Framework Code Improvements - Template, and CSS refactoring, Field-Component Improvements, Ajax Improvements

Purpose

There are still many areas of KRAD that need analysis and improvement. The recent bean renaming/refactoring was the first step in this work. We would like to do similar analysis and improvements to the jsp templates, a reworking and fixing of all css files based on new template structures and new css class names now available, general field/component improvements, and improvements to the ajax architecture to cover more cases and make it even easier to use. In addition, this would include file/property renaming and removal for additional clarity around these areas.

Detailed Description

Though Krad is fully usable in its current state we feel there are many areas that could use some extra enhancements and fixes so they do not become a problem in the immediate future or due to a larger developer user base become much more difficult and time consuming to fix in the future.

Full jsp/tag analysis and fixes:

We have some additional html elements that are not needed and make in more difficult to style portions of the page consistently and correctly. This would be an analysis of how to change the jsp/tags to make the elements they generate more concise and minimize the overall unnecessary code they generate. In addition to this some jsp elements being generated will follow a new structure entirely, one early identified item is the top level View generation. Another potential candidate is HeaderField (e.g. we need to be able to include some actionfields in headers). Other issues: some wrapping divs/spans around elements get the same css classes as the item they are wrapping.

Related JIRAs:

- https://jira.kuali.org/browse/KULRICE-6405

Full css rework:

Currently the css files and class names are unorganized, have a mixture of css class names due to name transitions that were never fully complete, contain additional classes that do not serve a purpose, css classes that belong in the base css file but not at the theme level (or defaults that belong at base and need to be built upon at theme level), and complexity added because lack of css class hooks on the elements of the page or because of the unneeded complexity of the elements of the page that were generated. Due to the addition of css class names recently to all relevant components, a new "uif" prefix standard (which also needs to be adopted in js and jsps that we have control of, as part of this work), and proposed improvements to the jsp/tag generation: all css files need to be analyzed. Proposal is that we start fresh, while still maintaining/improving the color/font scheme related to the 2 main themes, but completely reworking spacing, layout, css class cascading hierarchy etc.

Field improvements:

Many improvements are related to component/field hierarchy. We need to perform an analysis and architectural plan around some of the more problematic/confusing fields and possibly introduce new java classes to the overall component hierarchy to make certain fields such as, ActionField, FieldGroup, HeaderField make more sense from a usability standpoint. One specific case here is the removal of labels from fields that don't make sense that have them. Also related to this work is analysis of each property on fields and components to determine if they need to be removed and/or expanded upon for ease of use.

In addition this work, we may add new beans which represent common default settings for fields such as date, address, etc, so there is an easy field to grab when dealing with these types of fields.

Ajax Improvements:

While what we have in place for ajax retrieval covers many use cases - and is being widely used by all projects using krad - we would like to expand this functionality. Some improvements currently in discussion are:

1. Refresh in a sequence (workaround already used in ks)
2. Seperation of ActionField into 2 trees for html actions and ajax actions
3. Ajax Refresh Timer
4. Ajax retrieval from alternate controllers/views
5. Ajax retrieval onLoad

Usage Scenarios

Include at least one usage scenario, from the user's task perspective, that might be helpful in understanding the issue:

Mocks and Diagrams

View.jsp Structure:

- ApplicationDiv** AppHeader** View Div** Breadcrumbs** View header** View Content Div *
    -
        -
            - **Navigation Div**
            - **Page Div (future scrolling area)**
                - **Page Header**
                - **Page Content**
                - **Page Footer**
        - **View Footer Wrapper (padding calculated by js to accommodate left nav when present dynamically)**** View Footer**
    - **AppFooter**

Performance

If applicable, list expectations for performance (optimal and worst cases would be fine, give time in seconds):

References

Include links to other confluence pages or external resources that might be helpful for this issue:

Requirements Listing

List all requirements (individual verifiable statements) that indicate whether the work for this item has been complete. If there are requirements that are not essential to the functionality but would be nice to have if time allows, enter those under 'Non-Essential':

Primary:

1. JSP enhancements to View organization - rework the html structure - changes to js to work with updated structure/name
2. Changes to Groups and Layout Managers to reduce divs generated
3. New component in hierarchy called ContentElement - remove/convert jsps to use this when necessary - high risk
4. Fixes for template.jsp to remove complex wrapper logic needed by refresh/progressive disclosure - logic changes to the js and java to accommodate - high risk

Secondary:

1. Changes to use div where appropriate over span
2. Change navigation plugin to work with new view structure and change logic to use non-negative margin
3. Change radio, checkbox group, and field group to use field set - consider collection items do the same

