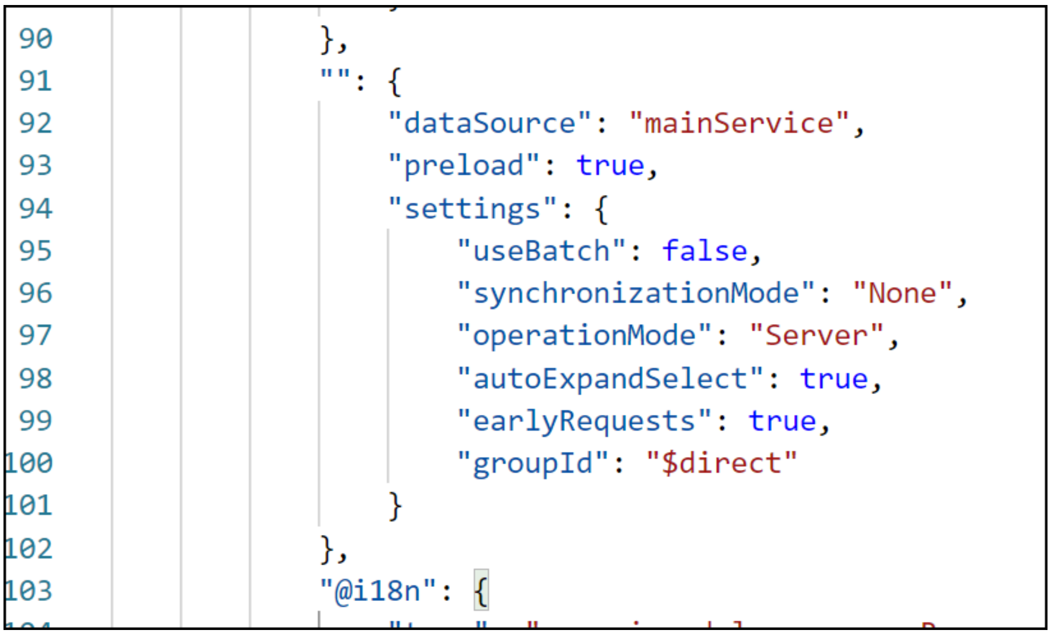
Extend Fiori Elements Overview Page Application

Task 1: Create a new Extension application CDS/OData V2

- Create a new Fiori Application using the Template Wizard for a *Overview Page*

Field	**Value**
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##_V2
Main entity	OverviewPageType
Module name	extend-ovp
Application title	Extend Overview Page
Application namespace	nato.workshop
Description	Extend Overview Page
Project folder path	/home/user/projects
Add deployment configuration	Yes
Add FLP configuration	Yes
Deployment Target	Cloud Foundry
Destination name	abap-cloud-default_xx(SCP)
Add application to managed application router	Yes
Semantic Object	NATO
Action	ExtendOVP
Title	ExtendOVP
Subtitle	Workshop

- Open the manifest.json file and add **"useBatch": false,** to the *settings* part of the *mainModel*

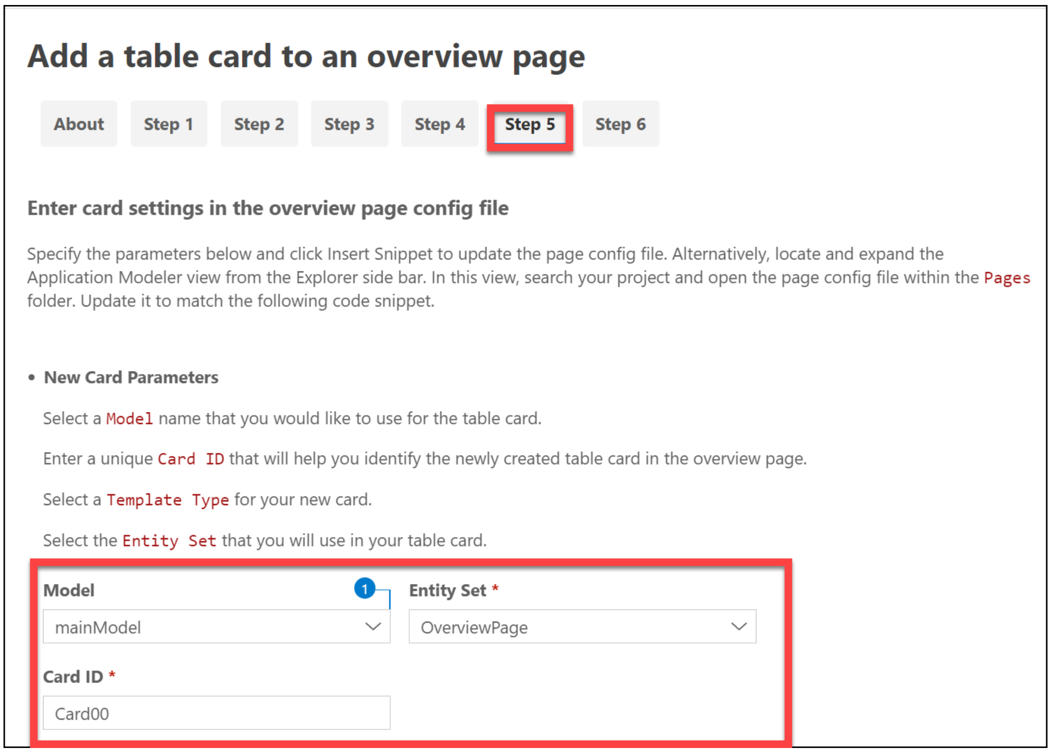


```
            },
            "": {
                "dataSource": "mainService",
                "preload": true,
                "settings": {
                    "useBatch": false,
                    "synchronizationMode": "None",
                    "operationMode": "Server",
                    "autoExpandSelect": true,
                    "earlyRequests": true,
                    "groupId": "$direct"
                }
            },
            "@i18n": {
```

Task 2: Add a table card

- Start a *Guided Development* for **Add a table card to an overview page**
- In *Step 5* use following **New Card Parameters**

Name	**Value**
Model	mainModel
Entity Set	OverviewPage
Card ID	Card00

- 

Add a table card to an overview page

Enter card settings in the overview page config file

Specify the parameters below and click Insert Snippet to update the page config file. Alternatively, locate and expand the Application Modeler view from the Explorer side bar. In this view, search your project and open the page config file within the `Pages` folder. Update it to match the following code snippet.

- New Card Parameters

  Select a `Model` name that you would like to use for the table card.

  Enter a unique `Card ID` that will help you identify the newly created table card in the overview page.

  Select a `Template Type` for your new card.

  Select the `Entity Set` that you will use in your table card.

Model: mainModel

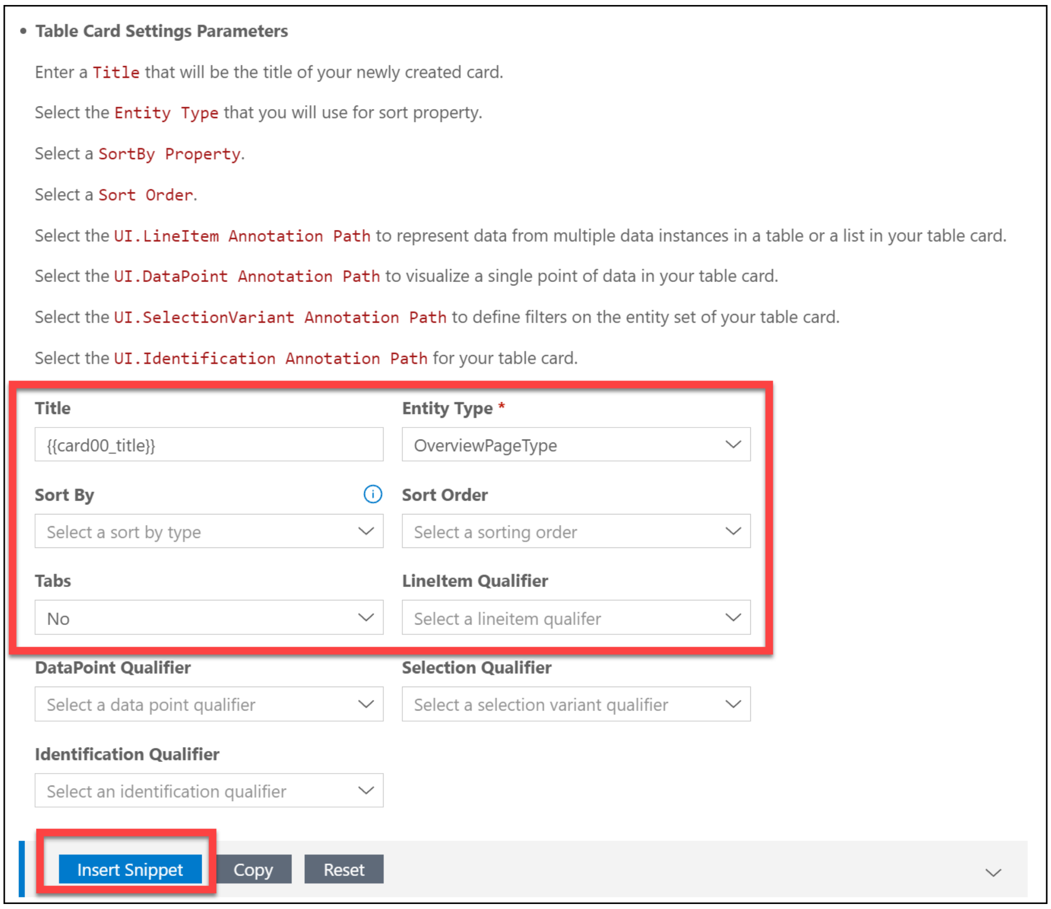
Entity Set *: OverviewPage

Card ID *: Card00

- Add following **Table Card Settings Paramters**

Name	Value
Title	{{card00_title}}
Entity Type	OverviewPageType
Tabs	No

- *Insert Snippet*



- **Table Card Settings Parameters**

Enter a `Title` that will be the title of your newly created card.

