Concept - Workbench consistency

Story

⚠ Unable to locate Jira server for this macro. It may be due to Application Link configuration.

Abstract

The workbench can get out of touch with reality after an action is performed on the content that it displays.

Container

We change the containers to identify the items within it not using their paths but instead in a format consisting of the node's identifier and the name of the property if it represents a property. We don't add a dedicated object for this, instead we use a string representation, to keep the memory footprint down.

The string representation is <uuid> for nodes and <uuid>@<propertyName> for properties.

Changing this will solve that we cannot distinguish between a child node and a property that has the same name. It will also make it easier to keep the selection in sync, although it will not be enough to completely fix the sync issues.

The layer we'll actually start using uuids at is the views - hence presenters have to be adapted to only pass uuids to the views

Locations

uuid in Locations would just be to technical/ugly so we'll use the following pattern:

- <path> for Nodes and <pathOfHostingNode>@<propertyName> for Properties - right now we cannot tell whether a location is pointing to a node or a property (also we have a potential problem with nodes and properties with the same name)

Actions

The workbench passes on the selected item to the action, if nothing is selected the passed on item is the node used as root for workbench (configured in WorkbenchDefinition.path).

The impact of an action on the workspace can be anything from:

- A node was added
- A property was added
- A property was removed
- A node was removed
- A node was renamed
- A node was moved
- A node and all its sub nodes had the activation status changed.
- The order of sub nodes was changed.
- More than one node was removed
- More than one property was added to a node
- Every sub node on a node got a new property
- Nothing changed
- A command was executed
- And there are many more examples...

Can the action be expected to know what changes a command made?

Workbench

After an action changes the workbench needs to show the workspace as it now is *and* update what item is now selected. Also, the actionbar and the statusbar must be updated based on the new selection.

When a node or a property is deleted the workbench must update its selection to not still think that the now deleted node or property is selected.

When renaming a node or a property the workbench must update its selection to not still think that the old name is what's selected.

When the workbench receives a ContentChangedEvent it must not change selection to the changed node/property because this event is meant to make the workbench stay in sync when the workspace changes. It can come from anywhere, another app and possibly even another user.

When the selection changes so must also the location.

Events

We're using events to synchronize the workbench after an action is executed. We will need to review the events, when they're sent, if they carry enough detail, and which bus they're sent on.

- to be defined how the look like (source & target?) -> e.g. check how JR does it, reuse?
- should fire uuid - also ensure consistency
- only use id’s?
- add new asset
- Which bus should be informed by content changing actions etc...

Moving in Tree (via drag & drop and via actions)

<to be verified: unwanted activation of drag & drop mode on ipad>

We will probably need an action for move, for two reasons, the ipad and due to the TreeTable not scrolling when you drag an item to the top or the bottom of the table.

Out of scope for this concept

Location encoding

Multi select and execution of action that operate on multiple items

Proposed solutions

1 Actions return list of changed or deleted paths

The returned list would contain paths dirtied by the operation. The list would be paths that have been deleted or updated, where updated means that there were changes made to the node itself, the order of its child nodes changed or larger changes made to the subtree below the node.

2 Actions fire detailed events about the changes made

The action would fire many events or a single event containing many dirtied paths (see above).

3 Bridging JCR observation events into the container

The container itself would be responsible for being in sync using a JCR observation listener to monitor changes. When something is changed the container updates its internal state and repaints its UI.

If necessary the events could be buffered and processed only on request after an action has been executed to limit the amount of repaints. Or buffered for a period of time in order to collect a bunch of events and repaint only once for all of them.

4 Actions use JCR observation to monitor all the changes made and fires events

Basically a combo of 2 and 3

Evaluating the proposals

For proposal 1 and 2 we assume that the action has the information of exactly what was changed. If the action executes a command we can't expect the action to have full knowledge about all the changes made.

For proposals 1, 2 and 3, when the list or the event(s) are processed by BrowserPresenter or WorkbenchPresenter, what do they do with the information? They can update the selection yes. But can they update the container properly? Won't calls to the container to remove an item also go through to the JCR attempting a new delete operation?