Best Practices for using existing VF in Lightning UI

Benjamin Lau, Enterprise Architect

Oct 26,2021

need to add standard disclaimer…….

Problem Statement

Workday will be migrating from Salesforce Classic to Lightning UI for Sales Users.

With the large number of users and profiles, there is a high possibility that Sales Users will be migrated in multiple waves by Profiles.

The Deal Close / Aptus team will be required to support the APTUS CPQ functionalities with Workday customization in both Salesforce Classic and Lightning UI for users with different profiles at the same time.

Solution

Salesforce Visualforce Pages are designed to work in Lightning Experience with minor or no revisions.

These slides will focus the technical aspect of embedding VF Pages in Lightning.

Discussion Topics

- 7 ways to display VF pages in Lightning
- Dynamic Navigation
- API to share VF Pages between Classic and Lightning Experience
  - VF Markup
  - JavaScript
  - Apex
  - SOQL <- not recommended
- Lightning Navigation API
- VF Customizations that will not work in Lightning Experience
- Summary

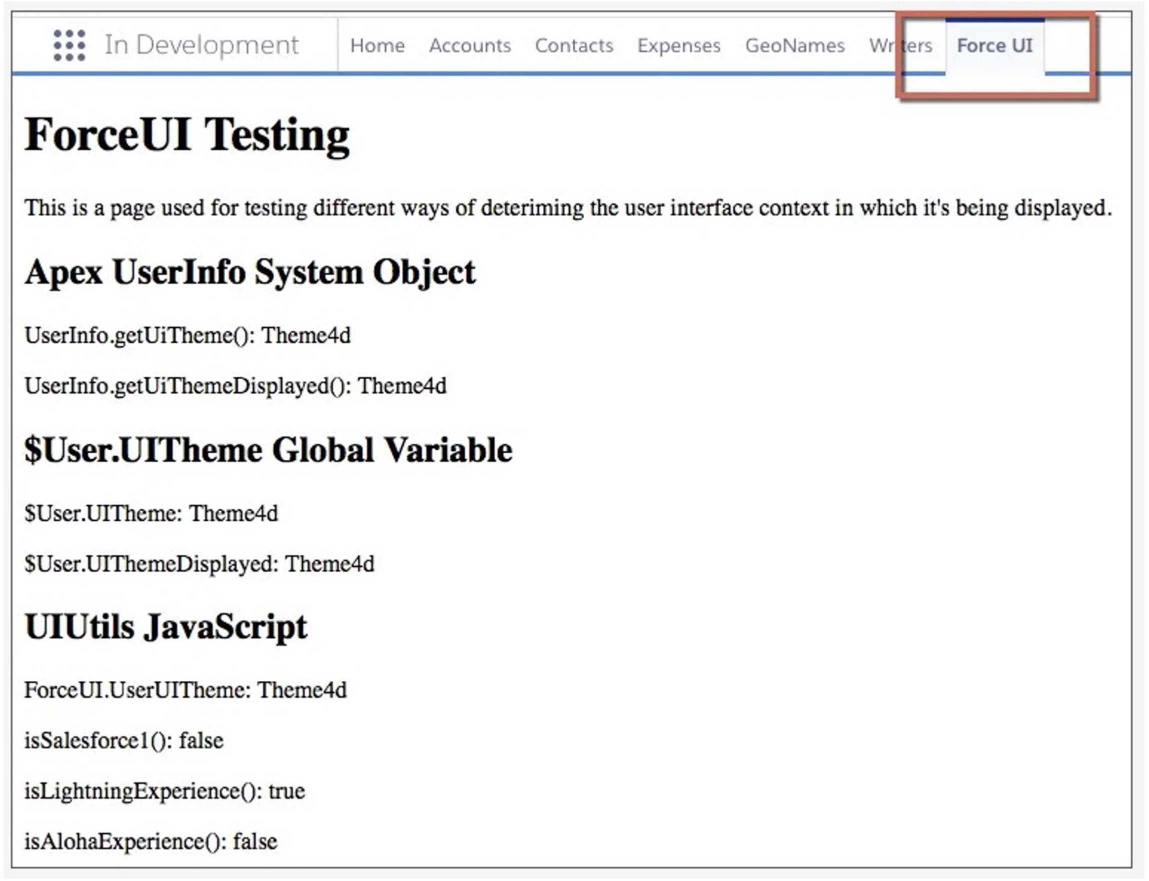
7 ways to display VF pages in Lightning

PoC Suggestions for WDAY APTTUS VF pages

- **Add VF Page as Tab to a Lightning App**
- **Add VF Page as a component in the Lightning App Builder**

