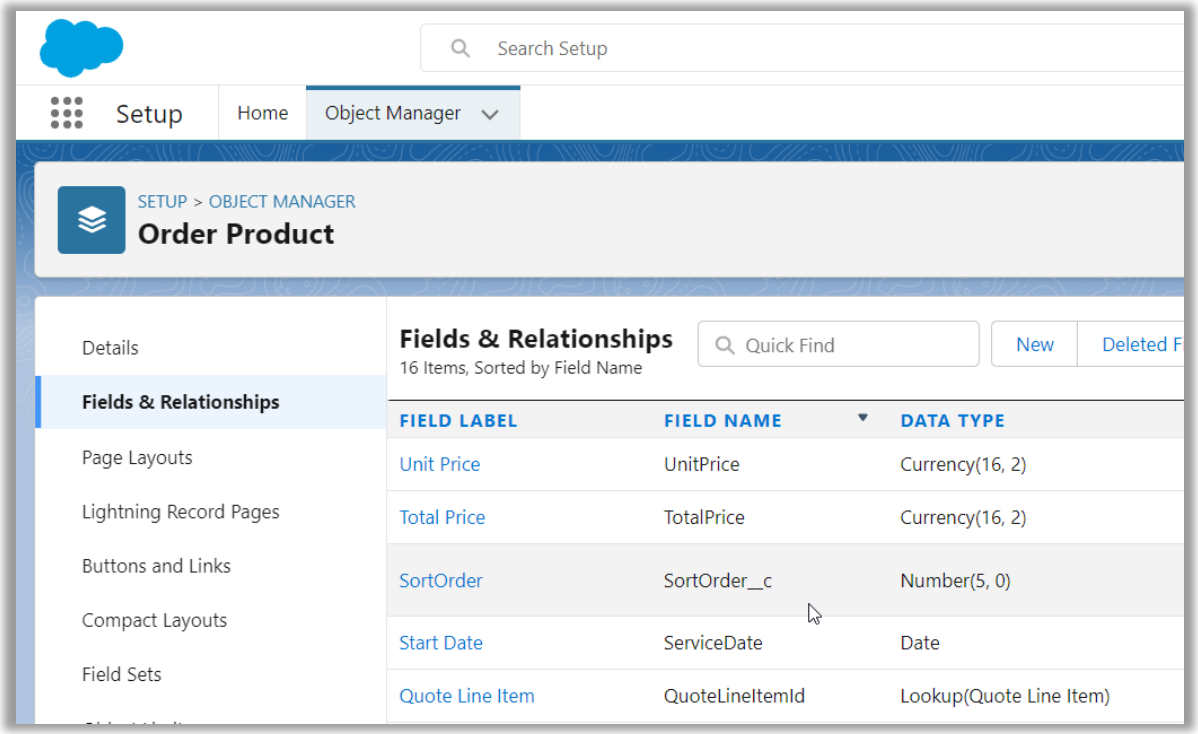


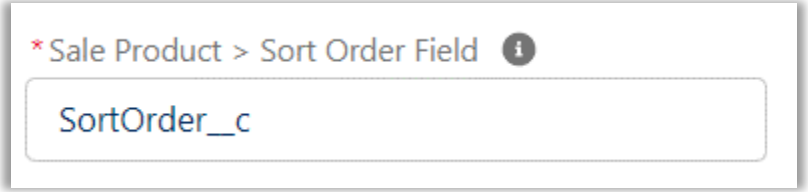
For attributes that asks for field lists, you can use empty strings or underscores between commas to create spaces between other fields. That helps keep things aligned.





When adding the component into an Order layout, you should create a custom number field in the Order Product object, and write its API Name into the **Sale Product > Sort Order Field** attribute, replacing its default value. This step is not necessary when working with Opportunities or Quotes.





Creating the Product Detail Section

There are **2 attributes** responsible to determine the fields and layout of the **details section**, for both Pricebook Products and Opportunity/Quote/Order Products, they are the **Pricebook Entry > Detail Field(s)** and **Sale Product > Detail Field(s)** respectively.

