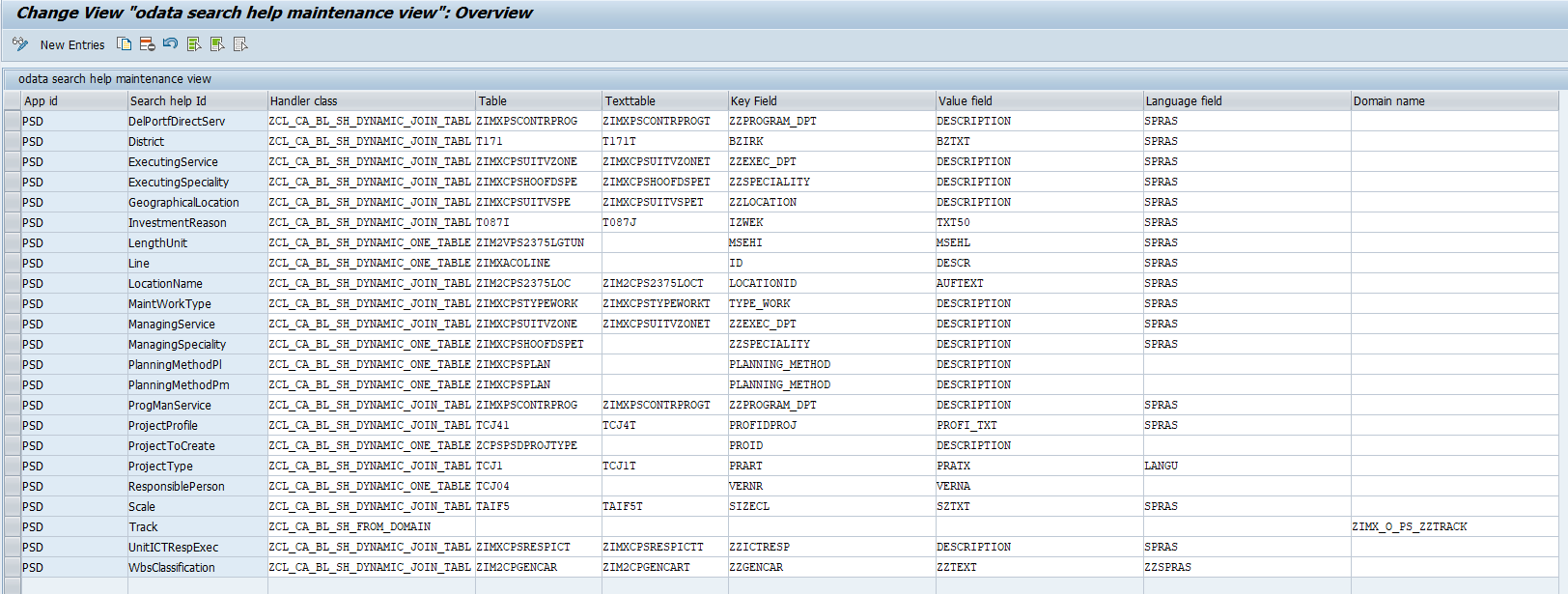
# Capabilities :

The reuse generic search help offers a simplified way to use combo box, select and input field with suggestion/search help in view.

It manage the data loading. All list needed must be initially defined in the backend trough maintenance view ZV\_CASEARCHHELP. In this view, per Application Id, we can define different list. It can be a simple table selection, or a combination between table and text table or domain value or even a call to a provider class ( for more complex selection )



There are two modes of loading. If list is tiny and used in combo box/select (+/-200 entries), the whole list is loaded from the backend and store in a local JSON model. If the list is longer ( > 200) it is managed as for an usual oData model (data loaded by interval / search and filter in the backend ).

The reusable component builds and handles the search help fragment. All corresponding function (close/search) is also fully managed by the reusable component.

Initialization, suggestion process and change on an input is processed fully by the reusable component. No additional function is needed in the application controller.

A Demo application that uses the reusable component can be found in GIT under <https://git/sapfiori/Libraries/projectsystemreuselibrary/tree/master/genericSearchDemoApp>

* Git clone the whole project https://git/sapfiori/Libraries/projectsystemreuselibrary.git
* Launch the index.html stored in genericSearchDemoApp directory

# How to install in your app :

It consists to :

* Declaration of the component usage in the manifest.
* Add a js file that make the middleware between the reusable component and the using application.
* Declaration in every controller of all view where the reuse search help will be used.