Other mechanisms to display/Launch APTUS VF pages

- Open VF Page from the App Launcher
- Display Visualforce Page within a Lightning Standard Page Layout
- Launch a VF Page as a Quick Action (same in Classic & Lightning)
- Display a VF Page by Overriding Standard Buttons or Links
- Display a VF Page using Custom Buttons or Links.



ForceUI Testing

This is a page used for testing different ways of deteriming the user interface context in which it's being displayed.

Apex UserInfo System Object

UserInfo.getUiTheme(): Theme4d

UserInfo.getUiThemeDisplayed(): Theme4d

$User.UITheme Global Variable

$User.UITheme: Theme4d

$User.UIThemeDisplayed: Theme4d

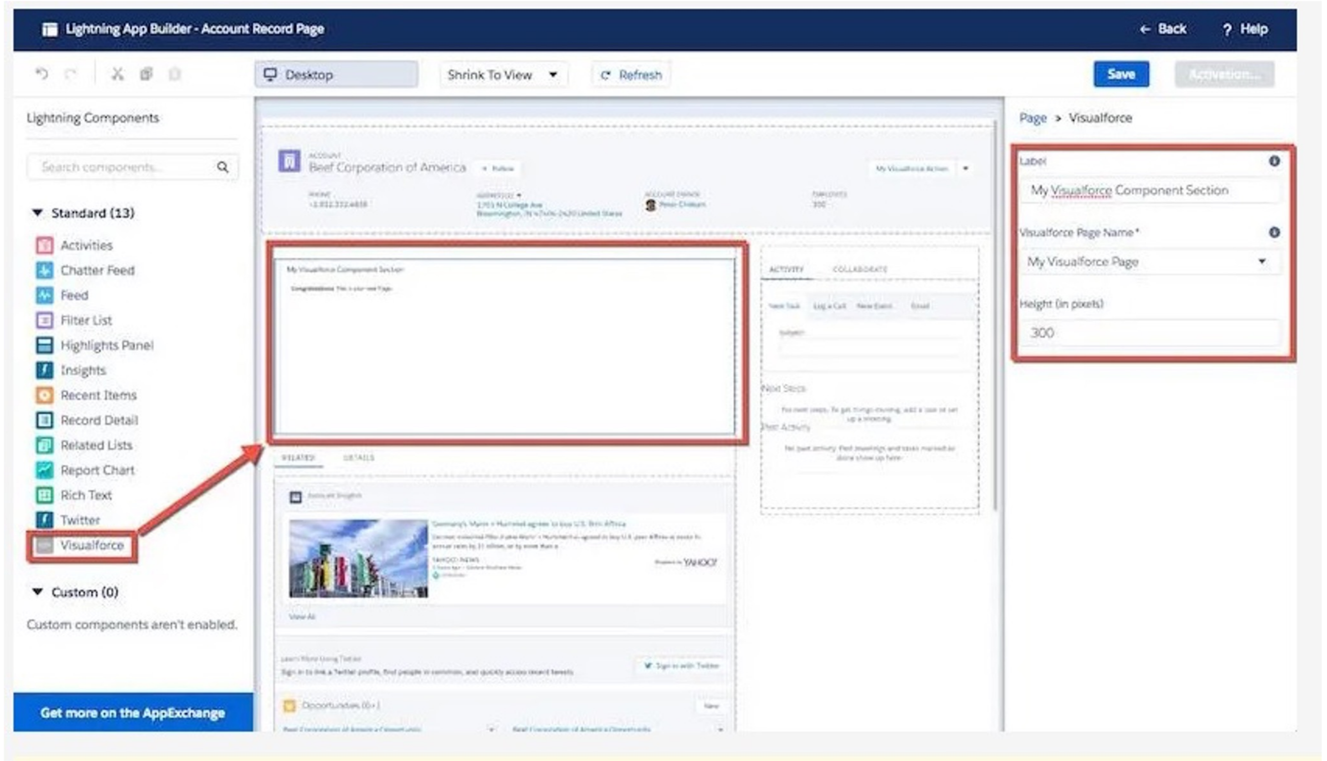
UIUtils JavaScript

ForceUI.UserUITheme: Theme4d

isSalesforce1(): false

isLightningExperience(): true

isAlohaExperience(): false



Label: My Visualforce Component Section

Visualforce Page Name*: My Visualforce Page

Height (in pixels): 300

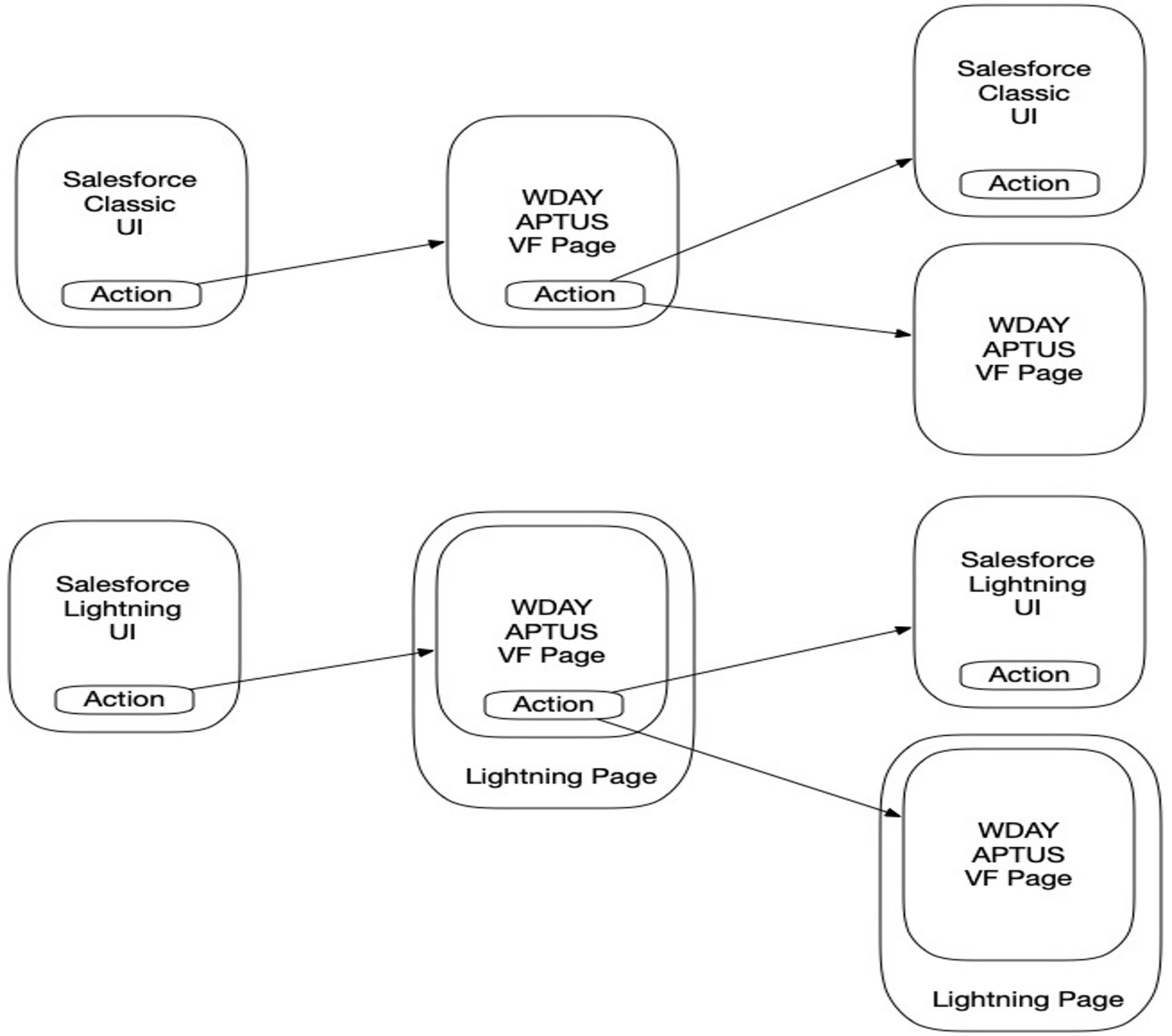
Dynamic Navigation

Goal

Use the same APTUS VF Page with customization to support both Classic & Lightning UI

