Context-Sensitive Help For XS Advanced Applications

Proposed Approach:

For the integration of context/page sensitive help with the individual pages/functionalities in an XS Advanced Application. The documentation feature will be closely coupled to the application and will contain the web-based documentation. The code for enabling context sensitive help will reside in the application.

The way to represent the context sensitive help in the user interface, is completely up to the discretion and UI guidelines followed by the application itself. The help feature will not impose any particular user experience design or development paradigm on the application developers, except that the application should use the SAP UI5 library. The way that the help content is visualized, is also left to the discretion of the application development team.

Technical Details and Steps:

The help documentation URLs are static URLs, reachable over the web. Keeping this in mind, if the URLs do change in the future, the changes can be incorporated easily by the knowledge management teams responsible. The same changed documentation should be shared with the development team to incorporate that in the application.

We are following a particular path for documentation files in the application. Other applications can also follow the same.(*sap/hana/xs2/<app>/docs/file)*

sap/hana/xs2/admin/docs/file/<static.html>

sap/hana/xs2/admin/docs/file/css

sap/hana/xs2/admin/docs/file/html

sap/hana/xs2/admin/docs/file/images

sap/hana/xs2/admin/docs/file/js

sap/hana/xs2/admin/docs/helpTexts.hdbtextbundle

What do Knowledge Management teams have to do to make the feature work?

- Create the configuration files- For making the help feature work, one configuration file HDBTextBundle file, needs to be created and maintained. This configuration file is required to enable globalization of content and to designate the pages for which the context-based documentation is valid in the application. The documentation team should communicate the paths for the documentation resources and the paths of Text Bundle files to the application development team (or the files could be directly created by the KM team and sent to the application development team).
- The hdbtextbundle file should have keys and corresponding base values for them. The values are the link URLs and link texts. Using the repository translation tool, the translated texts have to be uploaded to the HANA database.

NOTE: To download example of the hdbtextbundle file, click ***> Attachments***.

Implementation Overview:

To enable context-sensitive documentation support for an XS application, the feature does not impose any strict UI restriction on the application development team, except that the application should be built using the SAPUI5 library. Having said so, the feature does benefit from UI5 best practices. Some recommendations and guidelines on how to embed the feature are as follows:

1. We have a sapui5 control " **sap.ui.unified.ShellHeadItem** ": help icon (**?**) in our SAP UI5 application. User will click on this to see the list of help contents.

2. On click of ShellHeadItem the below controls will be created ; which will display the help content texts and on clicking them respective link will be opened in a separate browser.

   - **sap.ui.unified.Menu** and **sap.ui.unified.MenuItem**

Sample code:

1. View should contain the ShellHeadItem representing the HelpContent in UI.

MainView

```
<u:Shell id="userShell" class="sapUiSizeCompact">
  <u:headItems>
   <u:ShellHeadItem id="helpContentShellId" press="handleHelpContentItemPressed"
    icon="sap-icon://sys-help" />
  </u:headItems>
 </u:Shell>
```

2. Fragment having the Menu

HelpContent.fragment

```
<core:FragmentDefinition
 xmlns="sap.m"
 xmlns:core="sap.ui.core"
 xmlns:u="sap.ui.unified">
  <u:Menu id = "helpContentMenu">
  </u:Menu>

</core:FragmentDefinition>
```

3. Controller having the event handling code. Individual applications can extend the MainController and call helpContentMenuPressed(event, helpContents); parameter 'helpContents' is an array of Help Contents with text and link pairs.