They are more flexible in terms of how you can specify the fields and layout. There are 2 ways you can do it:

1st Way: Using Custom-Made Layout.

Advantages of it:

- You can create layouts with several fields in each row, as opposed to just two of the standard layouts.
- You don't need to create any page layout or other kind of metadata in your Org, just fill this attribute.
- You can generate this value dynamically, for example if you're using the component inside a Screen Flow.
- You don't need to create lots of formula fields if you want to specify fields from more than 1 object.

2nd Way: Using Existing Page Layout.

Advantages of it:

- It's simpler to write.

You **cannot** mix both ways at the same time inside a single attribute, but you **can** use one way in one attribute, and other way on the other attribute, fer example:

Pricebook Entry > Detail Field(s) -> **1st Way** Custom-Made Layout

Sale Product > Detail Field(s) -> **2nd Way** Existing Page Layout

Using the **1st Way**: Custom-Made Layout

It's basically a string made of several other concatenated strings, containing the fields, rows, section labels, layouts, and record type assignments, **all of them being optional**. The idea here is similar to the standard record layout + record type + their assignment.

Suppose a company that sells 2 types of products: **Hardware** and **Software**, so they created 2 record types for the Product Object: **Hardware Product** and **Software Product**, and several fields specific to each type of product.

You would like to show a different layout for each type of product when using this component, with those specific fields. Furthermore, you'd also like to separate all this info into labeled sections containing several rows each, with a different numbers of fields per row.

The way to solve this is to write down the layout you're thinking about just like the example below, where the first level represents the **record type names**, the second level represents the **section labels**, and the third level represents the **rows**:

```
Software Product
    Software Specs
        Product2.Soft_ver__c    Product2.Soft_size__c
        Product2.Soft_AddList__c
    Additional Info
        Product2.Description
    System Info
        Product2.CreatedDate    Product2.CreatedById
Hardware Product
    Hardware Config
        Product2.Ram_Amount__c  Product2.Clock_speed__c Product2.Board_ver__c
    Gallery
        Product2.Pic_1__c       Product2.Pic_2__c       Product2.Pic_3__c
    Additional Info
        Product2.Description
```

That being done, you need to join all this structure into a single line of text, concatenating the parts by using the following logic:

1) Place a **,** character between the field names, to create a **row**, like this:

```
Software Product
    Software Specs
        Product2.Soft_ver__c,Product2.Soft_size__c
        Product2.Soft_AddList__c
    Additional Info
        Product2.Description
    System Info
        Product2.CreatedDate,Product2.CreatedById
Hardware Product
    Hardware Config
        Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c
    Gallery
        Product2.Pic_1__c,Product2.Pic_2__c,Product2.Pic_3__c
    Additional Info
        Product2.Description
```

2) Place a **;** character between the rows, to create a **section content**, like this:

```
Software Product
    Software Specs
        Product2.Soft_ver__c,Product2.Soft_size__c;Product2.Soft_AddList__c
    Additional Info
        Product2.Description
    System Info
        Product2.CreatedDate,Product2.CreatedById
Hardware Product
    Hardware Config
        Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c
    Gallery
        Product2.Pic_1__c,Product2.Pic_2__c,Product2.Pic_3__c
    Additional Info
        Product2.Description
```

3) Place a **:** character between a section label and its section content, to create a **full section**, like this:

```
Software Product
    Software Specs:Product2.Soft_ver__c,Product2.Soft_size__c;Product2.Soft_AddList__c
    Additional Info:Product2.Description
    System Info:Product2.CreatedDate,Product2.CreatedById
Hardware Product
    Hardware Config:Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c
    Gallery:Product2.Pic_1__c,Product2.Pic_2__c,Product2.Pic_3__c
    Additional Info:Product2.Description
```

4) Place a **/** character between the full sections, to create a **layout** like this:

```
Software Product
    Software Specs:Product2.Soft_ver__c,Product2.Soft_size__c;Product2.Soft_AddList__c/Additional
    Info:Product2.Description/System Info:Product2.CreatedDate,Product2.CreatedById
Hardware Product
    Hardware
    Config:Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c/Gallery:Product2.Pic_1__c
    ,Product2.Pic_2__c,Product2.Pic_3__c/Additional Info:Product2.Description
```