Component Hierarchy/Type Changes

1. Create new package org.kuali.rice.krad.uif.element
2. Create new component interface ComponentElement (in element package) that extends Component
3. Create new base class ComponentElementBase that extends ComponentBase
4. Change Control and ControlBase to extend ComponentElement and ComponentElementBase
5. Create Header element with properties headerText and headerLevel
6. Create HeaderGroup (in group package) that extends Group and includes a nested Header element (with property setters/getter that set the header component)
7. Remove HeaderField
8. Create bean 'Uif-HeaderGroup' for the header group component
9. Create Iframe element component with properties source, height, and frameborder.
10. Remove IframeField
11. Create Image element with all properties from ImageField
12. Change ImageField to include a nested Image element. Create getter/setters that set the corresponding properties on Image
13. Create bean 'Uif-Image' that points to Image element, 'Uif-ImageField' will point to ImageField as it does now
14. Create Label element component with all current properties of LabelField
15. Change FieldBase to include the Label component

16. Remove LabelField
17. Create Link element that has current properties of LinkField
18. Change LinkField to include a nested Link component, add getter/setters that set the corresponding properties on Link
19. Change 'Uif-Link' bean to point to Link element, add new bean 'Uif-LinkField' that points to LinkField
20. Create Space element that self renders to generate ' '
21. Create SpaceField component that includes a nested Space element
22. Remove BlankField
23. Create 'Uif-Space' and 'Uif-SpaceField' beans, remove 'Uif-EmptyField'
24. Create new templates for the new element components that have content from the field
25. Create a generic field template that renders the label, adds the wrapper span, and invokes template tag for the nested element
26. Rename 'componentOptions' in Component to 'templateOptions'
27. Remove messageType from MessageField (doesn't have any use)
28. Rename 'label' and 'shortLabel' on Field to 'labelText' and 'shortLabelText'. (This will be consistent with the Label element and allow the nested label component on field to be named 'label')

Other Component Refactorings

1. Consolidate Inquiry and DirectInquiry widgets into one by adding a 'directInquiry' flag to the Inquiry widget. That way, we can have only one property in the input field and you don't have to configure the relationship twice to get both the standard inquiry and direct inquiry
2. Look at relationships with lighbox. For example, why does Action field have two nested Lightbox components? It seems like the Inquiry and Lookup widgets should operate on the lightbox for ActionField instead of ActionField having this dependency. In particular since it is not possible to use the Lightbox generically.
3. Also case of LinkField and the nested Lightbox widget. I think this is there for inquiry. Would be good to have a general lightbox component with an 'openInLightbox' flag or a 'lightbox' target option that states the linked contents should be displayed in a lightbox.

Ajax Improvements

1. Add property ajaxSubmit to ActionField. When set to true this will submit the form using Ajax instead of the browser submit. Will default to updating the page contents.
2. Add property preSubmitCall to ActionField. This will invoke script before the form is submitted. A boolean will be returned to indicate whether the form submission should continue.
3. Add property successCallback to AjaxActionField. This will be invoked for succesful ajax calls.
4. Add property errorCallback to AjaxActionField. This will be invoked for failed ajax calls.
5. Rename 'clientSideJs' to 'actionScript'. This will be used for script only actions (do not invoke the server).
6. In krad.ajax, do cleanup to have an ajaxSubmitForm and stantard submitForm (browser). Methods to do validation before hand.
7. Refactoring action parameters to not do script writing of hiddens, but to use new 'data' attributes. For ajax calls the data attributes will be picked up and added to the ajax data. For standard browser submits the data attributes will be picked up and added as hiddens

Improvements to ActionField for Lightbox enhancements

1. Add property 'displayResponseInLightbox' to display the response contents in a lightbox instead of the current window.
2. Add property 'showLightboxGroup' to display a group that have been configured to display in a lightbox. This would not trigger a server side call.

Improvements for Component Refresh

1. Add 'refreshTimer' property to component base. The value will be a time in seconds that the component will automatically be refreshed.
2. Add support for refreshing multiple components at a time. Add 'alsoRefreshComponents' property that can be configured with a component that has a refresh condition.

Smarter Ajax Returns

To support things like the dialog box return, or navigating to other views we need to make our Ajax returns 'smarter'. This essentially means when sending back an ajax response, we need to write data that is used by the ajax call to determine how the response should be handled. The handling options needs are:

- Redirect - Ajax call will perform a browser redirect to a new URL
- Open in Lightbox - The content being returned should be opened in a lightbox
- Replace View - Indicates the entire view contents should be replaced

We might consider having multiple actions that can occur each with a wrapper handler. For example, 'update title' with new title contents, 'update page' with page contents', and even 'show lightbox' with lightbox contents.

Dependencies

List any functional or technical work that must be completed before work on this item can begin:

1. item

Issues

List any issues that need to be resolved before work on this item can begin:

Functional:

1. item
2. item

Technical:

1. item
2. item

QA or Regression Testing Plan

List steps needed to test the basic functionality of this update, enhancement, bug fix

1. Make sure all current and new functionality still works

Checkoff

Functional Analysis Complete? **No**

Needs Review by KAI? **No**

Technical Analysis Complete? **No**

Needs Review by KTI? **No**

Estimate:

Field/Component Refactorings - 70 hours

Other Component Refactorings - 30 hours

Ajax Improvements (Not including lightbox) - 50 hours

Technical Design: 2.2 Framework Improvements

Jira: KULRICE-6671

Final Documentation: Link Here