Concept - Light dialogs

- Introduction
- Goal
- Configurable dialogs
- Refactoring

Notes on Current Dialogs

Introduction

This concept is meant to briefly analyse what is needed in the current Magnolia dialog implementation to have light dialogs.

By light dialog is meant a modal dialog which is generally smaller than a normal dialog and which does not make the background opaque. This should make for a less obtrusive UI element allowing a more fluent, natural working flow in some cases. For a full rationale behind light dialogs, please see Use a light dialog to collect quick input

Goal

- Implement light dialogs (duh!)
- Make dialog modality, e.g. **strong**, **light** configurable.
- Possibly clean up dialog related code

Configurable dialogs

We'll add the following methods to interfaces

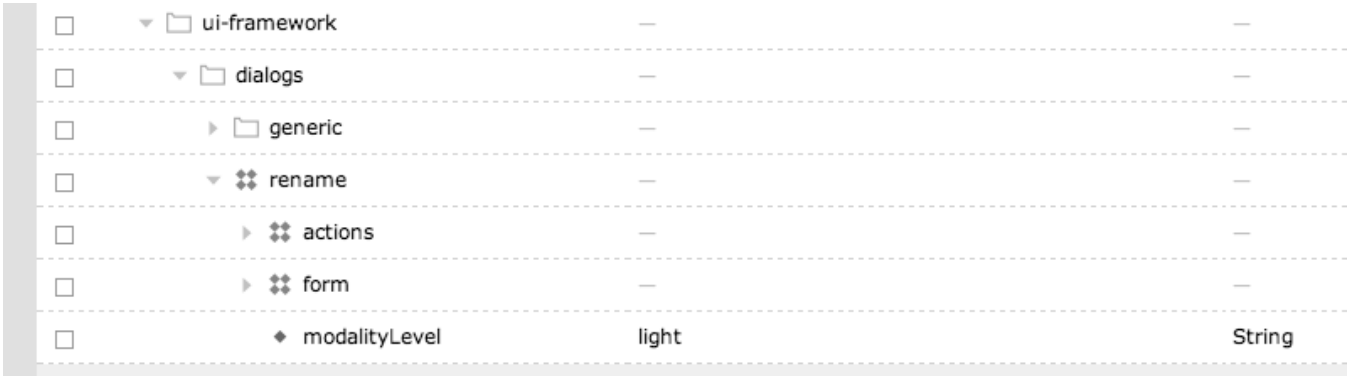
- ModalityLevel *DialogDefinition*.**getModality()**
- *DialogView*.**getModality()**
- *DialogView*.**setModality(ModalityLevel)**

info.magnolia.ui.api.overlay.OverlayLayer.ModalityLevel is an enum already available with **STRONG** (modal dark background), **LIGHT** (modal no background) and **NON_MODAL** modalities.

Below a sample of a light dialog configuration. As Node2Bean supports case insensitive enums

MAGNOLIA-5627 - Getting issue details... STATUS

one can also use lowercase enum.



BaseDialogPresenter

- reads DialogDefinition.getModality()
- calls DialogView.setModality(ModalityLevel)

BaseDialogViewImpl

- adds proper styling to dialog root

Use OverlayPresenter.openOverlay(DialogView, Modalitylevel) to open all dialogs. The Modality will be obtained via DialogView.getModality()

The following places in ui codebase should cover all cases we need.

- info.magnolia.ui.dialog.choosedialog.ChooseDialogPresenterImpl.start(ChooseDialogCallback, ChooseDialogDefinition, UiContext, String)
- info.magnolia.ui.dialog.formdialog.FormDialogPresenterImpl.start(Item, FormDialogDefinition, UiContext, EditorCallback)
- info.magnolia.ui.contentapp.movedialog.action.OpenMoveDialogAction.execute()

Refactoring

What about info.magnolia.ui.vaadin.dialog.LightDialog? It is used to create alerts and notifications, however could that naming be source of confusion with the light dialog concept itself?

After further investigation, it looks like the class hierarchy and views are inconsistent for our notifications, alerts and confirmations:

- UiContext has:
  - openNotification (Notification)
  - openAlert (LightDialog)
  - openConfirmation (ConfirmationDialog)
- These all share the 'light-dialog-panel' appearance.
- It is inconsistent that Notifications use their own "View" whereas LightDialog and ConfirmationDialog are just vaadin components, eventually inheriting each other without much sense.

We could/should have:

1. The Notification approach is quite right
   a. using a view to build/configure vaadin components
   b. but it should also use a so-called LightDialog Vaadin component internally
2. ConfirmationDialog should then turn into a View as well, not a component inheriting LightDialog
3. We miss the Alert
   a. directly instantiated as light dialog in OverlayPresenter
4. Rename Notification into NotificationView, ConfirmationDialog becomes ConfirmationView, and we also add AlertView
   a. and we move all those views OUT of common-widgets.
   b. common-widgets shouldn't depend on ui-api (where the View interface comes from)
   c. ui-framework is a decent candidate, where overlay presenter currently is
5. Hopefully the real light dialog views can be built in a similar fashion
   a. maybe complex base dialogs should 'extend' light dialogs and not vice-versa
   b. complex dialog = light dialog + header + help texts + introductory text

Notes on Current Dialogs

	Extends (Inherits)	IHas
BaseDialog	AbstractComponent (HasComponents)	• closeSelf() • DescriptionVisibility • setContent() • setHeaderToolbar() • setFooterToolbar() • DialogCloseEvent
LightDialog	BaseDialog	class=light-dialog-panel
ConfirmationDialog	LightDialog	• setMessage() • cancelButton and confirmButton added with setFooterToolbar() • ConfirmationEvent

BaseDialogViewImpl	Panel (DialogView)	• Has a BaseDialog as a member! "dialog". As its content. Wraps many dialog methods. • setDescriptionVisibility • Specific width 720px • 100% height for Dynamic dialog shrinking. • class=dialog-panel • addShortcut / removeShortcut
BaseDialogPresenter	(DialogPresenter, ActionListener)	• Builds from a DialogDefinition via a Dialog Factory. • Has a DialogView • ActionExecutor • EditorActionAreaPresenter • Adds shortcuts for save, cancel, escape.

Where should keyboard commands be handled?

- Across all dialogs
  - ESCAPE key is handled.
  - heavydialog
    - confirmation to close or not. "unsaved work"
  - lightdialogs, confirm, alert, notifications
    - closes dialog.
- On Heavydialogs, Confirmation, Alert dialog
  - ENTER key should commit or confirm.