Why an improved Callback action for dialogs/forms is needed?

Related JIRA issue: ~~MGNLUI-3576~~ - Provide validatable callback action for both dialogs and forms CLOSED

JIRA issue that could use the improved action: ~~MGNLRES-186~~ - Add create and delete actions for resources CLOSED

The problem

The pattern of a new resource creation action should look like this:

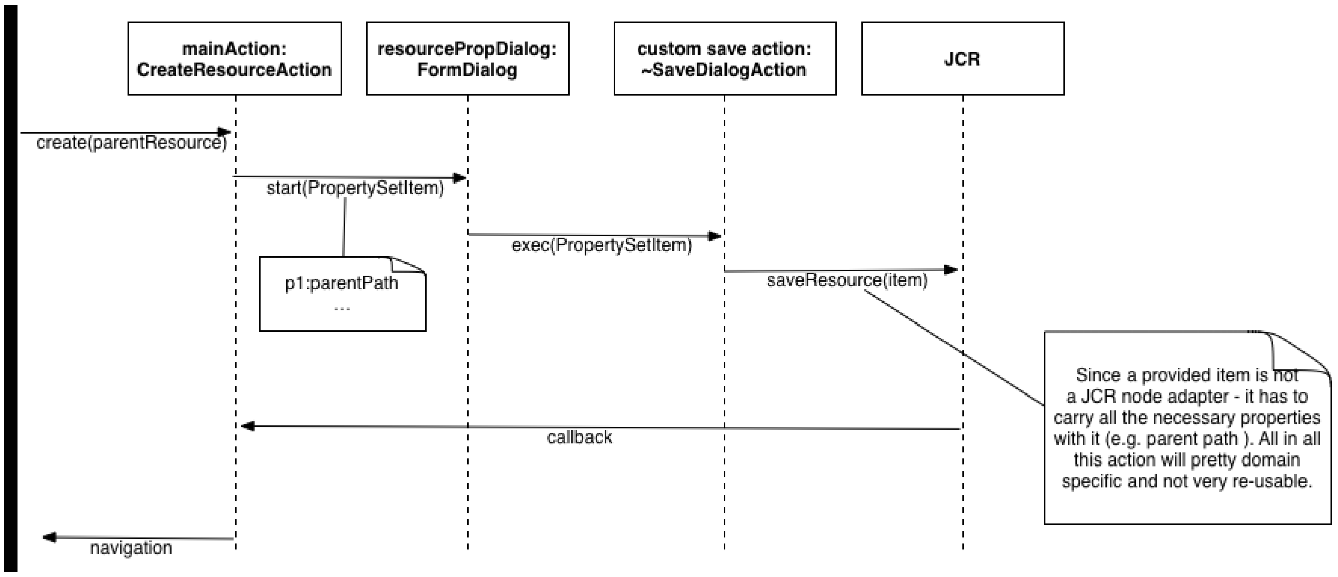
1. *Add resource* actionbar action brings in a dialog.
2. User specifies the name of a new resource.
3. User saves the dialog.
4. The new resource is created and the editor sub-app is automatically opened.

Possible quirks:

Main hurdle is that most of the 'hotfix'-type have to deal with two aspects of data - **ResourceOrigin** (on the higher level) and **JCR** (on the lower level).

1. The new Resource Files app operates over simple `PropertySetItem`'s, whereas the actions usually want to do some JCR work and hence expect `JcrNodeAdapters` as input (and also they output JCR-related information).
2. Resource Vaadin Item does not provide a link to the parent folder out of the box.
3. Resource Files app will not react on the `ContentChangedEvent` coming from JCR-related actions since the type of id's do not match (`JcrItemI` when mere String paths expected).
4. JCR actions more or less easily operate with 'new'/transient items (`JcrNewNodeAdapter`). However, `JcrResourceOrigin` knows nothing of such items and will throw a `ResourceNotFoundException` upon any attempt to query such a resource. This makes it impossible to 'pre-create' a resource in JCR and work with it by means of `JcrResourceOrigin`.

If we want to create a new resource in a 'usual' way trying to rely on just the existing actions:


mainAction: CreateResourceAction
resourcePropDialog: FormDialog
custom save action: ~SaveDialogAction
JCR
create(parentResource)
start(PropertySetItem)
p1:parentPath ...
exec(PropertySetItem)
saveResource(item)
Since a provided item is not a JCR node adapter - it has to carry all the necessary properties with it (e.g. parent path). All in all this action will pretty domain specific and not very re-usable.
callback
navigation

- A custom `CreateResourceAction` unwraps the parent resource/folder and starts a dialog.
- Dialog populates new resource properties into a PropertySetItem.
- Custom SaveResourceAction creates a resource entry in jcr and fires the callback
- The callback triggers the navigation to the new resource editor from the initial action (`CreateResourceAction`).