Select the `Entity Type` that you will use for sort property.

Select a `SortBy Property`.

Select a `Sort Order`.

Select the `UI.LineItem Annotation Path` to represent data from multiple data instances in a table or a list in your table card.

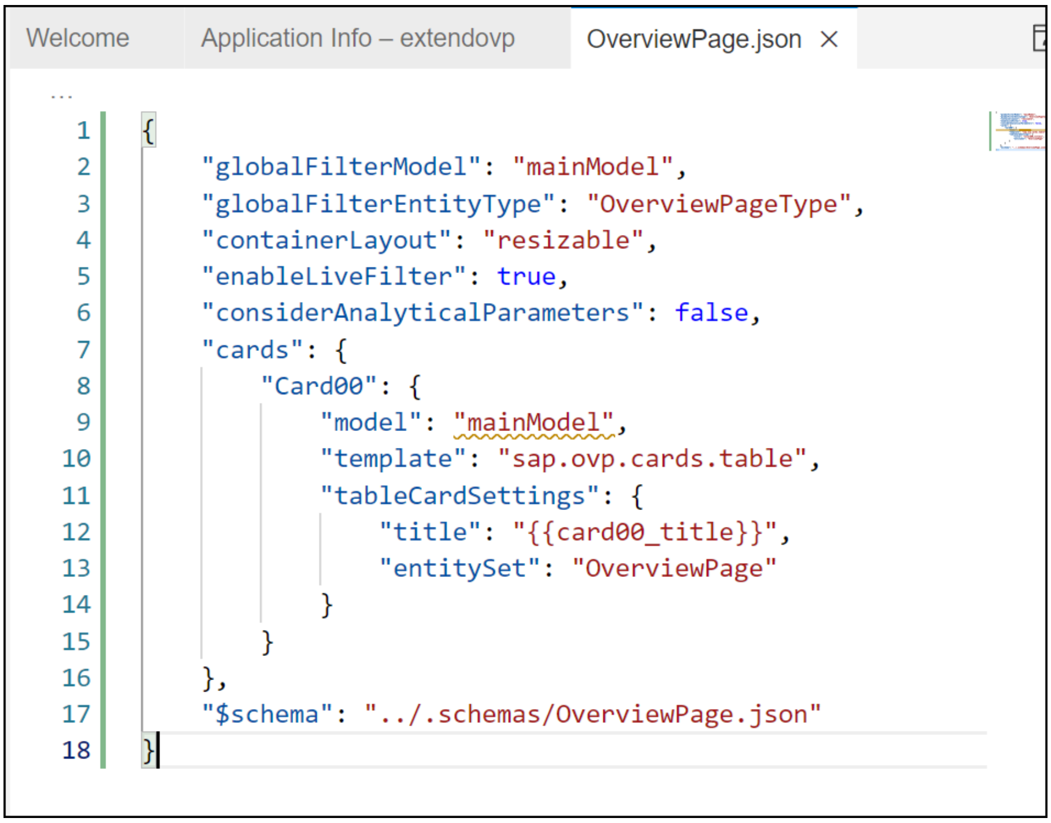
Select the `UI.DataPoint Annotation Path` to visualize a single point of data in your table card.

Select the `UI.SelectionVariant Annotation Path` to define filters on the entity set of your table card.

Select the `UI.Identification Annotation Path` for your table card.

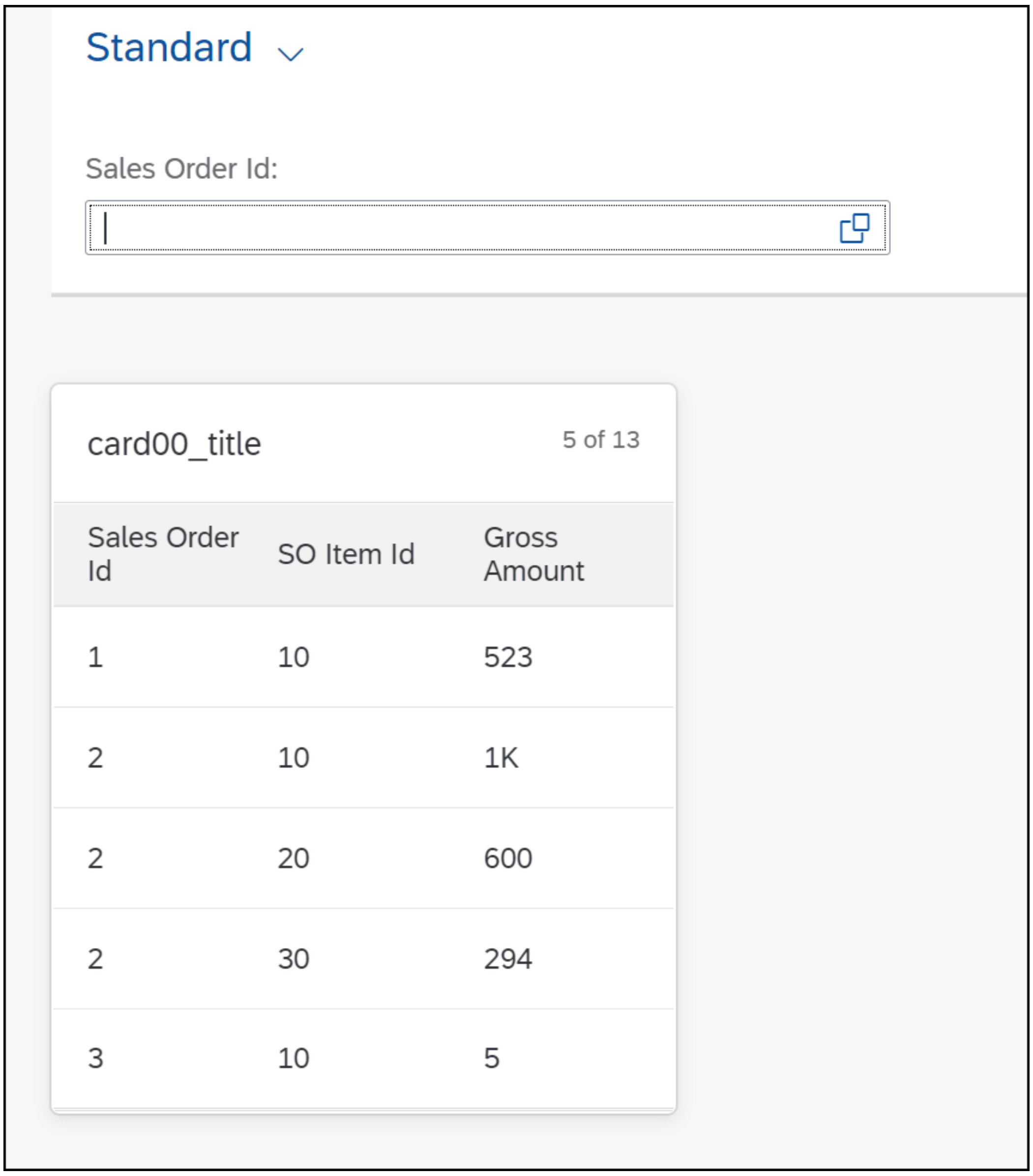
Title	**Entity Type ***
{{card00_title}}	OverviewPageType
Sort By	**Sort Order**
Select a sort by type	Select a sorting order
Tabs	**LineItem Qualifier**
No	Select a lineitem qualifer
DataPoint Qualifier	**Selection Qualifier**
Select a data point qualifier	Select a selection variant qualifier
Identification Qualifier	
Select an identification qualifier	

- OverviewPage.josn



```json
{
    "globalFilterModel": "mainModel",
    "globalFilterEntityType": "OverviewPageType",
    "containerLayout": "resizable",
    "enableLiveFilter": true,
    "considerAnalyticalParameters": false,
    "cards": {
        "Card00": {
            "model": "mainModel",
            "template": "sap.ovp.cards.table",
            "tableCardSettings": {
                "title": "{{card00_title}}",
                "entitySet": "OverviewPage"
            }
        }
    },
    "$schema": "../.schemas/OverviewPage.json"
}
```

- Preview Application



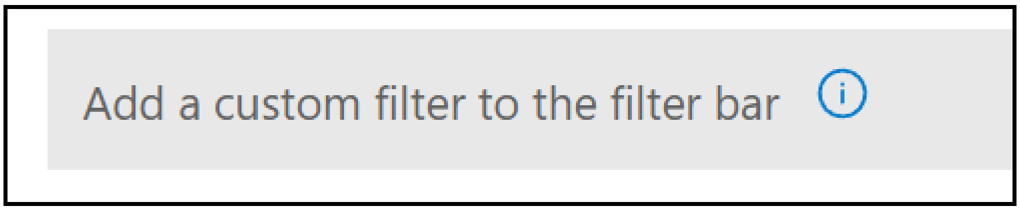
Sales Order Id	SO Item Id	Gross Amount
1	10	523
2	10	1K
2	20	600
2	30	294
3	10	5

