Maintenance Document Initiation Link 2

A business object's lookup page automatically determines if a maintenance document exists for the given business object and displays a "create new" button accordingly. Still, for fun, let's take a look at how to create lookup links in the portal and how to create other links on the lookup page itself.

Lookup links

Let's take a look at the link to the lookup for a business object, which just happens to have an associated maintenance framework. Here's what, for instance, the account lookup looks like:

```
<portal:portalLink displayTitle="true" title="Account"
url="kr/lookup.do?methodToCall=start&businessObjectClassName=org.kuali.module.chart.bo
.Account&docFormKey=88888888&returnLocation=${ConfigProperties.application.url}/portal
.do&hideReturnLink=true"
/>
```

Whoa! Let's break this up a bit. First, we're using the portal:portalLink tag. The text that is clickable will be "Account", and it the link will show up in the portal (because displayTitle="true"). Then we've got the url...

```
kr/lookup.do?methodToCall=start
```

This is the start, and all it says is that we want to defer to the embedded Kuali Rice part of the application, which has the lookup page - lookup.do. Then we start our parameters, which begins, as always, with a methodToCall - the method in the action class to call when the page is rendered, which for lookups is always "start".

```
&businessObjectClassName=org.kuali.module.chart.bo.Account
```

This makes much sense: the lookup needs a parameter to tell it what the class name of the business object it deals with is. In this case, it's org.kuali.module.chart.bo.Account.

```
&docFormKey=88888888
```

This key is basically a dummy, used to initialize the "docFormKey." The "docFormKey" parameter is used by the maintenance document framework, and basically holds the key of a maintenance document in the current user's session. So, for instance, when the user is editing a maintenance document form and performs a lookup on a field and then returns to the maintenance document form, that form has been held in the session for quick retrieval, and docFormKey holds the key to that form in the session. And it needs to be initialized; hence passing in this parameter.

```
&returnLocation=${ConfigProperties.application.url}/portal.do
```

We also need to return the url for the return location that the lookup would use. Here, we just return to the main portal page, using the preset variable "ConfigProperties.application.url"

```
&hideReturnLink=true
```

This is actually a parameter to the returnLocation url! It just says that the lookup screen itself shouldn't show a return link...because at initialization of the lookup, there's not really anything (besides the portal) to return to.

Once we're on a lookup page, the Kuali Rice framework will figure out if there's a maintenance document associated with the given lookup, based on the class that is being looked up on. If there is, it automatically shows the "create new" button, as well as shows us the "edit" and "copy" links in search results once we've gotten values from a search.

What about links to globals?

Now, on some lookup pages, including the lookup page for Account, we have *two* initiation links: a "create new," which we know now is automatically created for us, and a "create new global." Sadly, the maintenance document framework isn't as smart about which business objects have global variables, and so we've got to do that a bit more by hand.

Let's take a look at the data dictionary configuration, for the business object Account. Here's what the beginning of the lookup section looks like:

```
1. <lookup>
2.   <lookupableID>accountLookupable</lookupableID>
3.   <title>Account Lookup</title>
4.   <menubar>
5.     <![CDATA[<a
href="maintenance.do?methodToCall=start&businessObjectClassName=org.kuali.module.chart
.bo.AccountGlobal"><img
src="${kr.externalizable.images.url}tinybutton-createnewglobal.gif" alt="create new
global" width="98" height="15"/></a>]]>
6.   </menubar>
```

We see that the lookup has a "menubar" element, and that this menubar, within CDATA escaping, defines an "a" tag. This is, of course, the link to the global. We can add as many of these as we want; Account Delegate, for instance, has 3 buttons, 2 of them defined like this, in its menubar, in KFS 2.0.

Here the page that is linked to is "maintenance.do" and the business object class name is to AccountGlobal instead of Account. Finally, we have an image tag for the button...creating links like this is HTML-land, and that's unavoidable. However, it isn't so hard to add new links like these to the lookup page for any business object.

Unable to render {include} The included page could not be found.