APIs required
Detect context
Lightning Navigation

Minor modifications to the WDAY Aptus VF pages will be required to detect and use Lightning Navigation



API to share VF Pages between Classic and Lightning Experience

UITheme vs UIThemeDisplayed - global variables

- Theme1—Obsolete Salesforce theme
- Theme2—Salesforce Classic 2005 user interface theme
- **Theme3—Salesforce Classic 2010 user interface theme**
- **Theme4d—Modern “Lightning Experience” Salesforce theme**
- Theme4t—Salesforce mobile app theme
- Theme4u—Lightning Console theme
- PortalDefault—Salesforce Customer Portal theme
- Webstore—Salesforce AppExchange theme

Look for Theme3, Theme4d to display UI specific elements or add stylesheet

API to share VF Pages between Classic and Lightning Experience

VF Markup

```
<apex:outputPanel rendered="{! $User.UIThemeDisplayed == 'Theme3' }">
    <apex:outputText value="This is Salesforce Classic."/>
    <apex:outputText value="These are multiple components wrapped by an outputPanel."/>
</apex:outputPanel>

<apex:outputPanel rendered="{! $User.UIThemeDisplayed == 'Theme4d' }">
    <apex:outputText value="Everything is simpler in Lightning Experience."/>
</apex:outputPanel>
```

API to share VF Pages between Classic and Lightning Experience

Javascript

Use the UITheme .getUITheme global variable in Javascript to test

```
function isLightningDesktop() {
  return UITheme.getUITheme === "Theme4d";
}
```

API to share VF Pages between Classic and Lightning Experience

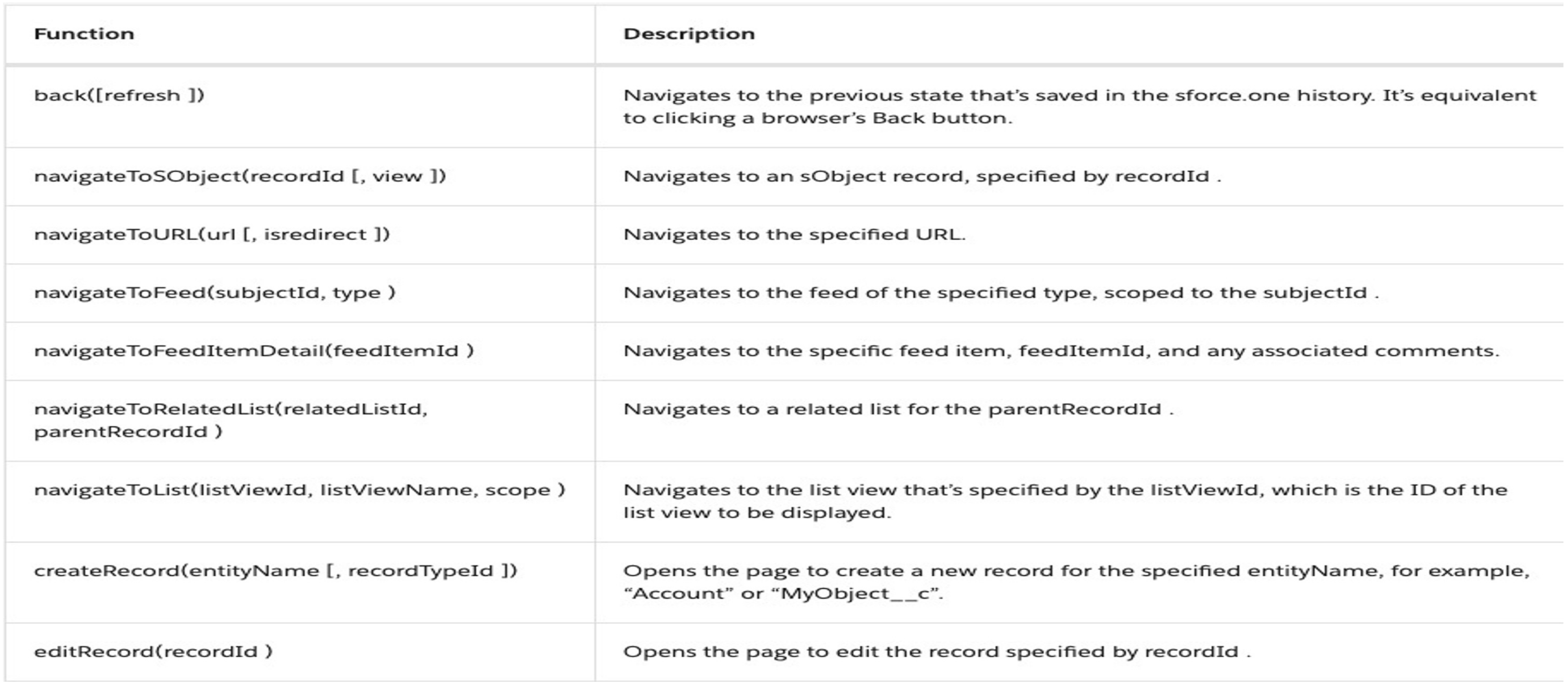
Apex or SOQL - Server side code…… avoid if possible.

```
public with sharing class ForceUIExtension {
    // Empty constructor, required for Visualforce controller extension
    public ForceUIExtension(ApexPages.StandardController controller) { }
    // Simple accessors for the System.UserInfo theme methods
    public String getContextUserUiTheme() {
        return UserInfo.getUiTheme();
    }
    public String getContextUserUiThemeDisplayed() {
        return UserInfo.getUiThemeDisplayed();
    }
}
```