Task 3: Add a custom filter to the filter bar

- Add **card00_title=Recent Sales card01_title=Recent Products sold** to *i18n.properties*

```
#XTIT: Card00_title
card00_title=Recent Sales

#XTIT: Card01_title
card01_title=Recent Products sold
```

- Start a *Guided Development* for **Add a custom filter to the filter bar**

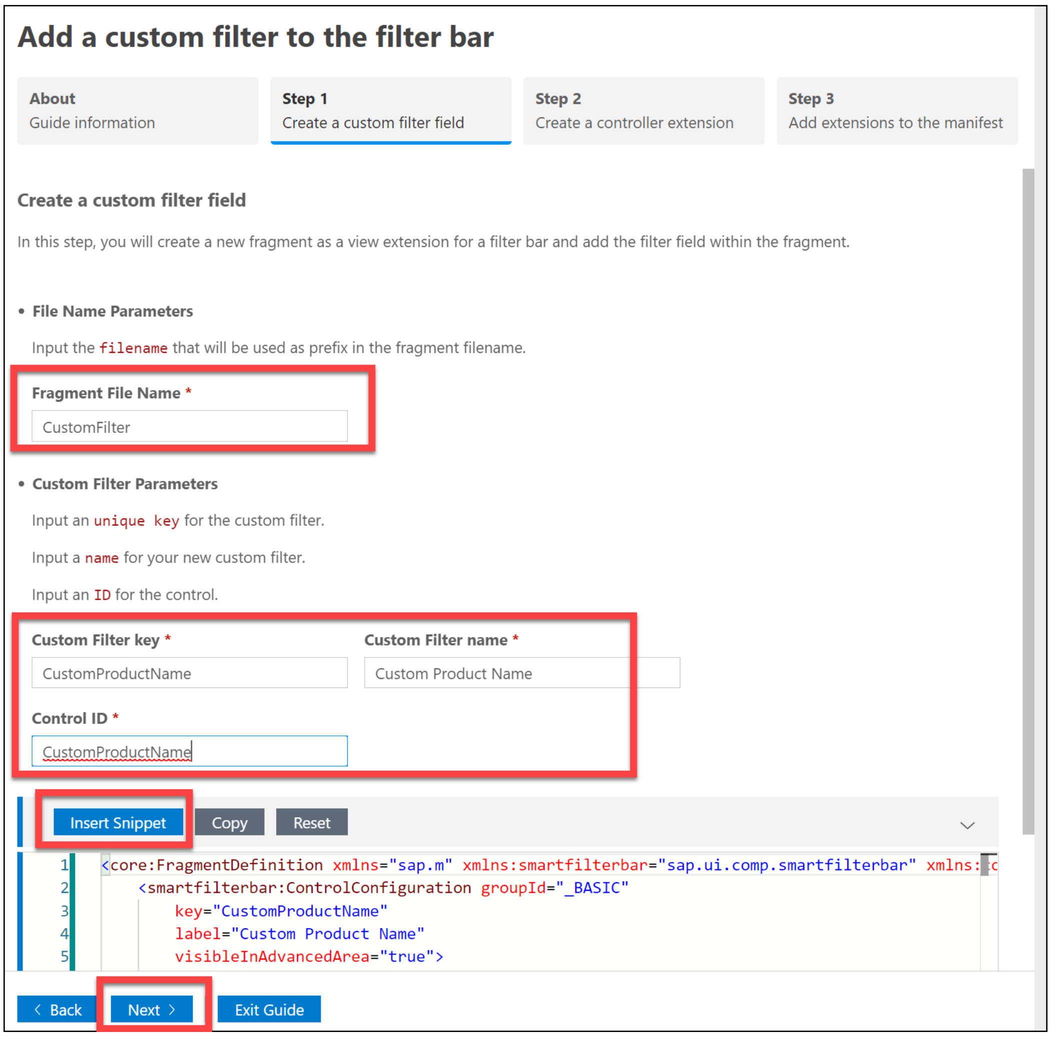


Add a custom filter to the filter bar

- In *Step 1* use following parameters

Name	Value
Fragment File Name	CustomFilter
Custom Filter Key	CustomProductName
Custom Filter Name	Custom Product Name
Control ID	CustomProductName

- *Insert Snippet* and *Next*



Add a custom filter to the filter bar

About: Guide information | Step 1: Create a custom filter field | Step 2: Create a controller extension | Step 3: Add extensions to the manifest

Create a custom filter field

In this step, you will create a new fragment as a view extension for a filter bar and add the filter field within the fragment.

- File Name Parameters

  Input the `filename` that will be used as prefix in the fragment filename.

  Fragment File Name *: CustomFilter

- Custom Filter Parameters

  Input an `unique key` for the custom filter.

  Input a `name` for your new custom filter.

  Input an `ID` for the control.

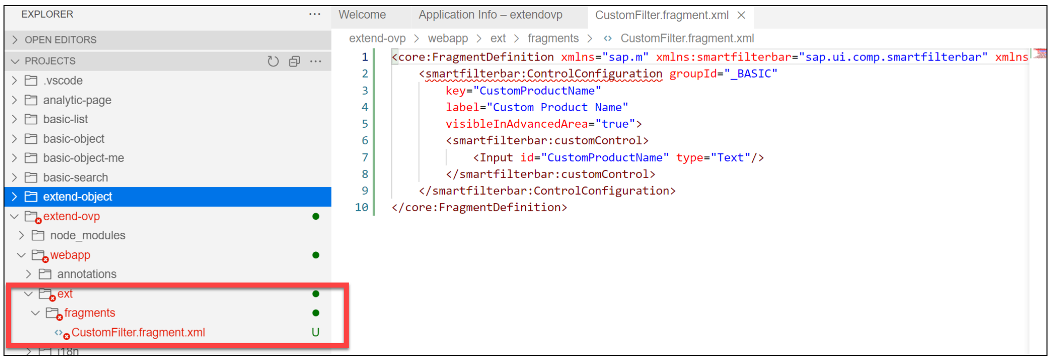
  Custom Filter key *: CustomProductName

  Custom Filter name *: Custom Product Name

  Control ID *: CustomProductName

```
<core:FragmentDefinition xmlns="sap.m" xmlns:smartfilterbar="sap.ui.comp.smartfilterbar" xmlns:c
    <smartfilterbar:ControlConfiguration groupId="_BASIC"
        key="CustomProductName"
        label="Custom Product Name"
        visibleInAdvancedArea="true">
```

- This creates a new folder **ext/fragments** with file **CustomFilter.fragment.xml**

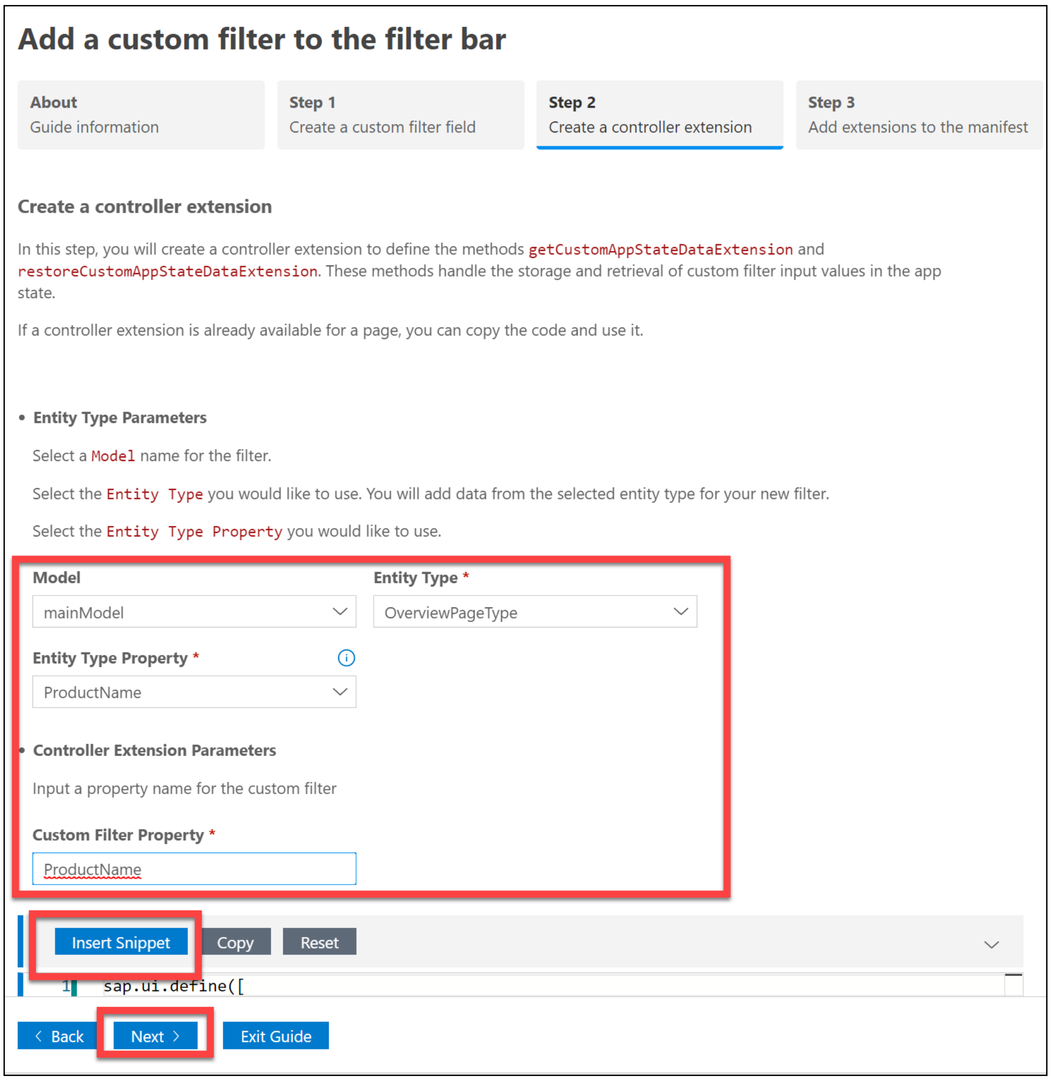


```xml
<core:FragmentDefinition xmlns="sap.m" xmlns:smartfilterbar="sap.ui.comp.smartfilterbar" xmlns
    <smartfilterbar:ControlConfiguration groupId="_BASIC"
        key="CustomProductName"
        label="Custom Product Name"
        visibleInAdvancedArea="true">
        <smartfilterbar:customControl>
            <Input id="CustomProductName" type="Text"/>
        </smartfilterbar:customControl>
    </smartfilterbar:ControlConfiguration>
</core:FragmentDefinition>
```

