4 Oct 2023

Image Zoom component(1.0.0)

# Overview

The Image Zoom component provides image zooming functionality. It takes remote URL as an input and caters pinch zoom in/out and double tap zoom in/out features.

## Use case

1. a Photo viewing app requires to provide the Zoom in/out functionality to see the images into more details.

2. In E commerce apps, to view the product image to minute details this component helps in zoom in/out the image.

B .Percentage of re-use:

Approximate 90% of reuse. It sets an expectation of how much can be used out of the box, and how much needs to be customized for a specific app.

C. Features

1. The component does not use any third party library, it is purely written in native.
2. Uses NFI 3.0

2. Getting Started

## A. Prerequisites

Before you start using the Image Zoom component, ensure the following:

• Volt MX Iris

## Platforms Supported

### Mobile

#### iOS

#### Android

### Tablets

## Importing the app

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

## **To import the Image Zoom component, do the following:**

## Open your app project in Volt MX Iris.

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

Graphical user interface, text, application

Description automatically generated

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

Graphical user interface, text, application, Teams

Description automatically generated

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.

## 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](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html" \l "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](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud) section of the Volt MX User Guide.

You can then run your app to see the Image Zoom work in real time.

# 3. References

## Dynamic Usage

You can also add **Image Zoom** component dynamically. To do so,

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



/\* Creating Image Zoom component instance \*/

var ImageZoom = new com.hcl.ImageZoom ({

"height": "100%",

"id": "Image Zoom",

"isVisible": true,

"left": "0dp",

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

"isModalContainer": false,

"skin": "slFbox",

"top": "0dp",

"width": "100%",

"zIndex": 1,

"overrides": {

"ImageZoom": {

"right": "viz.val\_cleared",

"bottom": "viz.val\_cleared",

"minWidth": "viz.val\_cleared",

"minHeight": "viz.val\_cleared",

"maxWidth": "viz.val\_cleared",

"maxHeight": "viz.val\_cleared",

"centerX": "viz.val\_cleared",

"centerY": "viz.val\_cleared"

}

}

}, {

"overrides": {}

}, {

"overrides": {}

});

/\*Adding the Image Zoom component to a Form\*/

this.view.add(ImageZoom);

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

4.Configuring Deployment Target - iOS

1. From the left navigation menu, click **Project Settings**.
2. In the **Project Settings** window, go to **Native** → **iPhone/iPad**.
3. Under **Target Versions**, from the **iOS Version** list, select **11.0** or higher.

A screenshot of a computer

Description automatically generated

Configuring Deployment Target - Android

1. From the left navigation menu, click **Project Settings**.
2. In the **Project Settings** window, go to **Native** → **Android Mobile/Tablet**.
3. Under **SDK Versions**, from the **Minimum SDK** list, select **21.0** or higher.

### A screenshot of a computer Description automatically generated

## 5. API’s

### i. updatePlaceHolderImage

|  |  |
| --- | --- |
| **Description:** | This API meant to set the place holder image. This image will be reflected when actual remote image fails to load. It accepts only local images as a placeholder. |
| **Syntax:** | updatePlaceHolderImage() |
| **Parameters:** | String - Image name(image must be placed at following locations  Android - /resources/mobile/native/android/drawable  iOS - /resources/mobile/native/iphone/mainbundle  ) |
| **Return Value:** | None |
| **Remarks:** | Image name(parameter) must be passed without extensions like .png or .jpeg |
| **Example:** | this.view.ImageZoom.updatePlaceHolderImage(“placeholder”); |

### ii. registerCompletionCallback

|  |  |
| --- | --- |
| **Description:** | Registers the callback to get an event when image loading is completed, this helps control the loading indicators. In this callback, it is preferred to remove the loading indicator. |
| **Syntax:** | registerCompletionCallback(function(status){}) |
| **Parameters:** | Callback |
| **Return Value:** | None |
| **Remarks:** | It returns the status flag in the callback parameter, based on that it can determined the status of image loading into the component. |
| **Example:** | this.view.ImageZoom.registerCompletionCallback(function(status) {  }); |

### 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| iii. imageWithURL  |  |  | | --- | --- | | **Description:** | This API accepts the remote Image URL, and loads that image into the component.  And loaded image can be performed zoom in/out operations. | | **Syntax:** | imageWithURL(“https://image.com/imagename”) | | **Parameters:** | String – Remote image address | | **Return Value:** | None | | **Remarks:** | None | | **Example:** | this.view.Image.imageWithURL(“https://image.com/imagename”); | |  |

App version 1.0.1:

## Limitations

B. Known Issue