Extend Fiori Elements List Report Object Page Application

Task 1: Create a Fiori elements Extension application **

Create a new Fiori Application using the Template Wizard for a *List Report*

Field	Value
Data source	Connect to a System
System	abap-cloud-default_xx-dev (BTP)
Service	ZUI_WKSP_##
Main entity	ListObject
Module name	extend-object
Application title	Extend List Object
Application namespace	nato.workshop
Description	Extend List Object
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	ExtendList
Title	Extend List Object
Subtitle	Workshop

- Entity Selection

  Entity Selection
  Configure the selected service.

  Main entity *

  ListObject

  Automatically add table columns to the list page and a section to the object page if none already exists?
  ◉ Yes ○ No

- Project Attributes

  # Project Attributes

  Configure the main project attributes.

  Module name ⑦ *

  extend-object

  Application title ⑦

  Extend List Object

  Application namespace ⑦

  nato.workshop

  Description ⑦

  Extend List Object

  Project folder path *

/home/user/projects

Minimum SAPUI5 version ⑦ *

1.96.7 (Source system version)

Add deployment configuration

🔘 Yes ⚪ No

Add FLP configuration

🔘 Yes ⚪ No

Configure advanced options

⚪ Yes 🔘 No

- Fiori Launchpad Configuration

Fiori Launchpad Configuration

Configure Fiori Launchpad settings