- In *Step 2* use following parameters

Name	Value
Model	mainModel
Entity Type	OverviewPageType
Entity Type Property	ProductName
Custom Filter Property	ProductName

- *Insert Snippet* and *Next*



Add a custom filter to the filter bar

About	Step 1	Step 2	Step 3
Guide information	Create a custom filter field	Create a controller extension	Add extensions to the manifest

Create a controller extension

In this step, you will create a controller extension to define the methods `getCustomAppStateDataExtension` and `restoreCustomAppStateDataExtension`. These methods handle the storage and retrieval of custom filter input values in the app state.

If a controller extension is already available for a page, you can copy the code and use it.

- **Entity Type Parameters**

Select a `Model` name for the filter.

Select the `Entity Type` you would like to use. You will add data from the selected entity type for your new filter.

Select the `Entity Type Property` you would like to use.

Model: mainModel
Entity Type *: OverviewPageType
Entity Type Property *: ProductName

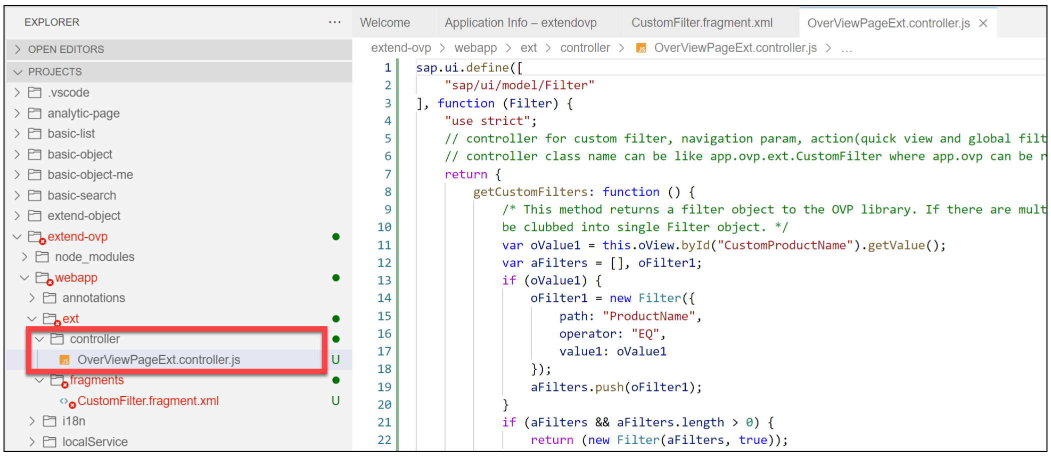
- **Controller Extension Parameters**

Input a property name for the custom filter

Custom Filter Property *: ProductName

```
sap.ui.define([
```

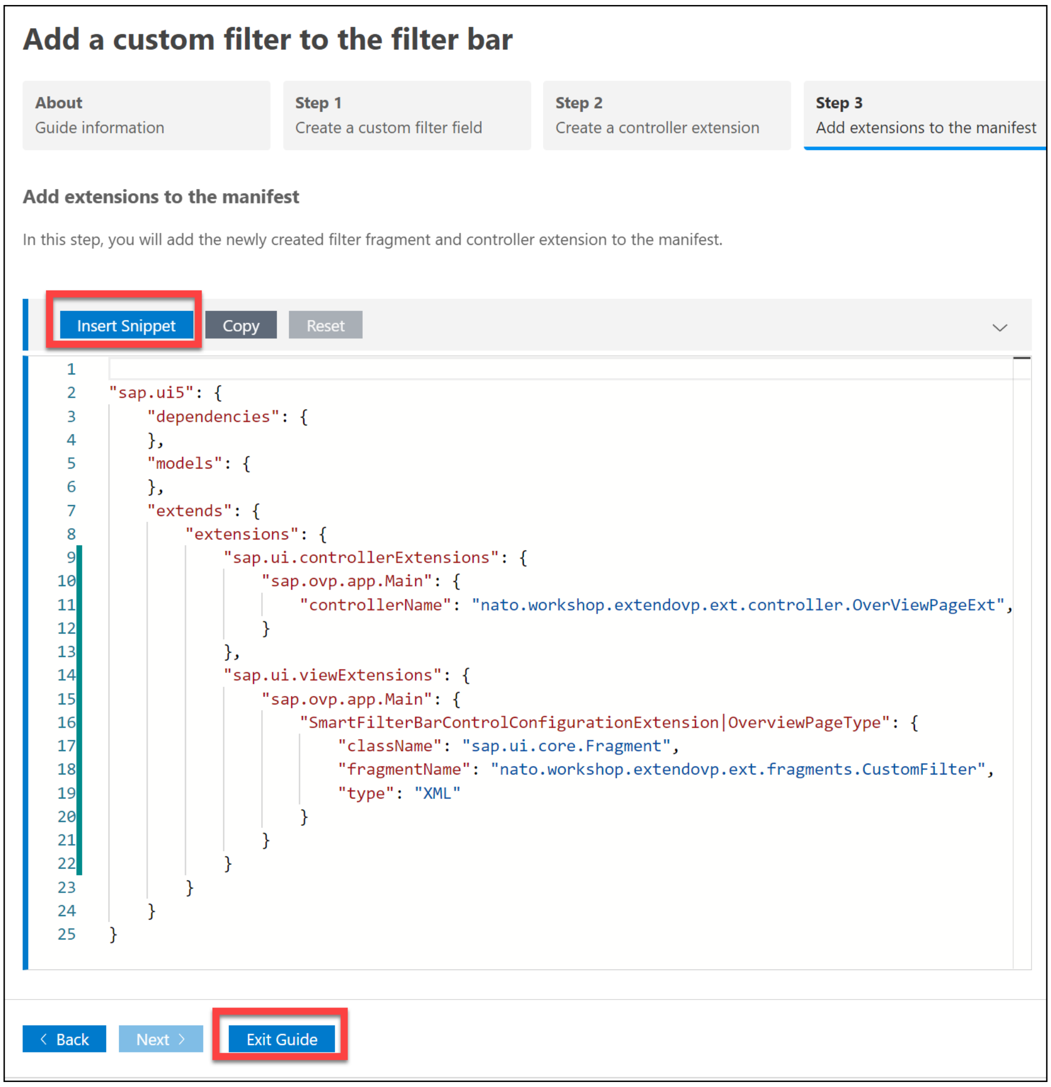
- This creates a new folder **ext/controller** with file **OverViewPageExt.controller.js**



EXPLORER
OPEN EDITORS
PROJECTS
.vscode
analytic-page
basic-list
basic-object
basic-object-me
basic-search
extend-object
extend-ovp
node_modules
webapp
annotations
ext
controller
OverViewPageExt.controller.js
fragments
CustomFilter.fragment.xml
i18n
localService
Welcome
Application Info – extendovp
CustomFilter.fragment.xml
OverViewPageExt.controller.js
extend-ovp > webapp > ext > controller > OverViewPageExt.controller.js > …

```
sap.ui.define([
    "sap/ui/model/Filter"
], function (Filter) {
    "use strict";
    // controller for custom filter, navigation param, action(quick view and global filt
    // controller class name can be like app.ovp.ext.CustomFilter where app.ovp can be r
    return {
        getCustomFilters: function () {
            /* This method returns a filter object to the OVP library. If there are mult
            be clubbed into single Filter object. */
            var oValue1 = this.oView.byId("CustomProductName").getValue();
            var aFilters = [], oFilter1;
            if (oValue1) {
                oFilter1 = new Filter({
                    path: "ProductName",
                    operator: "EQ",
                    value1: oValue1
                });
                aFilters.push(oFilter1);
            }
            if (aFilters && aFilters.length > 0) {
                return (new Filter(aFilters, true));
```

- In *Step 3 Insert Snippet* and *Exit Guide*


Add a custom filter to the filter bar
About
Guide information
Step 1
Create a custom filter field
Step 2
Create a controller extension
Step 3
Add extensions to the manifest
Add extensions to the manifest
In this step, you will add the newly created filter fragment and controller extension to the manifest.
Insert Snippet
Copy
Reset
Back
Next
Exit Guide

```

"sap.ui5": {
    "dependencies": {
    },
    "models": {
    },
    "extends": {
        "extensions": {
            "sap.ui.controllerExtensions": {
                "sap.ovp.app.Main": {
                    "controllerName": "nato.workshop.extendovp.ext.controller.OverViewPageExt",
                }
            },
            "sap.ui.viewExtensions": {
                "sap.ovp.app.Main": {
                    "SmartFilterBarControlConfigurationExtension|OverviewPageType": {
                        "className": "sap.ui.core.Fragment",
                        "fragmentName": "nato.workshop.extendovp.ext.fragments.CustomFilter",
                        "type": "XML"
                    }
                }
            }
        }
    }
}
```