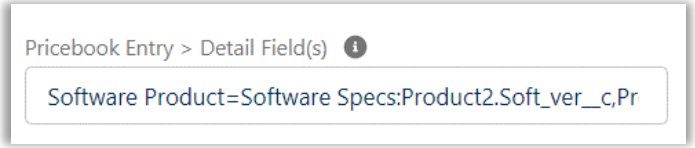
5) Place a **=** character between a record type name and its layout, to create an **assignment**, like this:

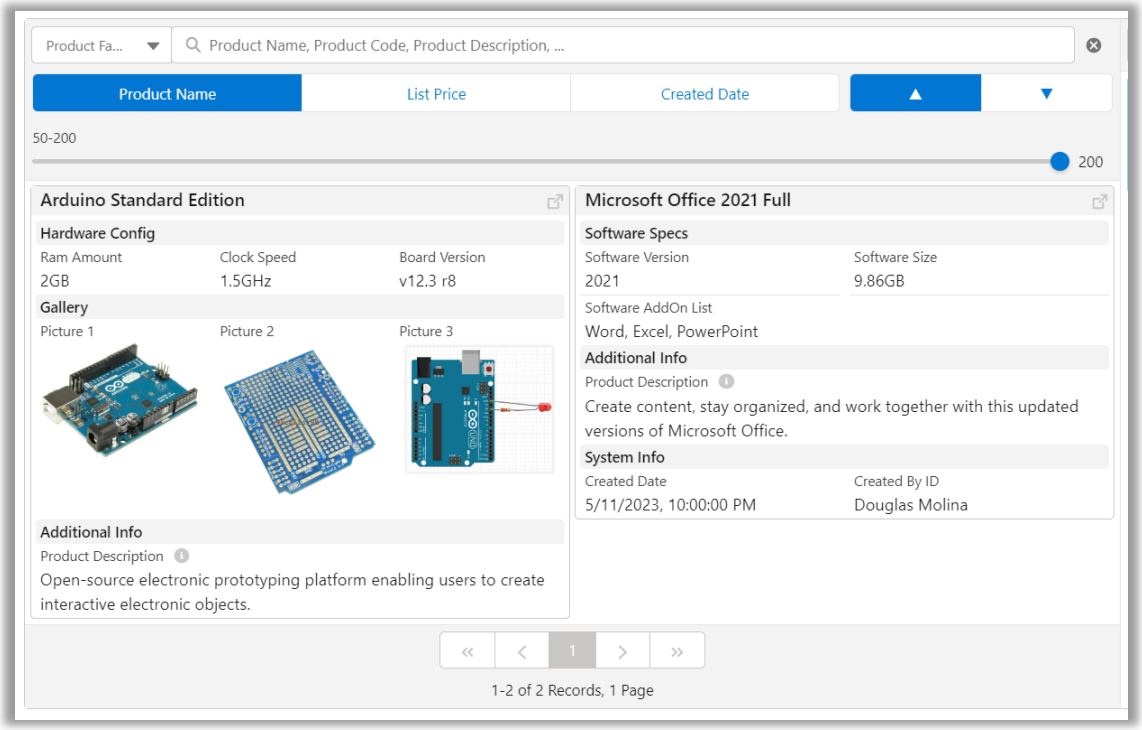
```
Software Product=Software
Specs:Product2.Soft_ver__c,Product2.Soft_size__c;Product2.Soft_AddList__c/Additional
Info:Product2.Description/System Info:Product2.CreatedDate,Product2.CreatedById
Hardware Product=Hardware
Config:Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c/Gallery:Product2.Pic_1__c,Pro
duct2.Pic_2__c,Product2.Pic_3__c/Additional Info:Product2.Description
```