Lightning Navigation API

Suggestion - use the Javascript - sforce.one object

see - https://developer.salesforce.com/docs/atlas.en-us.pages.meta/pages/salesforce1_dev_jsapi_sforce_one.htm



Function	Description
back([refresh])	Navigates to the previous state that's saved in the sforce.one history. It's equivalent to clicking a browser's Back button.
navigateToSObject(recordId [, view])	Navigates to an sObject record, specified by recordId .
navigateToURL(url [, isredirect])	Navigates to the specified URL.
navigateToFeed(subjectId, type)	Navigates to the feed of the specified type, scoped to the subjectId .
navigateToFeedItemDetail(feedItemId)	Navigates to the specific feed item, feedItemId, and any associated comments.
navigateToRelatedList(relatedListId, parentRecordId)	Navigates to a related list for the parentRecordId .
navigateToList(listViewId, listViewName, scope)	Navigates to the list view that's specified by the listViewId, which is the ID of the list view to be displayed.
createRecord(entityName [, recordTypeId])	Opens the page to create a new record for the specified entityName, for example, "Account" or "MyObject__c".
editRecord(recordId)	Opens the page to edit the record specified by recordId .

VF Customizations that will not work in Lightning Experience

- <apex:relatedList> - blockedList
- <apex:iframe> avoid if possible - hard to debug
- do not use JavaScript to set window.location directly
- 6 standard actions overrides for Standard / Custom Objects…..
- Static URL to salesforce resources (`like link = '/' + accountId + '/e'`)

Summary / Recommended Steps

- Complete Trailhead - **Visualforce & Lightning Experience Module**
  https://trailhead.salesforce.com/content/learn/modules/lex_dev_visualforce

- Review Aptus VF Customization for **known features that do not work in Lightning**
- **UI/UX Design** based on replicating business capabilities
- Select mechanism to display Aptus VF Customization in both Classic & Lightning
  - (suggest Lightning App Tab or VF component in Lightning Page)
- Map out navigation mechanism in both Classic & Lightning UI
- Select mechanism to enhance Navigation
  - (suggest VF Markup / Javascript)
- Test in both Classic & Lightning UI on desktop & mobile with all Sales Profile