- Implement **CustomFilter.fragment.xml** by adding a *Select*

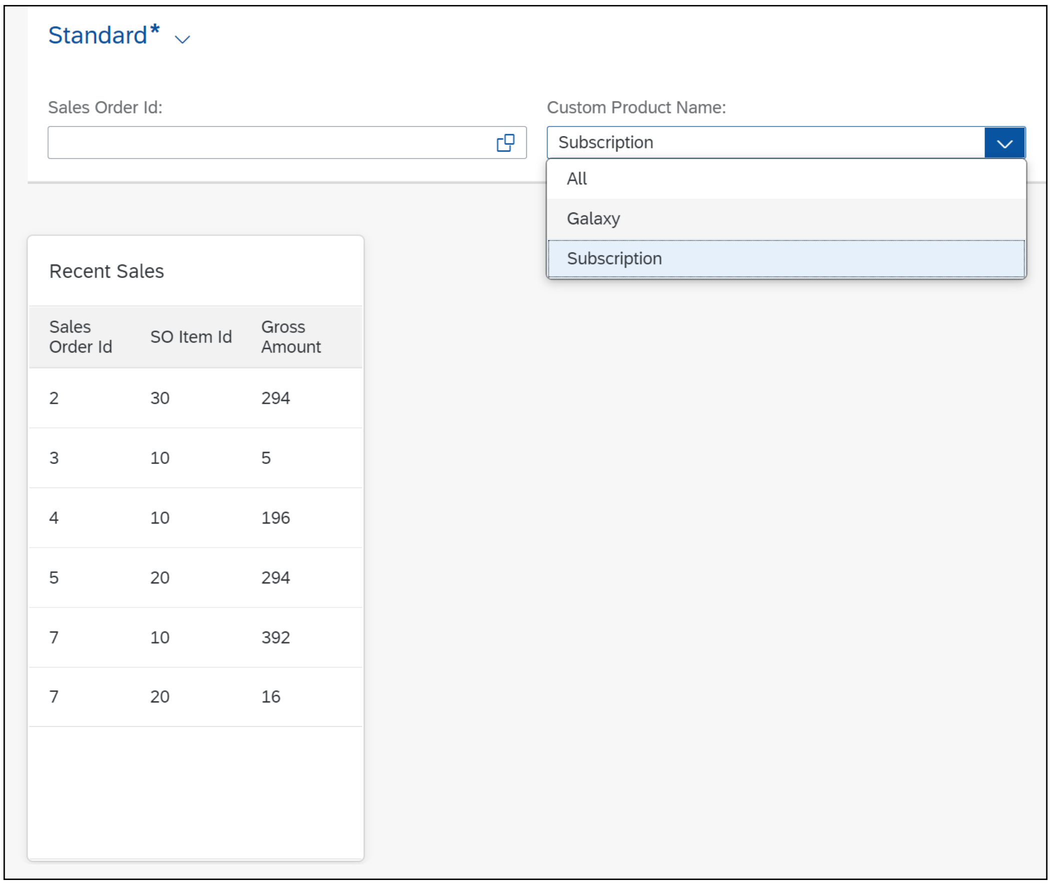
```
<core:FragmentDefinition xmlns="sap.m"
xmlns:smartfilterbar="sap.ui.comp.smartfilterbar"
xmlns:core="sap.ui.core">
    <smartfilterbar:ControlConfiguration id="CustomFilter"
groupId="_BASIC"
        key="CustomProductName"
        label="Custom Product Name"
        visibleInAdvancedArea="true">
        <smartfilterbar:customControl>
            <Select id="CustomProductName">
                <core:Item id="all" text="All" key="All"/>
                <core:Item id="Gal" text="Galaxy" key="Gal"/>
                <core:Item id="Sub" text="Subscription"
key="Sub"/>
            </Select>
        </smartfilterbar:customControl>
    </smartfilterbar:ControlConfiguration>
</core:FragmentDefinition>
```

- Implement **OverViewPageExt.controller.js**

```
sap.ui.define([
    "sap/ui/model/Filter",
    "sap/ui/model/FilterOperator"
], function (Filter,FilterOperator) {
    "use strict";
    // controller for custom filter, navigation param,
action(quick view and global filter), navigation target
    // controller class name can be like app.ovp.ext.CustomFilter
where app.ovp can be replaced with your application namespace
    return {
        getCustomFilters: function () {
            /* This method returns a filter object to the OVP
library. If there are multiple filters, they should
            be clubbed into single Filter object. */
            var sSelectedKey =
this.oView.byId("CustomProductName").getSelectedKey();
            var aFilters = [], oFilter1;

            switch (sSelectedKey){
                case "All":
                    break;
                case "Gal":
                    oFilter1 = new Filter({path:"ProductName",
operator: FilterOperator.StartsWith, value1: "Gal"})
                    aFilters.push(oFilter1);
                    break;
                case "Sub":
                    oFilter1 = new Filter({path:"ProductName",
operator: FilterOperator.NotStartsWith, value1: "Gal"})
                    aFilters.push(oFilter1);
                    break;
                }

            if (aFilters && aFilters.length > 0) {
                return (new Filter(aFilters, true));
            }
        },
        getCustomAppStateDataExtension: function (oCustomData) {
            //the content of the custom field will be stored in
the app state, so that it can be restored later, for example after
a back navigation.
```

```
            //The developer has to ensure that the content of the
field is stored in the object that is returned by this method.
            if (oCustomData) {
                var oCustomField1 =
this.oView.byId("CustomProductName");
                if (oCustomField1) {
                    oCustomData.ProductName =
oCustomField1.getSelectedKey();
                }
            }
        },
        restoreCustomAppStateDataExtension: function (oCustomData)
{
            //in order to restore the content of the custom field
in the filter bar, for example after a back navigation,
            //an object with the content is handed over to this
method. Now the developer has to ensure that the content of the
custom filter is set to the control
            if (oCustomData) {
                if (oCustomData.ProductName) {
                    var oCustomField1 =
this.oView.byId("CustomProductName");

oCustomField1.setValue(oCustomData.ProductName);
                }
            }
        }
    }
});
```

- Preview Application



Standard*

Sales Order Id:

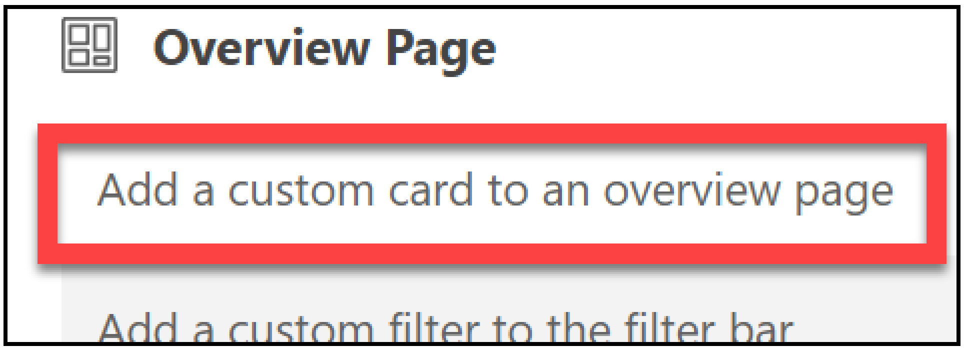
Custom Product Name: Subscription

Recent Sales

Sales Order Id	SO Item Id	Gross Amount
2	30	294
3	10	5
4	10	196
5	20	294
7	10	392
7	20	16