Semantic Object *

NATO

Action *

ExtendList

Title *

Extend List Object

Subtitle (optional)
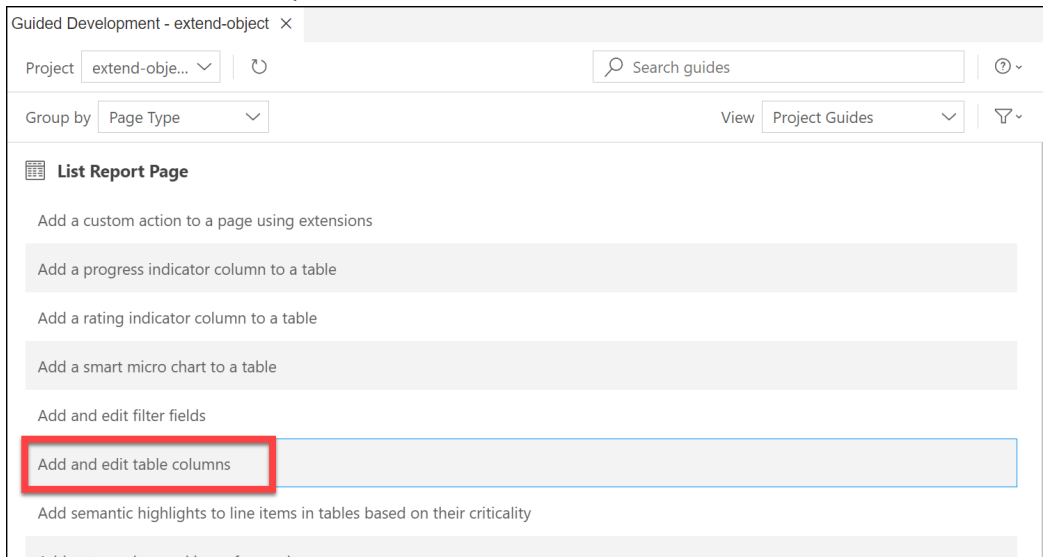
Workshop

- Preview Application

Task 2: Add extra column to Table

- Start the *Guided Development* and select **Add and edit table columns**

- Select **ListObjectType** as the *Entity Type*
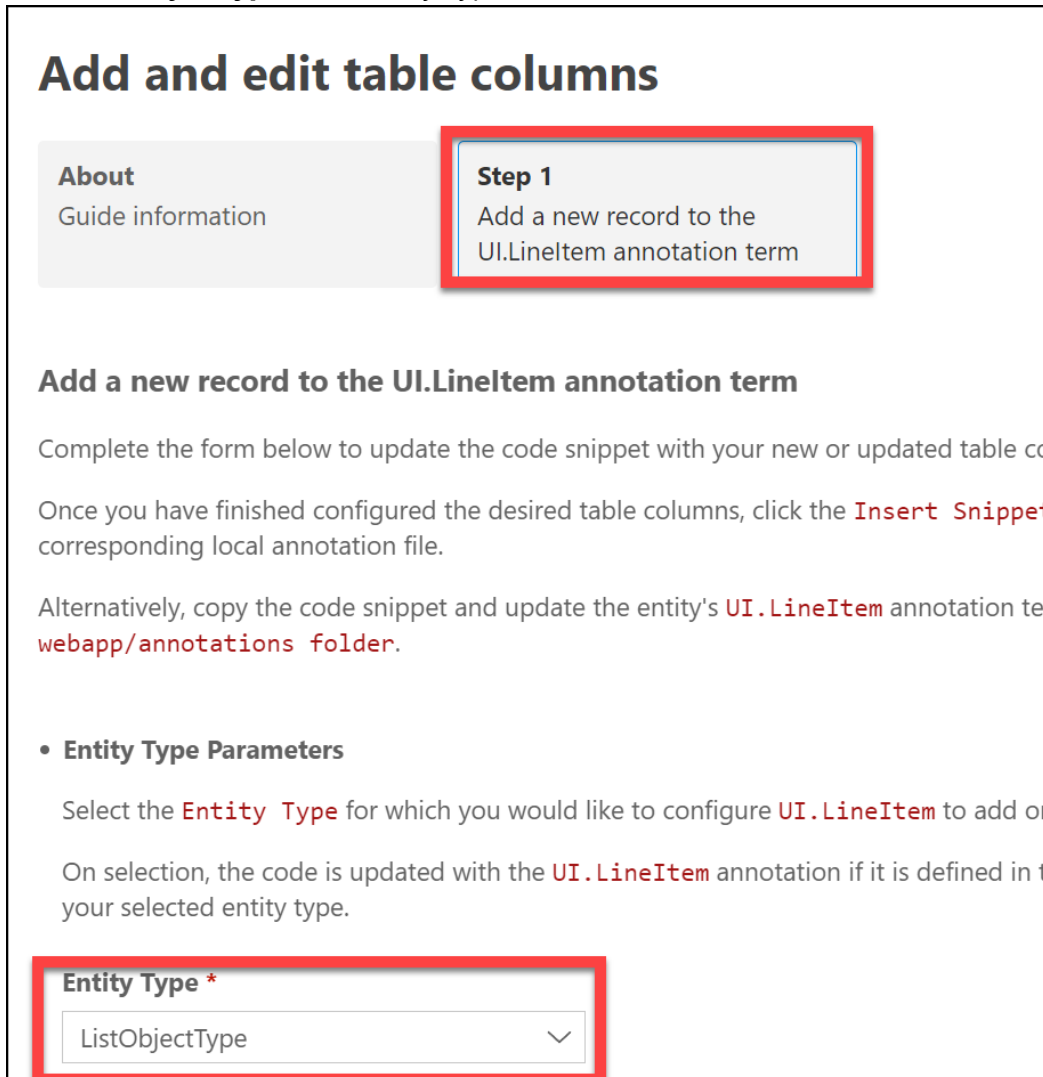
- Press button *Add Column*

- Enter **NetAmmount** as the *Property* and *Insert Snippet*

- Preview Application

Task 3: Add a custom Action

- Start the *Guided Development* and select **Add a custom action to a page using extensions**

- *Start Guide*
- Select **List Report Page** in *Page*, enter **onCalculate** as *Function Name* and Press *Insert Snippet* followed by *Next*

Add a custom action to a page using extensions

About	Step 1	Step 2
Guide information	Add a function in controller for the action	Update the manifest

Add a function in controller for the action

In this step, you will create a controller extension to define a handler function for the custom action button.

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

- **Page Parameters**

  Select the **Page** to which you want to attach new custom action.

  Page *

  List Report Page

- **New Function Parameters**

  Enter a **Function Name** that will be the name of your new custom function.

