Concept - DAM: Workspace structure for assets with originals

Requirements

The DAM will have a concept of originals, where the file you upload is kept as an original that you can revert an asset back to later on.

Remember that when an asset is duplicated, the new asset must still be able to revert to the original of the previously existing asset - even if the previously existing asset is deleted.

For full requirements please refer to UX spec Digital asset management

Alternative 1 - Separate workspaces

Assets in one workspace, originals in another.

Pros

➕ Simplicity, originals never show up when navigating/enumerating the tree of assets

Cons

➖ References will need to be by uuid stored as string only

➖ Streaming the original requires supporting an additional workspace

Alternative 2 - Separated by path

Assets under /assets and originals under /originals

Pros

➕ References can be JCR properties of reference type

Cons

➖ Navigating/enumerating asset must always start with the /assets node and never go to the root

➖ The hierarchy of assets start at level 1, creating a special case for the topmost folder. I.e. an asset shown in the root of DAM is not in the root of the workspace. This needs to be taken into account in code accessing the workspace.

Alternative 3 - Mixed in the same tree

Since an asset refers to exactly one master we could have a tree of folders and originals with the assets as sub nodes of the original.

Pros

➕ Findig the original given an asset is trivial

➕ Detecting when the original is not used by any assets is trivial

➕ Enumerating the assets using a particular original is trivial

Cons

➖ Not ideal for the workbench component as it will need to hide the original but show the assets beneath it

➖ Ordering assets will only be possible among the assets having the same original

➖ Enumerating/navigating the assets will require entering each original in a folder and collecting the assets from within them

➖ Uploading a new file into an asset will change its node path

Alternative 4 - Subnode on Asset

When an asset is edited, copy the resource subnode to a new subnode on the Asset.

Concept:

- /jcr:content-original node can save the original resource on the same asset node.
- On saving an edit, check if /jcr:content-original exists, if not: copy /jcr-content to it before saving edit.
- On uploading a new file, delete /jcr:content-original node
- On duplicating an asset - be sure /jcr:content-original is also duplicated (I guess everything is)(Of course this wastes space (duplication of original).)
- On "Revert to original" click, copy /jcr:content-original to /jcr:content and delete /jcr:content-original

Questions/Problems:

- What about the "asset family" - How could you find all variants of the original? Do you need to be able to? Probably yes -this could be implemented by storing an extra reference when things are duplicated.
- Repeated storage of an original when you create duplicates.

Implementation Notes:

- Resource node of asset is updated after upload and edit in FileItemWrapperImpl.populateJcrItemProperty() This could be the place to check if an original exists yet, and copy the resource there if not, before saving the changes.
- Upload saving happens in AbstactUploadFileField.uploadFinished() with fileItem.updateProperties() and fileItem.populateJcrItemProperty()

Node types

The original and the asset will use different node types.

mgnl:asset

Has a property linking to the original.

Depending on the provider has a content node containing the binary data.

Can have any property on it representing its meta data.

mgnl:original

Identical to asset except it has no link to an original and does not keep meta data.

Potentially has a content node containing the binary data. Depends on the provider.

Additionally

Q: What is the id/name of an original? Is it important at all?

Q: How do we create a hierarchy of originals? We want to avoid a flat structure in JCR

We must remove the original when the last asset linking to it is removed

Future improvements

Reuse of content when asset and original is identical to reduce the size of the repository