Task 4: Add a custom card

- Start a *Guided Development* for **Add a custom card to an overview page**

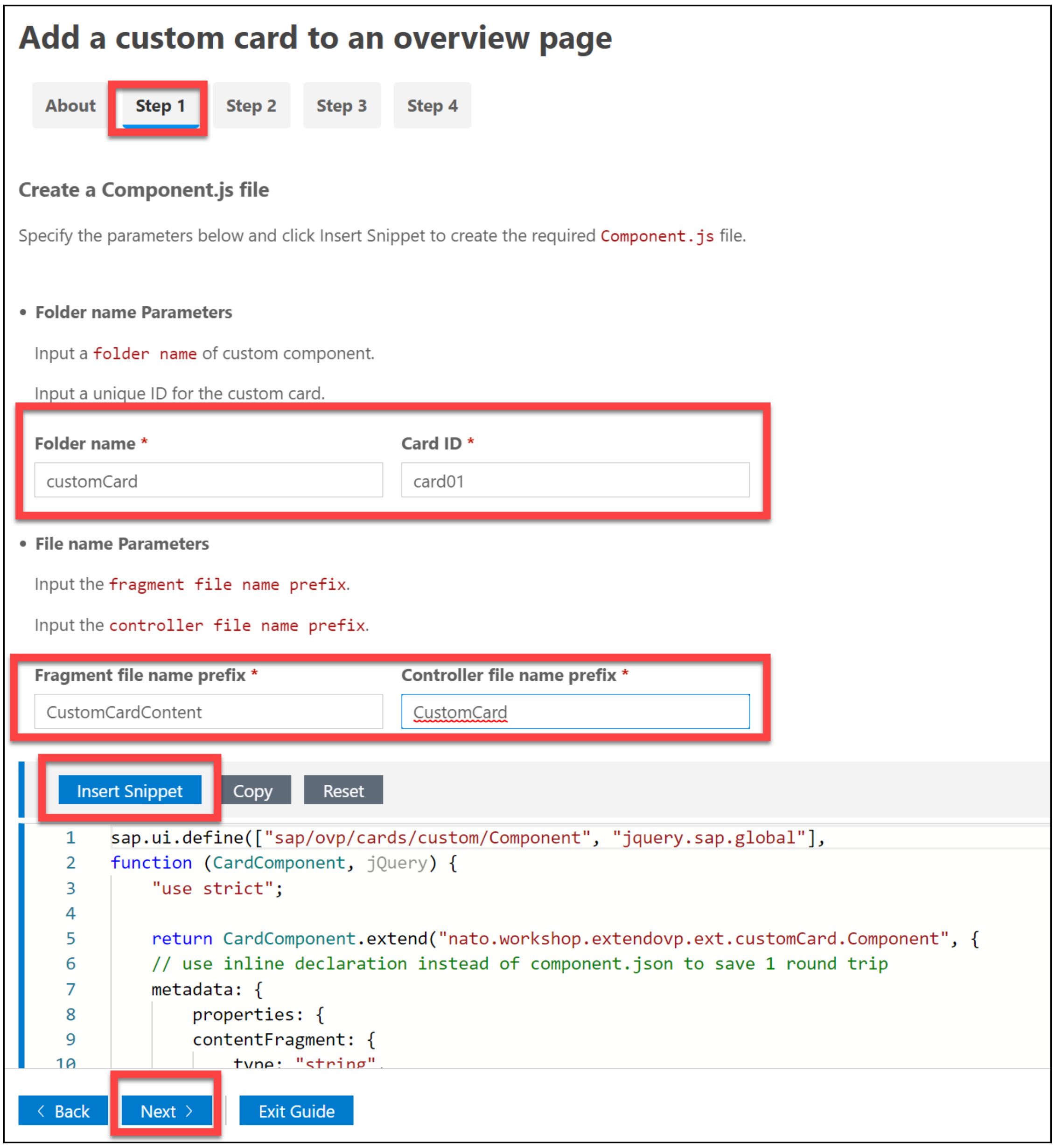


Overview Page
Add a custom card to an overview page
Add a custom filter to the filter bar

- In *Step 1* use following parameters

Name	Value
Folder name	customCard
Card ID	card01
Fragment file name prefix	CustomCardContent
Controller file name prefix	CustomCard

Insert Snippet and *Next*



Add a custom card to an overview page

Create a Component.js file

Specify the parameters below and click Insert Snippet to create the required `Component.js` file.

- Folder name Parameters

  Input a `folder name` of custom component.

  Input a unique ID for the custom card.

Folder name *	Card ID *
customCard	card01

- File name Parameters

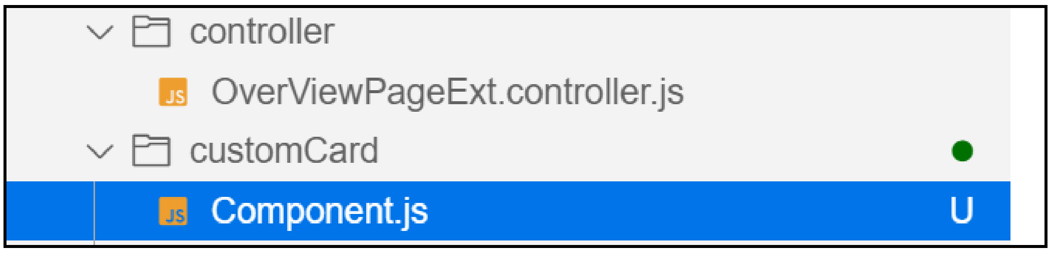
  Input the `fragment file name prefix`.

  Input the `controller file name prefix`.

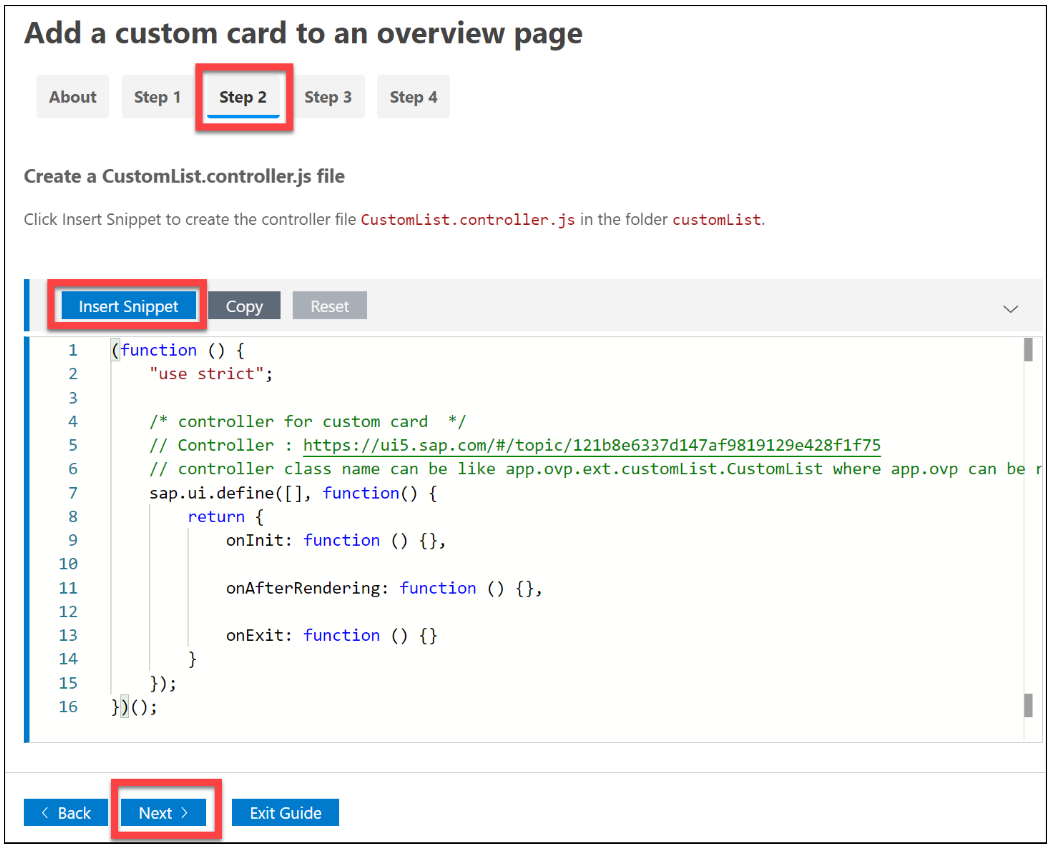
Fragment file name prefix *	Controller file name prefix *
CustomCardContent	CustomCard

```
sap.ui.define(["sap/ovp/cards/custom/Component", "jquery.sap.global"],
function (CardComponent, jQuery) {
    "use strict";

    return CardComponent.extend("nato.workshop.extendovp.ext.customCard.Component", {
    // use inline declaration instead of component.json to save 1 round trip
    metadata: {
        properties: {
        contentFragment: {
            type: "string",
```