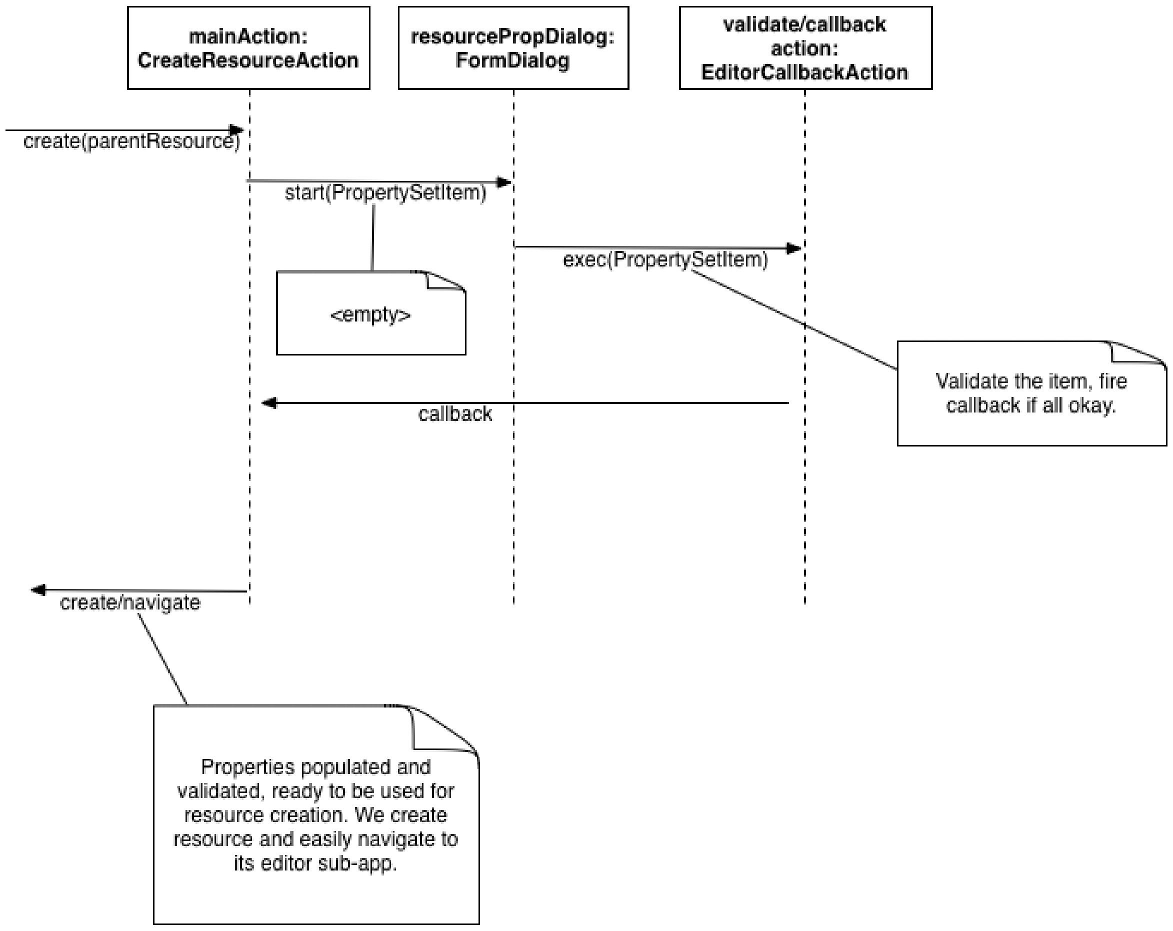
Problems I see here:

1. We have to add the parent path to the Vaadin item which we pass into dialog, so that when it saves - it knows where to create the new JCR node.
2. When the saving is done - the initial action will have hard times resolving where to navigate because the new resource's path is not communicated back.
3. We have to develop two custom, not really re-usable actions.

Possible alterations of this scenario:

1. We resolve the parent JCR node in `CreateResourceAction` and start dialog with an ad-hoc `JcrContentConnector`:
    a. We could then even delegate to `OpenCreateDialogAction, but:`
        i. it would create a new resource following the logic different from that defined in `ResourcesContentConnector#createNewResource`
        ii. It would fire the `ContentChangedEvent` with the item id type not treatable by Resource Files app.
        iii. it would still be problematic to navigate to the detail sub-app (??)

Proposed solution


mainAction: CreateResourceAction
resourcePropDialog: FormDialog
validate/callback action: EditorCallbackAction
create(parentResource)
start(PropertySetItem)
<empty>
exec(PropertySetItem)
Validate the item, fire callback if all okay.
callback
create/navigate
Properties populated and validated, ready to be used for resource creation. We create resource and easily navigate to its editor sub-app.

1. As **MGNLUI-3576** suggests - we introduce a new action called smth like `EditorCallbackAction` in **ui-framework** which supersedes both the existing `CallbackDialogAction/CallbackFormAction.`
    a. The semantics of the new action is the same - just notify the editor, that the detail sub-app/dialog finished editing the item without doing anything extra.
    b. The main difference from the aforementioned actions - it should also do validation to ensure the editing is correct.
2. According to this all the resource-related logic happens within the `CreateResourceAction`, the dialog action simply enclosed in it and only populates the properties like *resource name,* nothing else.
    a. We end up writing only one custom action (which is sort of logically complete in terms that the full cycle of resource creation happens here).
    b. No need to pass around context information (e.g. parent coordinates, resulting resource path etc).
    c. Actual JCR entry creation is postponed till the latest action phase, cancel action stays as dumb as usually.
    d. Navigation is trivial since all the data is at the hand.