MainController

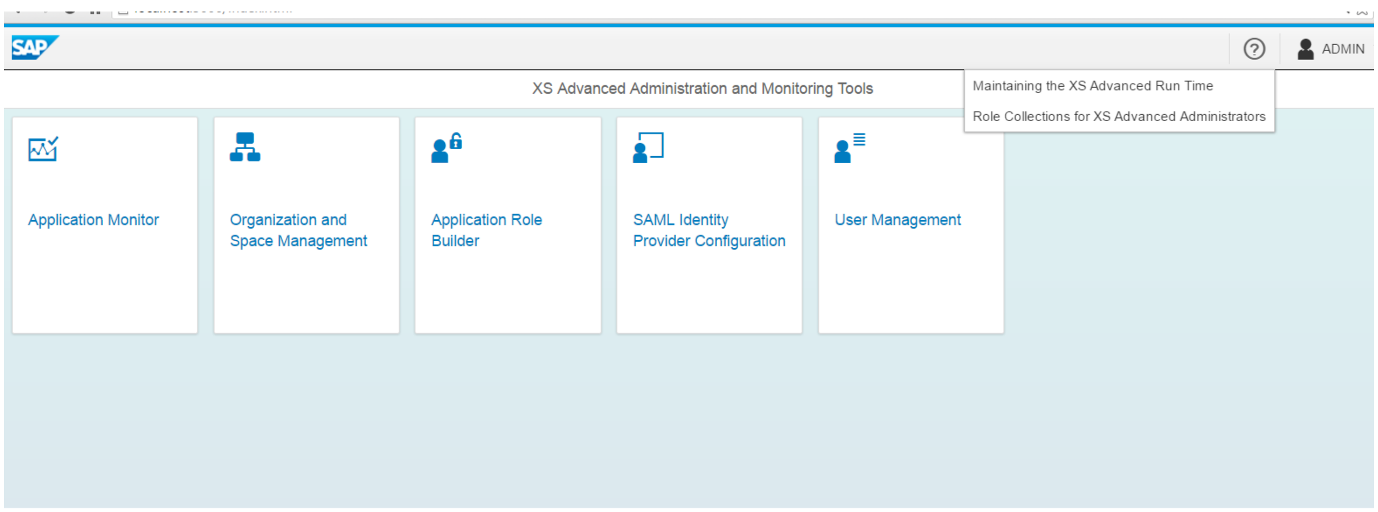
```
helpContentMenuItemPressed: function(oEvent, menuItemsObject) {
     var selectedMenuItem = oEvent.getSource();
     var link = this.getSelectedMenuItemLink(menuItemsObject,
selectedMenuItem.getText());
     var win = window.open(link, '_blank');
     win.focus();
 },

    createDynamicMenuItems: function(menuItemsObject, menu) {
     var that = this;
     for (var i = 0 ; i < menuItemsObject.length; i = i + 1) {
       var menuItem = new sap.ui.unified.MenuItem({
                 "text": menuItemsObject[i].text
             });
      menuItem.attachSelect(function navigateEntireToUrl(oEvent) {
        that.helpContentMenuItemPressed(oEvent, menuItemsObject);
      });
   menu.addItem(menuItem);

  }
    },

 getSelectedMenuItemLink: function(menuItemsObject, menuText) {

     for (var i = 0 ; i < menuItemsObject.length; i = i + 1) {
      if (menuItemsObject[i].text === menuText) {
       return menuItemsObject[i].link;
      }
     }
    },

    helpContentMenuPressed: function(oEvent, menuItemsObject) {

        var oButton = oEvent.getSource();
   // create menu only once
   if (!this._menu) {
    this._menu = sap.ui.xmlfragment(
     "helpContent",
     this
    );
    this.getView().addDependent(this._menu);
    this.createDynamicMenuItems(menuItemsObject, this._menu);
   }
   var eDock = sap.ui.core.Popup.Dock;
   this._menu.open(this._bKeyboard, oButton, eDock.BeginTop, eDock.BeginBottom,
oButton);
    },
```

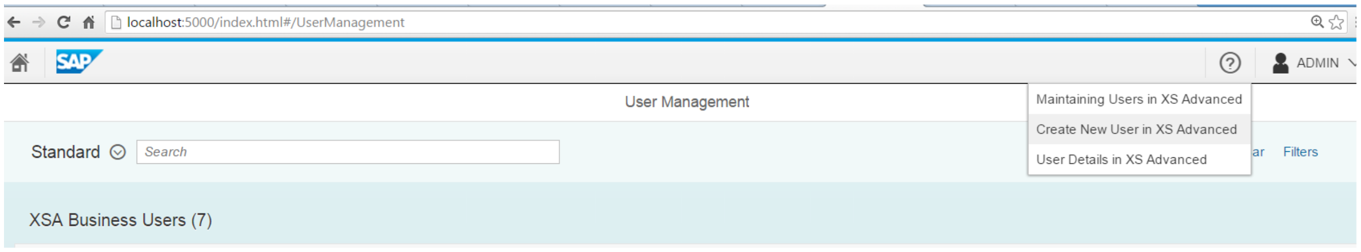
4. AppController which can have the respective helpcontent texts and links. To instantiate the helpContent we can send these text and keys to the MainController .

AppController

```
var AppController = MainController.extend("sap.hana.xs2.app.controller.AppView",
{

 //initialize
     onInit: function() {
            // require the jQuery.sap.resources module
            jQuery.sap.require("jquery.sap.resources");
            // load the resource bundle
            var helpContentBundle = jQuery.sap.resources({
              // specify url of the .hdbtextbundle
              url : "sap/hana/xs2/app/docs/helpTexts.hdbtextbundle"
            });
            this.getView().setModel(helpContentBundle, "helpContentBundle");
        },


  // Create the help content by adding objects with text and link for the menu
item and navigation;
        // Call MainController "helpContentMenuPressed" to manage the creation of
the items depending on the application.
        handleHelpContentItemPressed: function(oEvent) {

       var helpContentBundle = this.getView().getModel("helpContentBundle");
       var helpContentObj = {
         text : helpContentBundle.getText("USERS_MAINTAIN_TEXT"),
         link : helpContentBundle.getText("USERS_MAINTAIN_LINK")
       };
       var helpContentObj1 = {
         text : helpContentBundle.getText("USERS_CONFIGURE_TEXT"),
         link : helpContentBundle.getText("USERS_CONFIGURE_LINK")
       };
       var helpContentObj2 = {
         text : helpContentBundle.getText("USERS_DETAILS_TEXT"),
         link : helpContentBundle.getText("USERS_DETAILS_LINK")
       };
       var helpContents = [helpContentObj, helpContentObj1, helpContentObj2];
       this.helpContentMenuPressed(oEvent, helpContents);
         },
```

Sample Application preview:


ADMIN
XS Advanced Administration and Monitoring Tools
Maintaining the XS Advanced Run Time
Role Collections for XS Advanced Administrators
Application Monitor
Organization and Space Management
Application Role Builder
SAML Identity Provider Configuration
User Management


localhost:5000/index.html#/UserManagement
ADMIN
User Management
Maintaining Users in XS Advanced
Create New User in XS Advanced
User Details in XS Advanced
Standard
Search
Filters
XSA Business Users (7)