- This creates a new folder **customCard** with file **Component.js**



- In *Step 2 Insert Snippet* and *Next*



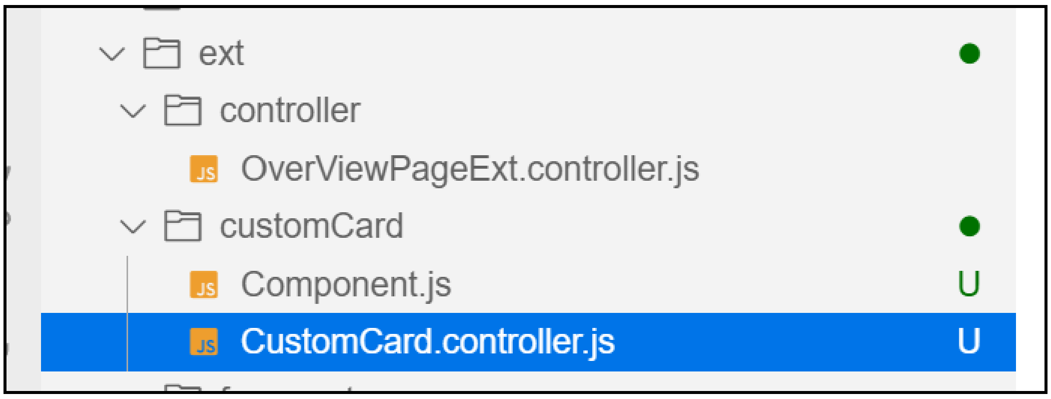
Add a custom card to an overview page

Create a CustomList.controller.js file

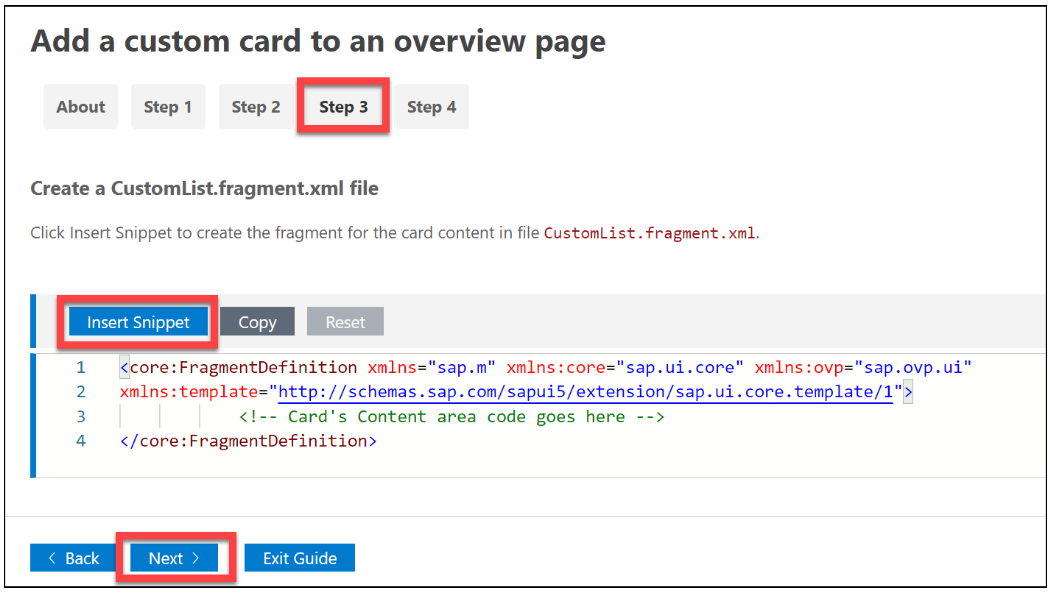
Click Insert Snippet to create the controller file `CustomList.controller.js` in the folder `customList`.

```
(function () {
    "use strict";

    /* controller for custom card  */
    // Controller : https://ui5.sap.com/#/topic/121b8e6337d147af9819129e428f1f75
    // controller class name can be like app.ovp.ext.customList.CustomList where app.ovp can be r
    sap.ui.define([], function() {
        return {
            onInit: function () {},

            onAfterRendering: function () {},

            onExit: function () {}
        }
    });
})();
```

- This creates a new file **CustomCard.controller.js**



- In *Step 3 Insert Snippet* and *Next*



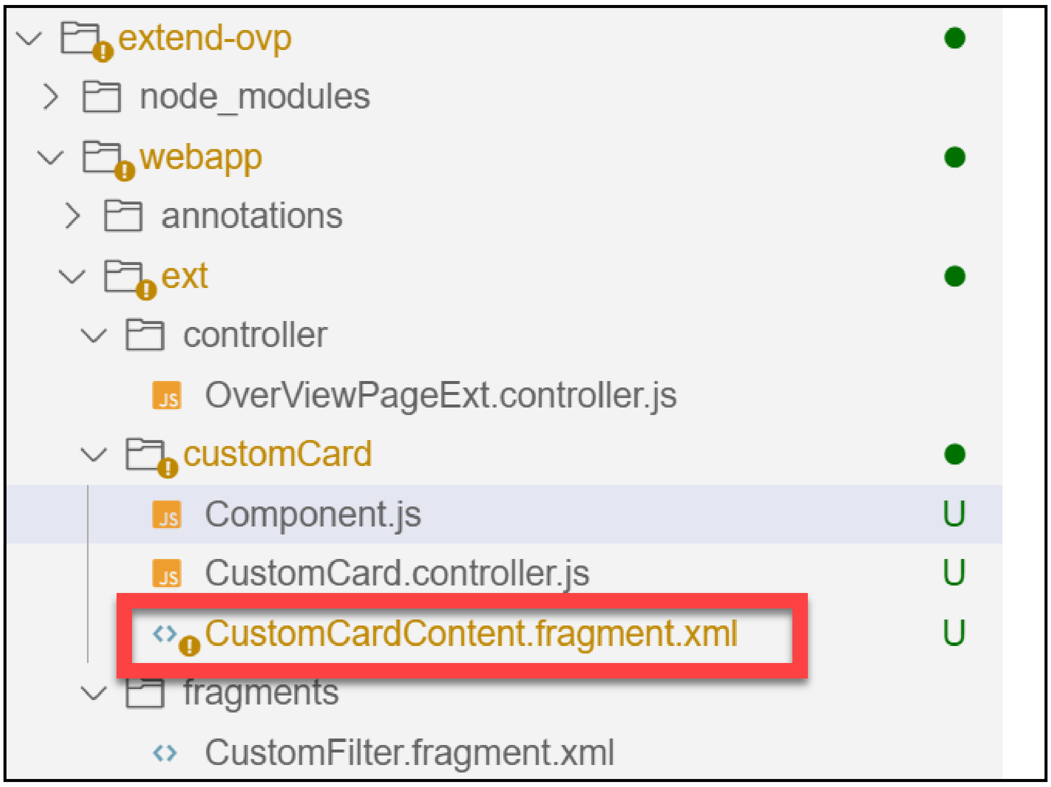
Add a custom card to an overview page

Create a CustomList.fragment.xml file

Click Insert Snippet to create the fragment for the card content in file `CustomList.fragment.xml`.

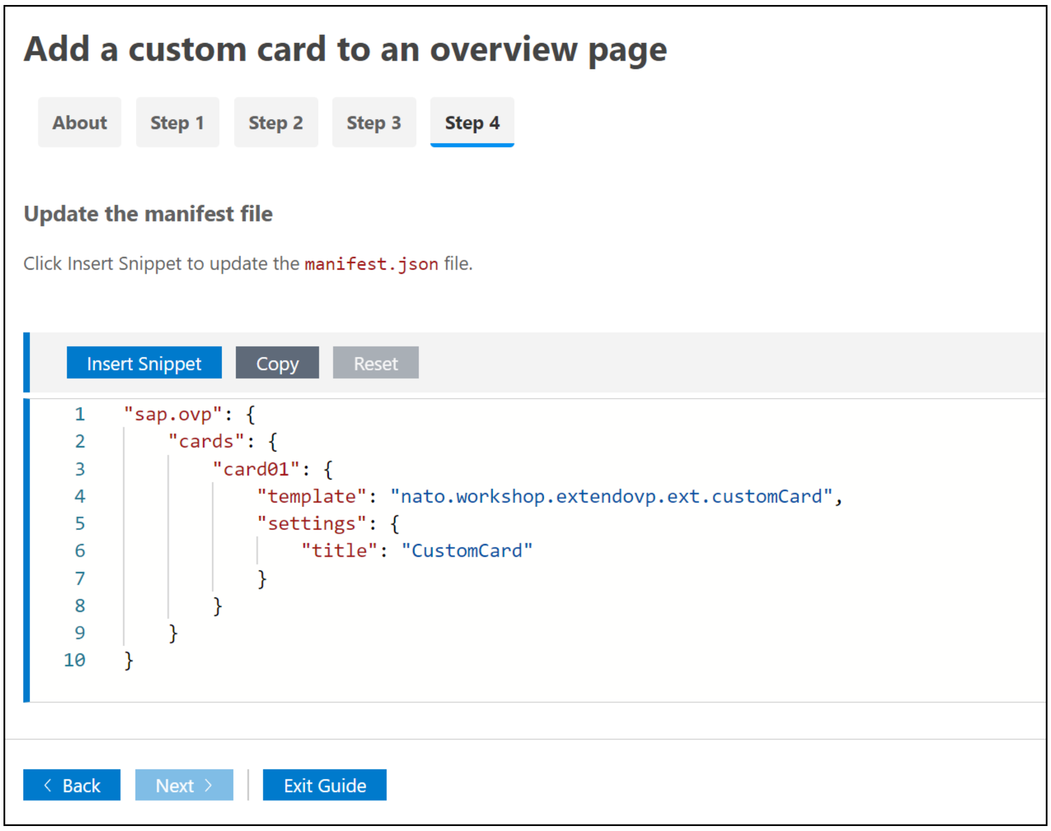
```xml
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core" xmlns:ovp="sap.ovp.ui"
xmlns:template="http://schemas.sap.com/sapui5/extension/sap.ui.core.template/1">
            <!-- Card's Content area code goes here -->
</core:FragmentDefinition>
```

- This creates a new file **CustomCardContent.fragment.xml**



```
extend-ovp
  node_modules
  webapp
    annotations
    ext
      controller
        OverViewPageExt.controller.js
      customCard
        Component.js                       U
        CustomCard.controller.js           U
        CustomCardContent.fragment.xml     U
      fragments
        CustomFilter.fragment.xml
```

