28 Mar 2023

**Refresh ANIMATION (GENERIC)**

**VERSION: 1.0.2**

# **Overview**

The Refresh Animation (Generic) component is a user-defined animated UI used to fill the space when a widget is pulled down to refresh the contents in the form.

## **Use case:**

Consider a scenario where you want to develop an online hotel booking app that displays a list of hotels based on the selected location and specified price range. In this app, you can utilize the Refresh Animation (Generic) component to fill the available space when the list is pulled down to refresh the accommodation fares. The Refresh Animation component can be hidden when the list refresh is complete.

## **Percentage of re-use:**

80-90%.

# **Getting Started**

## **Prerequisites**

Before you start using the Refresh Animation (Generic) component, ensure you have the following:

• [HCL Foundry](https://manage.hclvoltmx.com/)

• Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

## **Importing the app**

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

## **To import the Refresh Animation (Generic) component, do the following:**

## Open your app project in Volt MX Iris.

## In the Project Explorer, click the **Templates** tab. Graphical user interface, text, application Description automatically generated

## Right-click Components, and then select Import Component. The Import Component dialog box appears. Graphical user interface, text, application, Teams Description automatically generated

## 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.

## A screenshot of a computer Description automatically generated

Once you have imported a component to your project, you can easily add the component to a form. For more information, refer Add a Component to a Form

## **Building and previewing the app**

After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the Building and Viewing an Application section of the Volt MX User Guide.

You can then run your app to see the Refresh Animation (Generic) work in real time.

# **References**

## **Dynamic Usage**

You can also add **Refresh Animation (Generic)** component dynamically. To do so,

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

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

var componentInstance = new com.voltmxmp.genericrefresh({

"autogrowMode": voltmx.flex.AUTOGROW\_NONE,

"clipBounds": true,

"height": "30%",

"id": "genericrefresh",

"isVisible": true,

"layoutType": voltmx.flex.FLOW\_VERTICAL,

"left": "0%",

"masterType": constants.MASTER\_TYPE\_USERWIDGET,

"skin": "slFbox",

"top": "0%",

"width": "106.67%",

"zIndex": 1

}, {}, {});

/\*Adding the Refresh Animation Generic Component to the form\*/

this.view.add(componentInstance);

In the code snippet, you can edit the properties of the component as per your requirement. For more information, see Setting Properties.

2. Save the file

## **Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Refresh Animation (Generic) component. You can set the properties directly on the **Component** tab or by writing a JavaScript. This section provides information on how to set properties by writing a JavaScript.

**General Properties**

## **Skin**

## You can select skins from the **Exposed Skins** drop-down list on the **Skin** tab. This section provides information on how to set Skin by writing a JavaScript.

**1. [Background Skin](javascript:void(0);)**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin to be set as the background of the component. |
| **Syntax:** | backgroundSkin |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |
| **Example:** | this.view.componentInstance.backgroundSkin = "skinid"; |

## **B. APIs**

The following API pertains to the Refresh Animation (Generic) component:

### **i.** [**show**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | The API displays the refresh animation. |
| **Syntax:** | show() |
| **Parameter:** | None |
| **ReturnValue:** | None |
| **Example:** | this. view.componentInstance.show(); |

### **ii. hide**

|  |  |
| --- | --- |
| **Description:** | The API hides the refresh animation. |
| **Syntax:** | hide() |
| **Parameter:** | None |
| **ReturnValue:** | None |
| **Example:** | this. view.componentInstance.hide(); |

### **iii.** [**isShown**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | The API notifies whether the refresh animation is displayed or not. |
| **Syntax:** | isShown() |
| **Parameter:** | None |
| **ReturnValue:** | Boolean |
| **Example:** | var returnValue = this.view.componentInstance.isShown() |

**Sample Usage of the Component**

### **Component with the Segment Widget**

Consider a use case where user pulls down a list to update the list data. You can use the Refresh Animation component to fill the space when the list (segment widget) is pulled down to refresh the data. You can achieve the use case by following the below procedure.

**To use the component in the Segment Widget, do the following:**

1. Import the Refresh Animation (Generic) component.
2. Add a Flex Container Widget to the form.
3. Set the **Layout Type** of the FlexContainer Widget to **Flow Vertical**.
4. Drag the Refresh Animation(Generic) component and the Segment Widget onto the FlexContainer Widget.
5. Set the left and top margin values of the Refresh Animation (Generic) and segment widget component to required percent.
6. In the Project Explorer, on the Projects tab, click the context menu arrow of Controllers under the respective channel, and then click FormController. The FormController JavaScript file opens in the Code Editor.
7. Copy and paste the following code snippet in the Action editor:

define({

  onNavigate: function(){

     this.view.seg1.scrollingEvents ={

       onPull: this.PostShow.bind(this)

     };

  },

  PostShow : function(){

    this.view.genericrefresh.setVisibility(true);

    this.view.genericrefresh.show();

    this.view.seg1.top = "20%";

    this.view.flx1.forceLayout();

    voltmx.timer.schedule("timer4",this.hide,5, false);

  },

  hide:function(){

    this.view.genericrefresh.setVisibility(false);

    this.view.seg1.top = "7.5%";

    this.view.flx1.forceLayout();

  }

});

The above code displays the Refresh animation component on triggering the onPull event of the Segment Widget.

8. Save the file.

### **Component with the FlexContainer Widget**

To use the component in the FlexContainer Widget, do the following:

1. Import the Refresh Animation (Generic) component.
2. Drag the Refresh Animation (Generic) component and the FlexContainer Widget onto the Form.
3. Set the left and top margin values of the Refresh Animation (Generic) component to Zero percent.
4. Set the left and top margin values of the FlexContainer Widget to Zero percent.
5. Define the onTouchStart, onTouchMove, and onTouchEnd events of the Form as shown in following code snippet:

/\*Call the methods in the forms Action editor\*/  
/\*onTouchStart of the Form \*/

this.onStart(y);  
/\*onTouchMove of the Form \*/

this.onMove(y);  
/\*onTouchEnd of the Form \*/

this.onEnd();

1. In the Project Explorer, on the Projects tab, click the context menu arrow of Controllers under the respective channel, and then click FormController. The FormController JavaScript file opens in the Code Editor.
2. Copy and paste the following code snippet in the FormController JavaScript file:

/\* Define methods in the Form Controller \*/

define({

yCoordinates: "0",

\_top: "0",

\_started: 0,

onStart: function (yy) {

this.yCoordinates = yy;

if (this.\_started === 0) {

this.view.componentInstance.show();

this.\_started = 1;

}

},

onMove: function (y) {

this.\_top = y - this.yCoordinates;

if (this.\_top > 0) {

this.view.flexContainer.top = this.\_top;

} else {

this.view.flexContainer.top =   
parseInt(this.view. componentInstance.height) + this.\_top + "dp";

}

},

onEnd: function () {

if (this.\_top >= 0) {

this.view.flexContainer.top = this.view.componentInstance.height;

} else {

this.view.flexContainer.top = "0dp";

this.view.componentInstance.hide();

this.\_started = 0;

}

}

});

The above code displays the Refresh Animation (Generic) component on triggering the Form events.

1. Save the file.

# **Revision History**

**App version 1.0.2:**