6) Place a | character between the assignments, to create the **final attribute value**, like this:

```
Software Product=Software
Specs:Product2.Soft_ver__c,Product2.Soft_size__c;Product2.Soft_AddList__c/Additional
Info:Product2.Description/System Info:Product2.CreatedDate,Product2.CreatedById|Hardware
Product=Hardware
Config:Product2.Ram_Amount__c,Product2.Clock_speed__c,Product2.Board_ver__c/Gallery:Product2.Pic_1__c,Pro
duct2.Pic_2__c,Product2.Pic_3__c/Additional Info:Product2.Description
```

That being done, just copy and paste this string into one of the details attributes of the component, and you should get results like this:





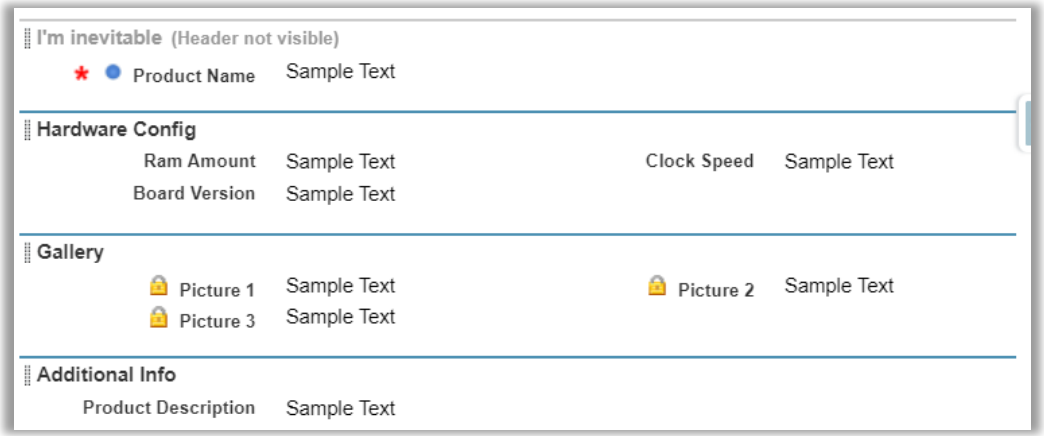
You can skip any of those components (ex: assignments, record types, section labels, rows) and apply only the concatenations that actually make sense to the remaining components.

Using the **2nd Way**: Existing Page Layout

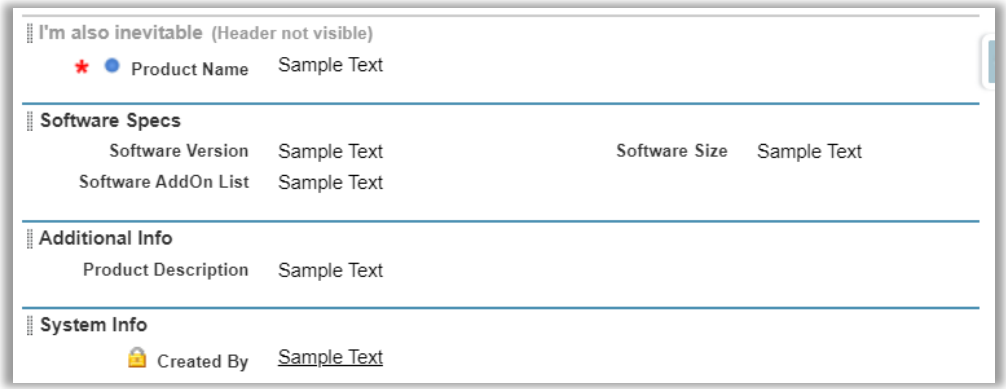
Similar to the 1st way but simpler and more limited. Instead of creating a custom-made layout structure, you just reference an existing Page Layout.

Still using the previous example of Hardware and Software products, but this time you already have 2 Product Page Layouts created:

Hardware Product Layout



Software Product Layout



You would like to show these different Page Layouts for each type of product when using this component.

