28 Mar 2023

**Refresh ANIMATION (ROTATING OBJECTS)**

**VERSION: 1.0.1**

# **Overview**

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

## **Use case:**

Consider a case that you want to develop an entertainment app that helps a user to access latest songs. In this app, you can add the Refresh Animation (Rotating Objects) component to fill the space when a list (or page) is pulled down to update the play-list. Using the APIs, you can hide the Refresh Animation component when the list refresh is complete.

The Refresh Animation (Rotating Objects) component is a ready-to-use component. You can import the Refresh Animation (Rotating Objects) component into your app (created in Voltmx iris), and achieve the Refresh Animation (Rotating Objects) feature without developing it from scratch. The Refresh Animation (Rotating Objects) component also facilitates a set of properties and APIs that helps you customize the UI elements and the functionality.

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

80-90%.

# **Getting Started**

## **Prerequisites**

Before you start using the Refresh Animation (Rotating Objects) 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 (Rotating Objects) 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.

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 (Rotating Objects) work in real time.

# **References**

## **Dynamic Usage**

You can also add **Refresh Animation (Rotating Objects)** 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 refreshcircle = new com.voltmxmp.refreshcircle({

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

"clipBounds": true,

"height": "16.49%",

"id": "refreshcircle",

"isVisible": true,

"layoutType": voltmx.flex.FREE\_FORM,

"left": "0dp",

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

"skin": "voltmxmprcsknFlxBackground20223C",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

refreshcircle.backgroundSkin = "sknFlxBackground20223C";

refreshcircle.durationOfAnimation = "1";

this.view.add(refreshcircle);

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 (Rotating Objects) 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**

**1.** **ANIMATION SPEED**

|  |  |
| --- | --- |
| **Description:** | Specifies the duration of the refresh animation. The duration value is specified in seconds. |
| **Syntax:** | durationOfAnimation |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | The property throws an exception if you provide an invalid value. You must handle the exception. |
| **Example:** | this.view. refreshcircle. durationOfAnimation = "1"; |

## **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.

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

|  |  |
| --- | --- |
| **Description:** | Specifies the skin to be set as the background of the component. |
| **Syntax:** | backgroundSkin |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. Changes made to the skin are reflected only during run time, but not during the design time |
| **Example:** | this.view. refreshcircle.backgroundSkin = "skinname"; |

**B. IMAGE SOURCE**

**3.** [**Center Circle Source**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the center circle during the refresh animation. |
| **Syntax:** | centerCircleSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | * Ensure that the image file exists in the *workspace\resources\common* directory. * Do not add any uppercase characters in the file name of the image. * Ensure that the image is in PNG format. * Specify the file name of the image along with the extension. |
| **Example:** | this.view. refreshcircle.centerCircleSource = "centercircle.png"; |

**4.** [**Small Circles Source**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the small circles during the refresh animation. |
| **Syntax:** | smallCircleSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before setting the property:   * Ensure that the image file exists in the *workspace\resources\common* directory. * Do not add any uppercase characters in the file name of the image. * Ensure that the image is in PNG format. * Specify the file name of the image along with the extension. |
| **Example:** | this.view.refreshcircle.centerCircleSource ="smallcircles.png"; |

**5.** [**Large Circles Source**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the large circles during the refresh animation. |
| **Syntax:** | largeCircleSource |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before setting the property:   * Ensure that the image file exists in the *workspace\resources\common* directory. * Do not add any uppercase characters in the file name of the image. * Ensure that the image is in PNG format. * Specify the file name of the image along with the extension. |
| **Example:** | this.view.refreshcircle.largeCircleSource ="bigcircles.png"; |

## **C. APIs**

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

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

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

### **ii. hide**

|  |  |
| --- | --- |
| **Description:** | The API hides the refresh animation. |
| **Syntax:** | hide() |
| **Parameter:** | None |
| **ReturnValue:** | None |
| **Example:** | this. view. refreshcircle.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. refreshcircle.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 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 component and the Segment Widget onto the FlexContainer Widget.
5. Set the left and top margin values of the Refresh Animation 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 Form Controller:

define({

onNavigate: function(){

this.view.segment.scrollingEvents ={

onPull: this.PostShow.bind(this)

};

},

PostShow : function(){

this.view.refreshcircle.setVisibility(true);

this.view.refreshcircle.show();

this.view.segment.top = "23.9%";

this.view.flxmain.forceLayout();

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

},

hide:function(){

this.view.refreshcircle.setVisibility(false);

this.view.segment.top = "8%";

this.view.flxmain.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 (Rotating Objects) component.
2. Drag the Refresh Animation (Rotating Objects) component and the FlexContainer Widget onto the Form.
3. Set the left and top margin values of the Refresh Animation (Rotating Objects) 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 (Rotating Objects) component on triggering the Form events.

1. Save the file.

# Revision History

**App version 1.0.1:**