1. **Manifest component usage ( under "sap.ui5" )**

Properties:

appId 🡪 the application Id for the backend customising ( ex : PSD )

localModelName 🡪 the name of the model that will contains all locally loaded list

backendModelName 🡪 the name of the model linked with the backend

searchHelpToLoad 🡪 list of search help id that must be loaded locally

"componentUsages": {

            "dynamicValueHelpUsage": {

                "name": "be.infrabel.ps.reuse.dynamicSearchHelpData",

                "componentData": {

                    "appId": "PSD",

                    "localModelName": "searchHelp",

                    "backendModelName": "BackendValue",

                    "searchHelpToLoad": [

                        "GeographicalLocation",

                        "ManagingService"

                    ]

                }

            }

        }

1. **Instantiate the value help tool class in component:**
2. Copy the ValueHelpTool.js from *be/Infrabel/ps/reuse/dynamicSearchHelpData* and store it in your project under Utilities ( for ex.)
3. Declare the valueHelpTool in component

sap.ui.define([

    "sap/ui/core/UIComponent",

    "sap/ui/model/json/JSONModel",

        "<your\_namespace>/utilities/ValueHelpTool"

], function (UIComponent, JSONModel, ValueHelpTool) {…

1. Instantiate the tool in the Init function of component :

this.\_valueHelpTools = new ValueHelpTool(this, "dynamicValueHelpUsage")

      .then(function (oValueHelpComponent)

                // initlize router -> view will be called only when value help component is loaded

           this.getRouter().initialize();

       }.bind(this))

       .catch(function (oError) {

                MessageToast.show(oError.message);

        }.bind(this));

1. **Define valueHelpTool in base controler or controller for usage in view**

All input controls that will use the reusable component will use some event. Functions that manage these events are declared in the valueHelpTool instance. In order to be able to use these function, the valueHelptool must be defined in controller of the view.

sap.ui.define([

    "sap/ui/core/mvc/Controller",

    "sap/ui/core/routing/History",

    "sap/ui/model/json/JSONModel",

    "<your\_namespace>/utilities/ValueHelpTool"

], function (Controller, History, JSONModel, ValueHelpTool) {

    "use strict";

    var oBaseController = Controller.extend("<your\_namespace>.controllers.BaseController", {

        ValueHelpTool: new ValueHelpTool()

    });

1. **How to use in View :**

With sap.m.Input control :

Custom data are used to send some settings to the reusable search help In order to adapt the search help to the input :

* searchHelpId 🡪 the search help id to use with the input
* displayMode 🡪 if = “key” , only the key is displayed in the input field, if = ‘value”, only the description is shown in the input field. If empty or any other value -> key + description is shown in the input field
* listWithKey 🡪 if true, the key is displayed in the search help result list .This is optional, default = false
* Title 🡪 the title for the searchhelp popup. This is optional. If not given, the text of the label attached to the input is taken. Note : if input is used in a form or simple form, the label is automatically linked with the input control, otherwise the label must be linked to the input control through ‘labelFor’ association.

<Input type="Text"

showValueHelp="true/false"

valueHelpRequest=".ValueHelpTool.handleEvent" not required if showValueHelp ="false"

valueHelpOnly=”true/false”

change=".ValueHelpTool.handleEvent" not required if valueHelpOnly ="true"

showSuggestion="true/false"

suggest=".ValueHelpTool.handleEvent" not required if showSuggestion="false"

                suggestionItemSelected=".ValueHelpTool.handleEvent" not required if showSuggestion="false"

modelContextChange=".ValueHelpTool.handleEvent" used to initialize the control value corresponding with the key

selectedKey="<the path to the key in your model>">

<customData>

<core:CustomData key="searchHelpId" value="<the search help id to use>" />

<core:CustomData key="displayMode" value="key/value/…" />

<core:CustomData key="listWithKey" value="true/false" />

<core:CustomData key="title" value="<the search help title> " />

</customData>

</Input>

For locally uploaded list, data are recorded in a JSON model. The name of the model is defined in the manifest when declaring the reusable component usage:

"componentUsages": {

"dynamicValueHelpUsage": {

"name": "be.infrabel.ps.reuse.dynamicSearchHelpData",

"componentData": {

"appId": "PSD",

"localModelName": "searchHelp",

"backendModelName": "BackendValue",

"searchHelpToLoad": [

"GeographicalLocation",

"LengthUnit",

"District",

"Track"

]

}

}

}

The JSON model contains all list stored under their search help id.

You can use that model directly in a combo box or Select :

                   <ComboBox items="{ path: 'searchHelp>/Track', sorter: { path: 'key' } }"

                            selectedKey="{main>Track}">

                            <core:Item key="{searchHelp>key}" text="{searchHelp>key} {searchHelp>value}" />

                        </ComboBox>

1. **Events :**

**The reusable component triggers 3 different events :**

1. **dataLoaded :**

**This event is raised when local data are completely loaded during component initialization**

1. **dataError:**

**This event is raised when an error occurs at local data are loading during component initialization**

1. **itemSelected:**

**This event is raised when data are selected in the input control ( by suggestion/be seachhelp selection, by direct input ). The event have three parameters:**

**control: the input control instance**

**setting: the setting defined for this input control**

**selecteItem: the key and value selected**