The way to solve this is to write down the layout names just like the example below, where the first level is always the string "[pl]" that just tells the component that you're using the 2nd Way, the second level represents the **record type names**, and the third level represents the **Page Layout Full Names** (Object name + a minus sign + the Page Layout Name):

```
[pl]
    Software Product
        Product2-Software Product Layout
    Hardware Product
        Product2-Hardware Product Layout
```

That being done, you need to join all this structure into a single line of text, concatenating the parts by using the following logic:

1) Place a **=** character between a record type name and its page layout full name, to create an **assignment**, like this:

```
[pl]
    Software Product=Product2-Software Product Layout
    Hardware Product=Product2-Hardware Product Layout
```

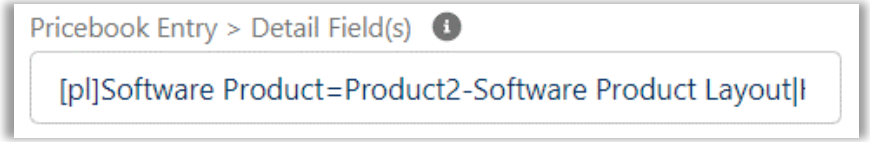
2) Place a **|** character between the assignments, to create the **pre-final value**, like this:

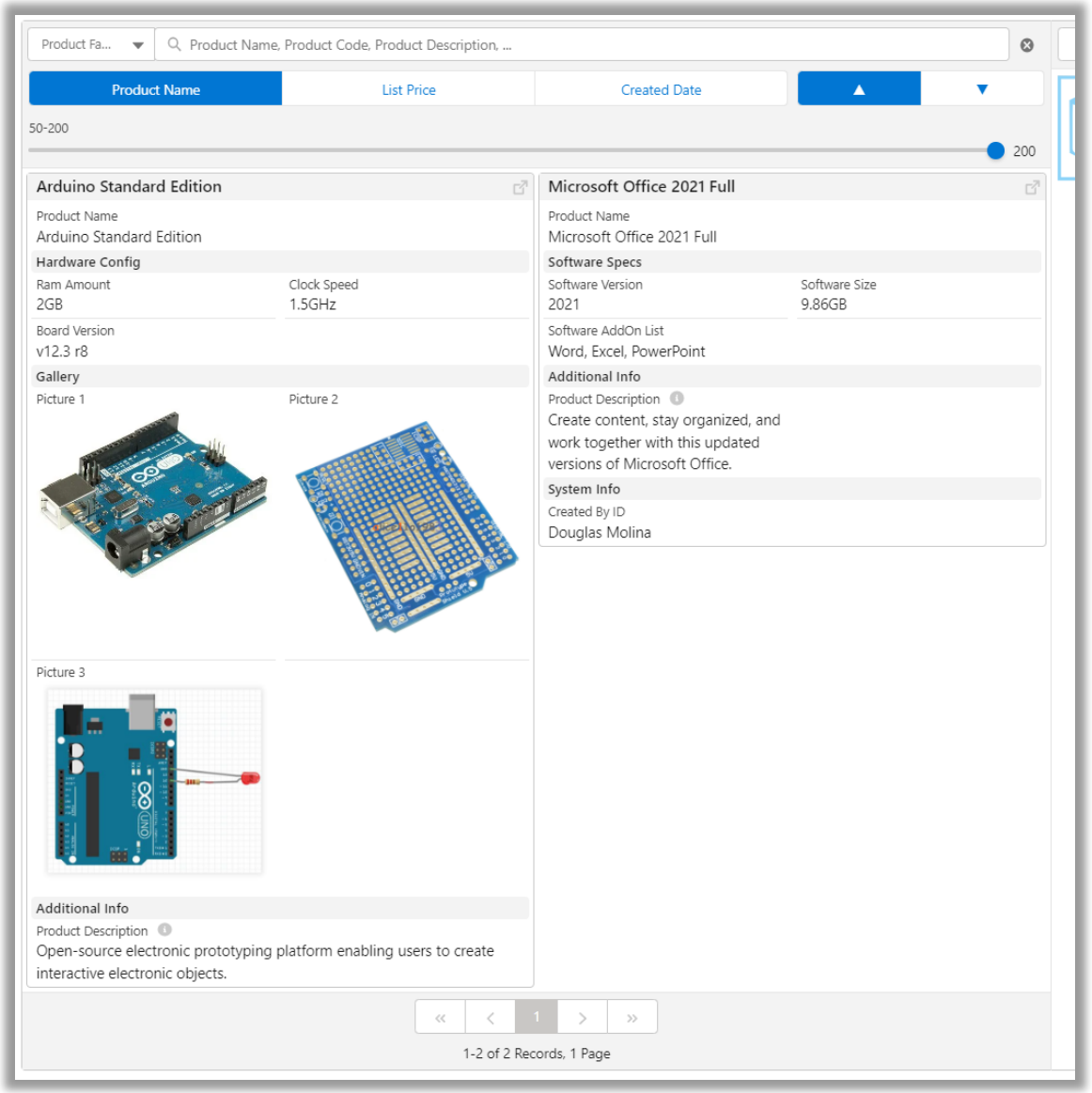
```
[pl]
    Software Product=Product2-Software Product Layout|Hardware Product=Product2-Hardware Product Layout
```

