Date : 05-Feb-24

Skeleton Screen Template - Segment

version: 1.0.2

1. **OVERVIEW**

The Skeleton Screen Template - Segment component displays a page that gradually loads information. The component creates a wireframe of the page that contains labels for text and images.

skeleton screen Template - Segment benefits the user by indicating progress, thereby shortening the perceived waiting time. You can use a skeleton screen while creating a page that loads a list of entries from backend services and to indicate the wait time of the service call.

## **Use case:**

A Skeleton screen Template - Segment can be used as an alternative to a spinner to indicate loading progress. It gives a visual cue that things are happening immediately the skeleton of the page has been loaded and the information is gradually loading.

##  **Features:**

* Useful component when a screen takes a long time to load
* Appears if a screen is loading

## **C. 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.

# **Getting Started**

## **Prerequisites**

Before you start using the Skeleton Screen Template - Segment component, ensure the following:

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

## **Platforms Supported**

###  Mobile

### Android

### iOS

### Tablets

###  PWA

## **Importing the Component**

##  **To import the** **Skeleton Screen Template - Segment component, do the following:**

## Open your app project in Volt MX Iris.

## 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.



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%22%20%5Cl%20%22add-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 VoltMX Iris User Guide.

You can then run your app to see the Skeleton Screen Template - Segment work in real time.

# **References:**

## **A. Dynamic Usage of Skeleton Screen Template - Segment component**

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 like the sample code mentioned below.

|  |
| --- |
| define({  //Type your controller code here  serviceName : "BestBuySkeletonScreen", productsOperationName : "getProducts", currentPage: 1, onNavigate: function(){ var productId ="pcmcat378600050009"; this.view.postShow=this.postShow(productId); var skeletonScreen = new com.voltmx.skeletonSegment( { "autogrowMode": voltmx.flex.AUTOGROW\_NONE, "clipBounds": true, "height": "100%", "id": "skeletonScreen", "isVisible": true, "layoutType": voltmx.flex.FREE\_FORM, "left": "0%", "masterType": constants.MASTER\_TYPE\_USERWIDGET, "skin": "slFbox", "top": "0%", "width": "100%" }, {}, {}); this.view.add(skeletonScreen);  this.view.skeletonScreen.loadMore = this.segReachEnd.bind(this, productId);  }, postShow : function(productId) { if (voltmx.net.isNetworkAvailable(constants.NETWORK\_TYPE\_ANY)) { var integrationSvc = KNYMobileFabric.getIntegrationService(this.serviceName); var headers = {}; data = {"categoryid" : productId , "pageNo": this.currentPage, "pageSize": 10};  integrationSvc.invokeOperation(this.productsOperationName, headers, data,this.initLoadSuccesscallback, this.initLoadErrorcallback); } else alert("Network unavailable");  }, initLoadSuccesscallback : function (response){ if(response !==null && response.opstatus === 0){ if(response.subCatList.length > 0){ var sampleData = {lblData: "name", imgData: "image"}; this.view.skeletonScreen.setData(response.subCatList,response.remainingResults,sampleData); } } }, initLoadErrorcallback: function(){ voltmx.model.ApplicationContext.dismissLoadingScreen(); alert("Failed to fetch the products...try again"); }, segReachEnd : function(productId){ //basically postshow //in callback, invokes adddata instead of setDta if (voltmx.net.isNetworkAvailable(constants.NETWORK\_TYPE\_ANY)){  var integrationSvc = KNYMobileFabric.getIntegrationService(this.serviceName); var headers = {}; this.currentPage=this.currentPage+1; data = {"categoryid" : productId, "pageNo": this.currentPage, "pageSize": 10} ;  integrationSvc.invokeOperation(this.productsOperationName, headers, data,this.paginatedLoadSuccesscallback, this.paginatedLoadErrorcallback); } else alert("Network unavailable");  }, paginatedLoadSuccesscallback : function (response) { if(response !==null && response.opstatus === 0){ if(response.subCatList.length > 0){ this.view.skeletonScreen.addData(response.subCatList,response.remainingResults); } } }, paginatedLoadErrorcallback : function(){ voltmx.model.ApplicationContext.dismissLoadingScreen(); alert("Failed to fetch the products...try again"); }}); |

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

1. **Save** the file.

## **B. 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/Iris/iris_user_guide/Content/C_ComponentsOverview.html) section of the VoltMX 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

 **1. General Properties**

**1 . widgetDataMap**

**Description:** Specifies the label Text and images that you want to display on the skeleton screen.

**Type :**   Array of JSON

**Read/Write :**  Read + Write

**Example:** var sampleData = {lblData: "name",

imgData: "image"}; this.view.skeletonScreen.setData(response.subCatList,response.remainingResults,sampleData);

##  **C. Events**

#### **1. loadMore**

**Description***:* To use Pagination you have to use custom event loadMore have to bind it to function which invokes the service to get future page data.

**Syntax**: loadMore

**Example**: You have to create the function to attach the loadMore event . In that function you have to invoke the service and in that success callback of the function you have to invoke addData with the list and remainingResults respectively.

|  |
| --- |
| this.view.skeletonScreen.loadMore = this.segReachEnd.bind(this, productId);/\* creating the function to attach loadMore Event \*/ segReachEnd : function(productId){ //basically postshow/\* invokes the service\*/ if (voltmx.net.isNetworkAvailable(constants.NETWORK\_TYPE\_ANY)){  var integrationSvc = KNYMobileFabric.getIntegrationService(this.serviceName); var headers = {}; this.currentPage=this.currentPage+1; data = {"categoryid" : productId, "pageNo": this.currentPage, "pageSize": 10} ;  integrationSvc.invokeOperation(this.productsOperationName, headers, data,this.paginatedLoadSuccesscallback, this.paginatedLoadErrorcallback); } else alert("Network unavailable");  }, /\*in callback, function invokes adddata along with key value and remainingResults \*/ paginatedLoadSuccesscallback : function (response) { if(response !==null && response.opstatus === 0){ if(response.subCatList.length > 0){ this.view.skeletonScreen.addData(response.subCatList,response.remainingResults); } } }, |

## **D. APIs**

## **1. setData**

## **Description:** To get the initial data from service or local data you have to call setData with the list of responses with the particular key value and remainingResults**.**

## **Syntax:** setData(parameters);

## **Parameters**: data

##  recordsRemaining

##  dataMap

**Return Value:** None

****

## **Example:** this.view.skeletonScreen.setData(response.subCatList,response.remainingResults, sampleData);

## **2. addData**

## **Description:** To get the future page data from service or local data you have to call addData with the list of responses with the particular key value and remainingResults

## **Syntax:** addData(parameters);

## **Parameters**: data

##  recordsRemaining

**Return Value:** None

## **Example:** this.view.skeletonScreen.addData(response.subCatList,response.remainingResults);

**4. REVISION HISTORY**

App version 1.0.2

##

#

##