- In *Step 4 Insert Snippet* and *Exit Guide*



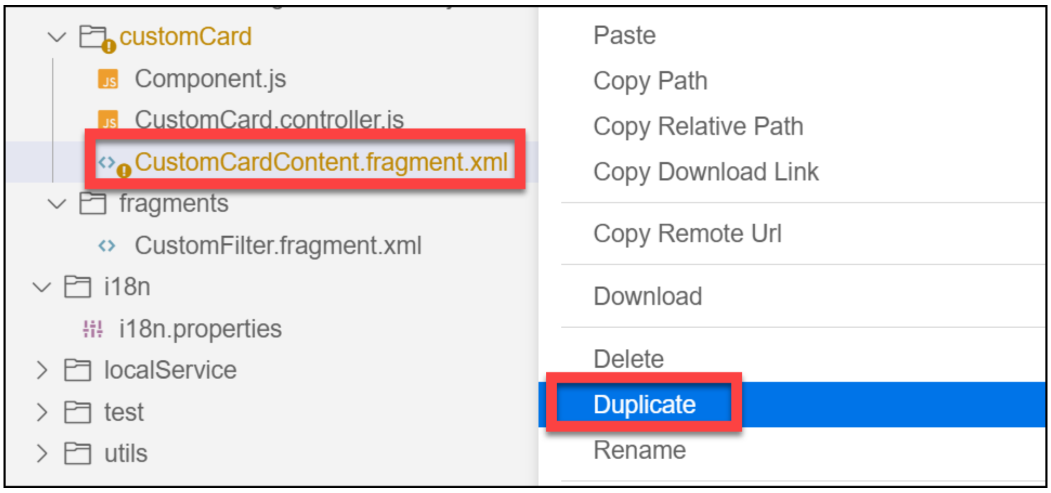
Add a custom card to an overview page

Update the manifest file

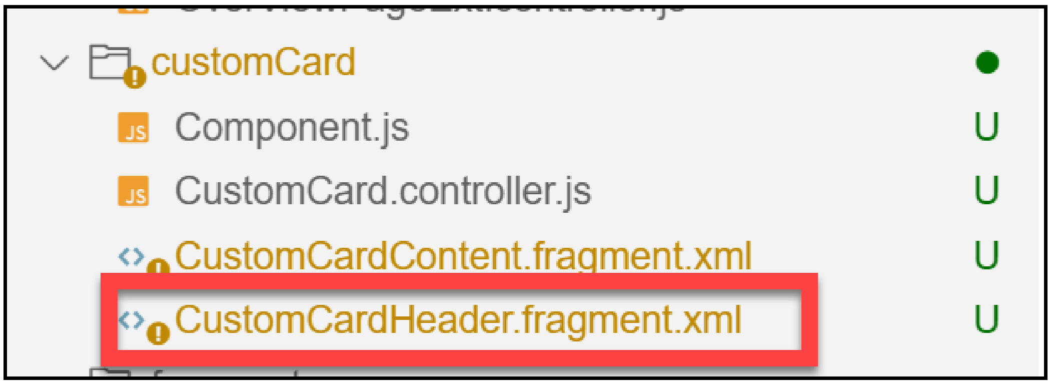
Click Insert Snippet to update the `manifest.json` file.

```
"sap.ovp": {
    "cards": {
        "card01": {
            "template": "nato.workshop.extendovp.ext.customCard",
            "settings": {
                "title": "CustomCard"
            }
        }
    }
}
```

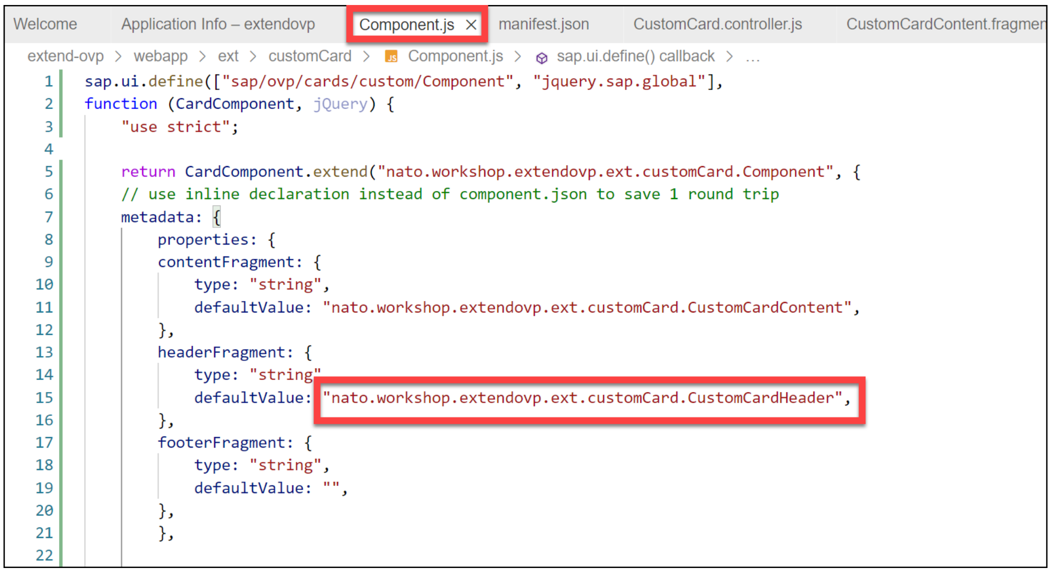
- Duplicate the **CustomCardContent.fragment.xml** file to **CustomCardHeader.fragment.xml**, because we also want a header in our custom card



- CustomCardHeader.fragment.xml


customCard
Component.js U
CustomCard.controller.js U
CustomCardContent.fragment.xml U
CustomCardHeader.fragment.xml U

- in file **ext/customCard/Component.js** copy the *defaultValue* from **ContentFragment** to **headerFragment** and change it to **CustomCardHeader**


Welcome
Application Info – extendovp
Component.js
manifest.json
CustomCard.controller.js
CustomCardContent.fragmen
extend-ovp > webapp > ext > customCard > Component.js > sap.ui.define() callback > ...

```
sap.ui.define(["sap/ovp/cards/custom/Component", "jquery.sap.global"],
function (CardComponent, jQuery) {
    "use strict";

    return CardComponent.extend("nato.workshop.extendovp.ext.customCard.Component", {
    // use inline declaration instead of component.json to save 1 round trip
    metadata: {
        properties: {
        contentFragment: {
            type: "string",
            defaultValue: "nato.workshop.extendovp.ext.customCard.CustomCardContent",
        },
        headerFragment: {
            type: "string",
            defaultValue: "nato.workshop.extendovp.ext.customCard.CustomCardHeader",
        },
        footerFragment: {
            type: "string",
            defaultValue: "",
        },
        },
```

- Implement **CustomCardHeader.fragment.xml**

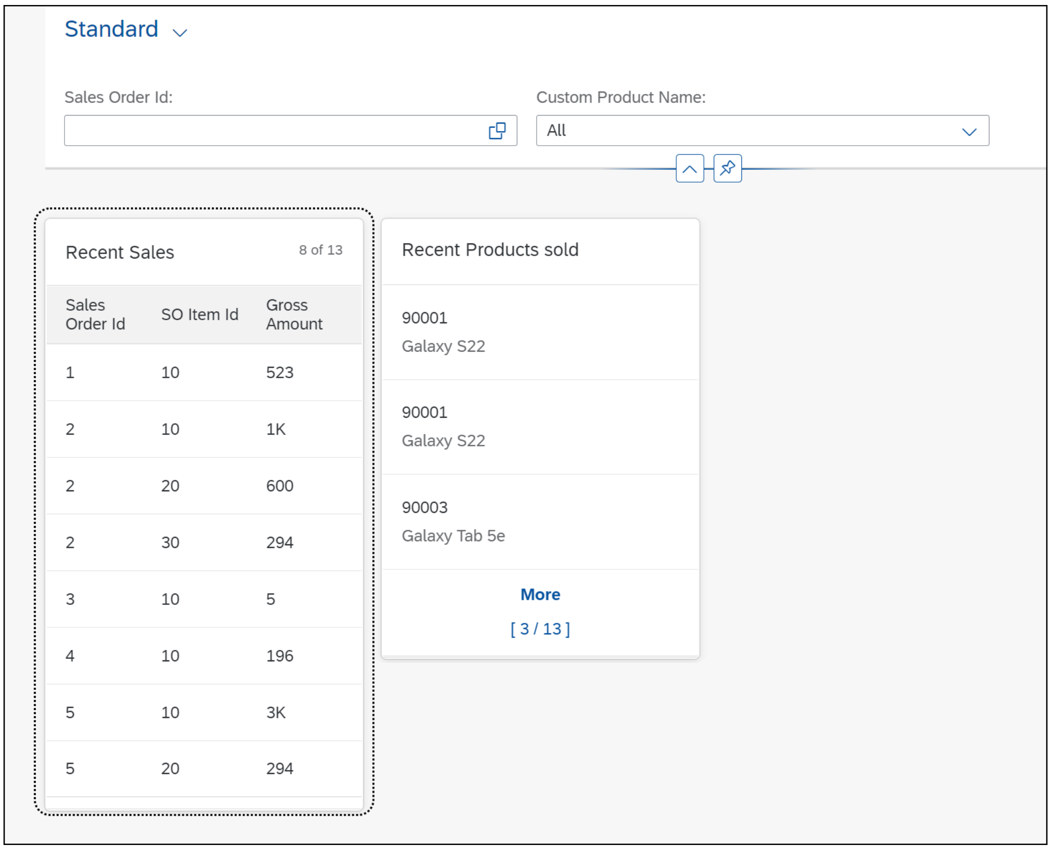
```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"
xmlns:ovp="sap.ovp.ui"
xmlns:template="http://schemas.sap.com/sapui5/extension/sap.ui.cor
e.template/1">
    <VBox id="idVbox">
    <Title id="idTitle" text="{i18n>card01_title}"
class="sapUiSmallMargin"/>
    </VBox>
</core:FragmentDefinition>
```

*Implement **CustomCardContent.fragment.xml**

```
<core:FragmentDefinition xmlns="sap.m" xmlns:core="sap.ui.core"
xmlns:ovp="sap.ovp.ui"
```

```
xmlns:template="http://schemas.sap.com/sapui5/extension/sap.ui.cor
e.template/1">
    <List id="idList" items="{ path: '/OverviewPage',
templateShareable: false}"
         growing="true"
         growingThreshold="3">
        <items>
            <StandardListItem id="idSLI" title="{ProductId}"
description="{ProductName}"/>
        </items>
    </List>
</core:FragmentDefinition>
```

- Preview Application



Recent Sales 8 of 13

Sales Order Id	SO Item Id	Gross Amount
1	10	523
2	10	1K
2	20	600
2	30	294
3	10	5
4	10	196
5	10	3K
5	20	294

Recent Products sold

90001
Galaxy S22

90001
Galaxy S22

90003
Galaxy Tab 5e

More
[3 / 13]