  Function Name *

  onCalculate

  Insert Snippet   Copy   Reset
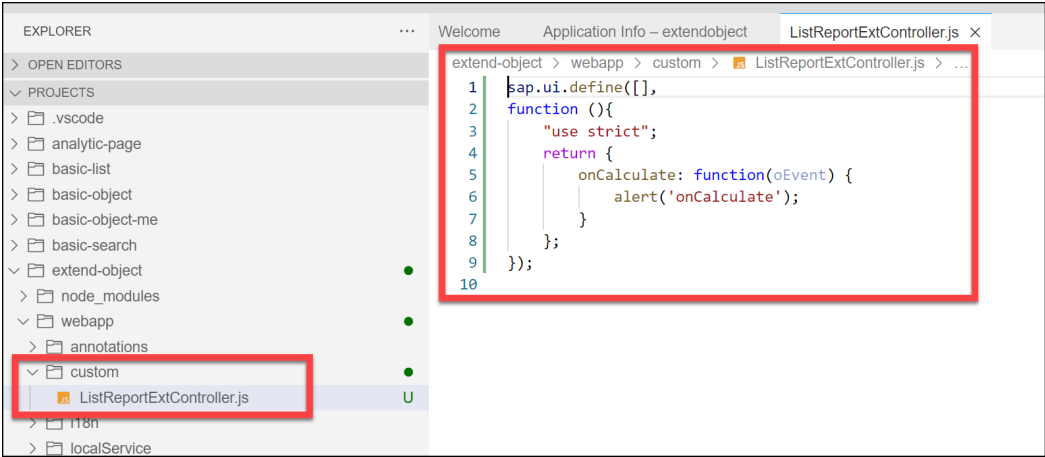
```
1   sap.ui.define([],
2   function (){
3       "use strict";
4       return {
5           onCalculate: function(oEvent) {
6               alert('onCalculate');
7           }
8       };
9   });
10
```

‹ Back   Next ›   Exit Guide

- This will add a new *folder* **custom** and new *file* **ListReportExtController.js** with a template snippet for the javascript code

- At the next step set the following parameters, *Insert Snippet* and *Exit Guide*

Name	Value
Entity Set	ListObject
Action Position	Table Toolbar
Action ID	Calculate
Button Text	Calculate Difference
Row Selection	yes

Add a custom action to a page using extensions

About	Step 1	Step 2
Guide information	Add a function in controller for the action	Update the manifest

Update the manifest

In this step, you will add the extension settings of the custom action to the manifest.

- **Entity Set Parameters**

  Select the `Entity Set` in which you would like to add your new custom action.

  Entity Set *

  | ListObject ⌄ |

- **New Actions Parameters**

  New Actions Parameters

  Action Position *

  | Table Toolbar ⌄ |

  Action ID *

  | Calculate |

  Button Text *

  | Calculate Difference |

  Row Selection

  | yes ⌄ |

  [Insert Snippet] [Copy] [Reset]

```
1   {
2       "sap.ui5": {
3           "routing": {
4               "targets": {
5                   "ListObjectList": {
6                       "options": {
7                           "settings": {
8                               "controlConfiguration": {
9                                   "@com.sap.vocabularies.UI.v1.LineItem": {
10                                      "actions": {
11                                          "Calculate": {
12                                              "id" : "CalculateButton",
13                                              "text" : "Calculate Difference",
14                                              "press" : "nato.workshop.extendobject.custom.L
```

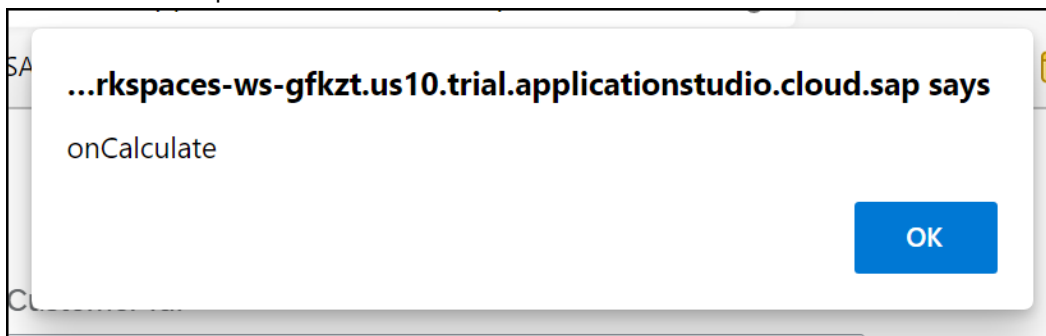[< Back] [Next >] [Exit Guide]

- Manifest.json file is updated

- Preview Application

- Select a line and press the **Calculate Difference** button

- Now implement the javascript template in the new **ListReportExtController.js**, Calculate the difference between de *GrossAmount* and *NetAmount* and show it using a MessageBox

```
sap.ui.define([
    "sap/m/MessageBox"
],
function (MessageBox){
    "use strict";
    return {
        onCalculate: function(oEvent) {
            let nGrossAmount =
parseFloat(this.getSelectedContexts()[0].getObject().GrossAmount);
            let nNetAmount = parseFloat(this.getSelectedContexts()
[0].getObject().NetAmount);
            let nDiff = nGrossAmount - nNetAmount;
            MessageBox.success("Difference = " + nDiff);
        }
    };
});
```