Date: 29-SEP-2023

LIST WITH SAMPLE DATA

 Version: 1.2.1

**1. OVERVIEW**

A list of items, coupled with a details page per item, is a popular UX pattern. The component offers configurable list and detail views, with prepackaged sample data, fetched from Volt MX Foundry, which can be easily modified to talk to production data.

1. **Use Case:**

You can change the text on the header and filter the list based on the year or the mileage values. You can also configure the Object Service to add and display your own data on the component. You can use this component in scenarios where you want to test any data in your app.

**B. Percentage of re-use:**

       90%

**2. GETTING STARTED**

1. **Prerequisites**

             Before you start using the List with Sample Data component, ensure the following:

* [HCL Foundry](https://manage.hclvoltmx.com/)
* Volt MX Iris

1. **Platforms Supported**
2. Mobile
3. *iOS*
4. *Android*
5. **Importing the Component**

You can import the Forge components only into the apps that are of the Reference Architecture type.

**To import the List with Sample Data component, do the following:**

1.Open your app project in Volt MX Iris.

2.In the Project Explorer, click the **Templates** tab.



3.Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.



4.Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.



**3.REFERENCES**

 **A. Dynamic Usage**

              Create a function called createComponent() and write the code inside it to create and con figure the component .You can refer to the given sample code for more information

In the **Project Explorer**, on the **Projects** tab, click **Controllers** section to access the respective **formController**. Create a method and implement the code snippet like the sample code mentioned below.

/\* creating a component's Object \*/

define({

 onNavigate: function(){

 this.createComponent();

 },

createComponent: function()

{

/\* Creating the component's object \*/

var listSampleData = new com.voltmxmp.listsampledata(

{

 "clipBounds": true,

 "height": "100%",

 "id": "listSampleData",

 "isVisible": true,

 "left": "0dp",

 "top": "0dp",

 "width": "100%",

 "zIndex": 1

}, {}, {});

/\* Setting the text for the header \*/

listSampleData.listHeaderText = "CARS FOR SALE";

/\* Mapping the back-end data to the component \*/

listSampleData.dataMapping =

{

data:

[

{

 "widgets":"Label1",

 "Display\_Names":"Model",

 "Backend\_Fields":"model"

},

{

 "widgets":"Label2",

 "Display\_Names":"Price",

 "Backend\_Fields":"Price"

},

{

 "widgets":"Label3",

 "Display\_Names":"Mileage",

 "Backend\_Fields":"Mileage"

},

{

 "widgets":"Label4",

 "Display\_Names":"Year",

 "Backend\_Fields":"Year"

},

{

 "widgets":"Image",

 "Display\_Names":"none",

 "Backend\_Fields":"Image\_URL"

}

]};

/\* Adding the component to the Form \*/

this.view.add(listSampleData);

/\* Fetching data from voltmx foundry\*/

this.view.listSampleData.fetchAllDetails();

},

});

Now, you need to call this function using **Actions**. For more information, refer to the [Add Actions](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/index.html) section of the Iris User Guide

1. **Save** the file.
2. **Setting Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the list **with sample data** component. You can set the properties directly on the **Component** tab or by writing a JavaScript.

* **General Properties**

## Properties

You can use a component's Properties to customize and configure the elements. These elements can be UI elements, service parameters, and so on. For more information about properties, you can refer to the [Components Overview](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/index.html) section of the Iris User Guide.

You can set the properties from the Iris Properties panel on the right-hand side. You can also configure these properties using a JavaScript code.

#### **List View**

* **List Header Text**

|  |  |
| --- | --- |
| **Description:** | Specifies the text that you want to display on the header of the component. |
| **Syntax:** | listHeaderText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Default Value:** | "CARS FOR SALE" |
| **Example:** | this.view.componentID.listHeaderText = "CARS FOR SALE"; |

## C. Events

The component invokes events when its corresponding action is performed. You can configure any logic you want the component to perform whenever an event occurs. You can configure the events directly on the Actions tab or by writing a JavaScript, For more information, refer to [Add Actions](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/index.html) in the Iris User Guide.

* **onListItemClick**

|  |  |
| --- | --- |
| **Description:** | Invoked when the user selects an item from the list. |
| **Syntax:** | onListItemClick |
| **Parameters:** | *listItemObject [JSON]* :Contains information about the list item. |
| **Example:** | this.view.componentID.onListItemClick = function(listItemObject){ alert("List Item Selected: "+JSON.stringify(listItemObject));}.bind(this); |

* **onErrorCallback**

|  |  |
| --- | --- |
| **Description:** | Invoked when an error occurs in the component. |
| **Syntax:** | onErrorCallback |
| **Parameters:** | *error [JSON]* :Contains information about the error, such as the error code, and error message. |
| **Example:** | this.view.componentID.onErrorCallback = function(error){ alert("Error: "+JSON.stringify(error));}.bind(this); |

## D. APIs

The following APIs pertain to the List with Sample Data component:

* **filterDataByField**

|  |  |
| --- | --- |
| **Description:** | Filters the list based on a range of the **Year** or the **Mileage** values. |
| **Syntax:** | filterDataByField(filterObject) |
| **Parameters:** | *filterObject [JSON]* :Contains the key and values that the component uses to filter the data. |
| **Return Value:** | None |
| **Example:** | var filterObject ={ "key":"Year", "min":2010, "max":2020}this.view.componentID.filterDataByField(filterObject); |

* **fetchAllDetails**

|  |  |
| --- | --- |
| **Description:** | Fetches the data from the Object Service on Volt MX Foundry. |
| **Syntax:** | fetchAllDetails() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | Use this API to fetch the data after you create the component dynamically. |
| **Example:** | this.view.componentID.fetchAllDetails(); |

**4.REVISION HISTORY**

 App version 1.2.1:

1. **Limitations**

To change the height of the component, you first need to change the height of the **row Template** and then the height of the component. Otherwise, the UI of the component appears distorted.