3) Put together the string “[pl]” and the pre-final value, to create the **final attribute value**, like this:

```
[pl]Software Product=Product2-Software Product Layout|Hardware Product=Product2-Hardware Product Layout
```

That being done, just copy and paste this string into one of the details attributes of the component, and you should get results like this:





Creating the Field Mapping for the Product Import Tool

The tools **Import Products from parent Opportunity** (available for Quotes and Orders) and **Import Products from parent Quote** (available for Orders) both utilize the same attribute **Sale Product > Import Tool Mapping** to determine how the Product fields are going to be mapped when cloning them to the current Quote or Order. The component already brings some standard fields mapped, but you can always add your specific/custom ones to the list as well.

To do so, just write down the desired mapping like the example below, where the fields on the left are the **target fields** (of the products that will be created), and the fields on the right are the **source fields** (of the original products). In this example we are configuring the clone from the Opportunity Products (source) to the Quote Products (target), but the idea is the same for all the other scenarios, just be sure to use the appropriate Object name:

```
QuoteLineItem.Quantity        OpportunityLineItem.Quantity
QuoteLineItem.UnitPrice       OpportunityLineItem.UnitPrice
QuoteLineItem.My_txt__c       OpportunityLineItem.My_add_descr__c
```

That being done, you need to join all this structure into a single line of text, concatenating the parts by using the following logic:

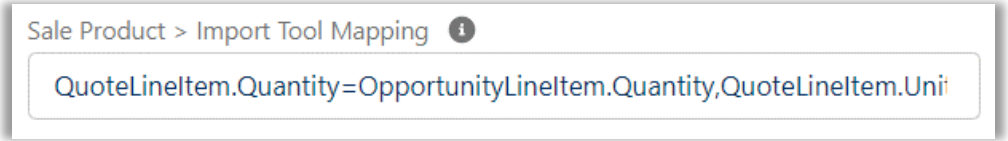
1) Place a **=** character between a target field and its source field, to create a **field mapping**, like this:

```
QuoteLineItem.Quantity=OpportunityLineItem.Quantity
QuoteLineItem.UnitPrice=OpportunityLineItem.UnitPrice
QuoteLineItem.My_txt__c=OpportunityLineItem.My_add_descr__c
```

2) Place a **,** character between the **field mappings**, to create the **final value**, like this:

```
QuoteLineItem.Quantity=OpportunityLineItem.Quantity,QuoteLineItem.UnitPrice=OpportunityLineItem.UnitPrice,Quote
LineItem.My_txt__c=OpportunityLineItem.My_add_descr__c
```

And then just copy and paste this string in the corresponding attribute of the component:



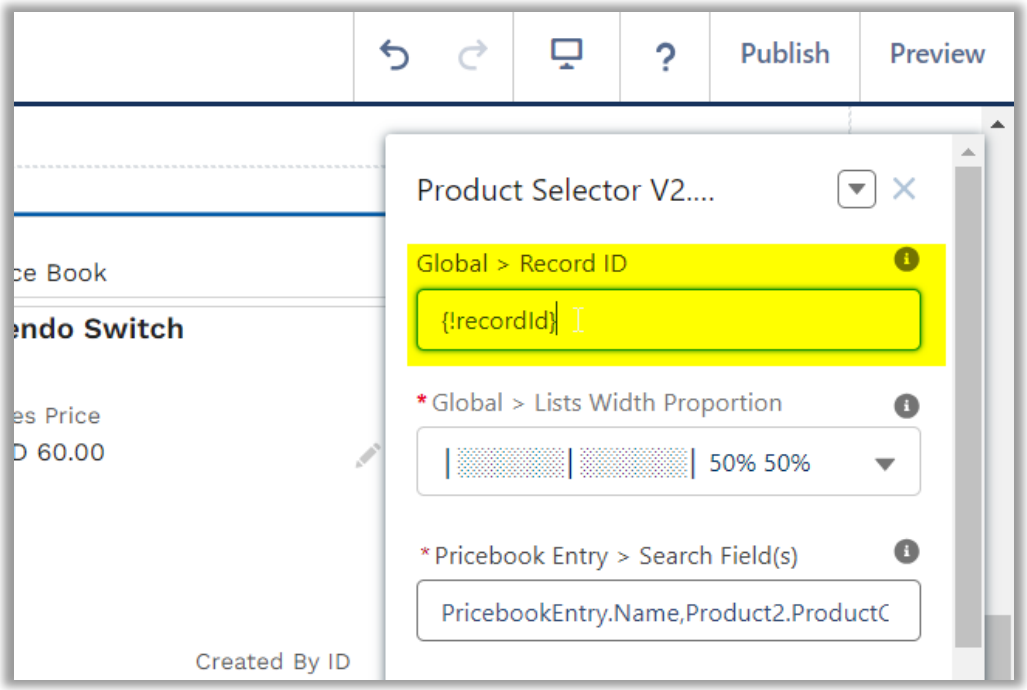
Please note that when creating the field mapping for Orders(target), you can put all the field mappings for both Opportunity(source) and Quote(source) in the same field at the same time, the component will identify the correct ones based on the object names. For example, this would be a valid mapping:

```
OrderItem.Quantity          OpportunityLineItem.Quantity
OrderItem.UnitPrice         OpportunityLineItem.UnitPrice
OrderItem.Special_desc__c   OpportunityLineItem.My_notes__c
OrderItem.Quantity          QuoteLineItem.Quantity
OrderItem.UnitPrice         QuoteLineItem.UnitPrice
OrderItem.Special_desc__c   QuoteLineItem.Memo_2__c
```

```
OrderItem.Quantity=OpportunityLineItem.Quantity,OrderItem.UnitPrice=OpportunityLineItem.UnitPrice,OrderItem.Spe
cial_desc__c=OpportunityLineItem.My_notes__c,OrderItem.Quantity=QuoteLineItem.Quantity,OrderItem.UnitPrice=Quot
eLineItem.UnitPrice,OrderItem.Special_desc__c=QuoteLineItem.Memo_2__c
```

Also note that the **sort order** is not mapped by default, but it can be mapped just like any other field. Just keep in mind that when mapped, **it will only replicate the original sort values, rather than recalculate the values to match any already existing Product on the current Quote or Order**. You can however resort them to your taste after the import has been completed.

One last note, now about mapping the Unit Price of the **Order Product (target)** object. This object **does not have the default Discount field**, so if you use the Discount field of the Opportunity or Quote Products, in order to keep the values consistent, you may want to first create a formula field on those objects calculating the unit price considering the discount, and then you can use this formula to map with the Unit Price of the Order Product.

Using the Component Inside Lightning Communities

This component can be added to Lightning Community pages using the Experience Builder, in the same way you would add it internally using the App Builder. The only difference is that **you have to specify the record id expression** (not the value) at the moment you add the component to the page, by typing the fixed expression **{!recordId}** into the **Global > Record ID** attribute**,** like this: