Workflow Configuration and Actions

Starter

Enabling Workflow for content apps is very hard. It's one of the main reasons why customers complain and also one reason why I have been very reluctant to install the workflow by default for DAM e.g.

The idea of this document is to provide a concept for making workflow configuration easier. When looking at info.magnolia.module.workflow.setup. WorkflowBaseModuleVersionHandler you see the problems we are dealing with when installing workflow for pages app. This is also a problem for people trying to understand how stuff work together.

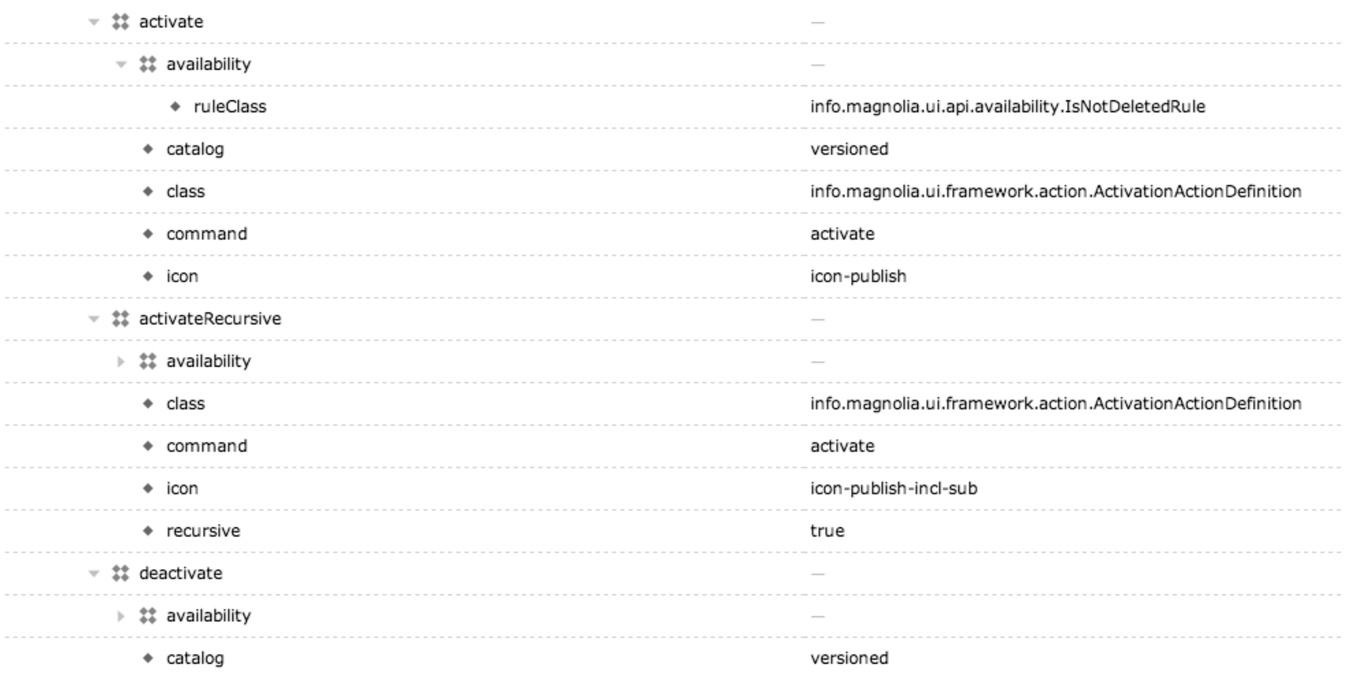
By cleaning up the configuration the documentation will be easier, providing Install tasks to install or even deinstall workflow per content app will be much easier, we will solve problems we have with update tasks failing and so on.

Current Problems

I have identified two problems. First of all, in CE we never use extends for actions in content apps. Second, workflow action configuration is tedious.

CE actions

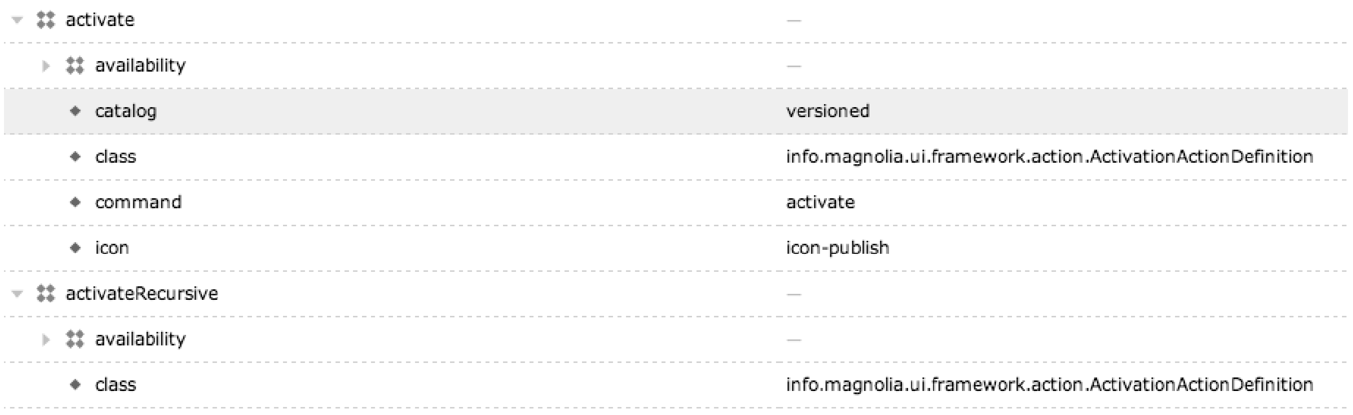
Here's an example of actions used for activation inside one random content app. These actions should be extending each other, this would make changing properties much easier.



activate	—
availability	—
ruleClass	info.magnolia.ui.api.availability.IsNotDeletedRule
catalog	versioned
class	info.magnolia.ui.framework.action.ActivationActionDefinition
command	activate
icon	icon-publish
activateRecursive	—
availability	—
class	info.magnolia.ui.framework.action.ActivationActionDefinition
command	activate
icon	icon-publish-incl-sub
recursive	true
deactivate	—
availability	—
catalog	versioned

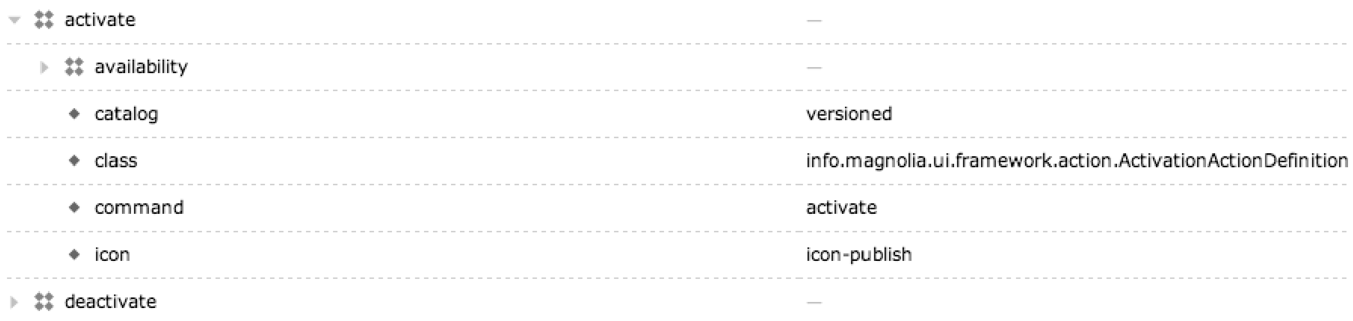
Further more those actions are again reused inside each and every content app with minor to none changes. Here's a comparison of the asset app with contacts:

Assets:



activate	—
availability	—
catalog	versioned
class	info.magnolia.ui.framework.action.ActivationActionDefinition
command	activate
icon	icon-publish
activateRecursive	—
availability	—
class	info.magnolia.ui.framework.action.ActivationActionDefinition

Contacts:



activate	—
availability	—
catalog	versioned
class	info.magnolia.ui.framework.action.ActivationActionDefinition
command	activate
icon	icon-publish
deactivate	—

Proposal

If we would move those Content App related actions into a generic node tree, which we could extend, we would make our lives much easier.

Workflow Actions

First off, the same proposal applies to workflow actions. There should be a generic node tree inside the workflow module, which you could easily extend.

But here we have a bigger problem. When we launch a publication workflow we do several steps before we actually launch the workflow: First we open a dialog, which allows us to add a comment and publication date as parameters. We add the message view parameter, which is used to render the pulse message and so on.

This is currently solved by defining callback actions, and launching actions from inside an action by injecting the actionExecutor.

Instead of launching the workflow directly then clicking on publish (in CE this launches the activate action) we have configured a "startPublication" Action which opens a dialog for adding needed parameters and this dialog then calls back into the activate action, which launches the workflow command.

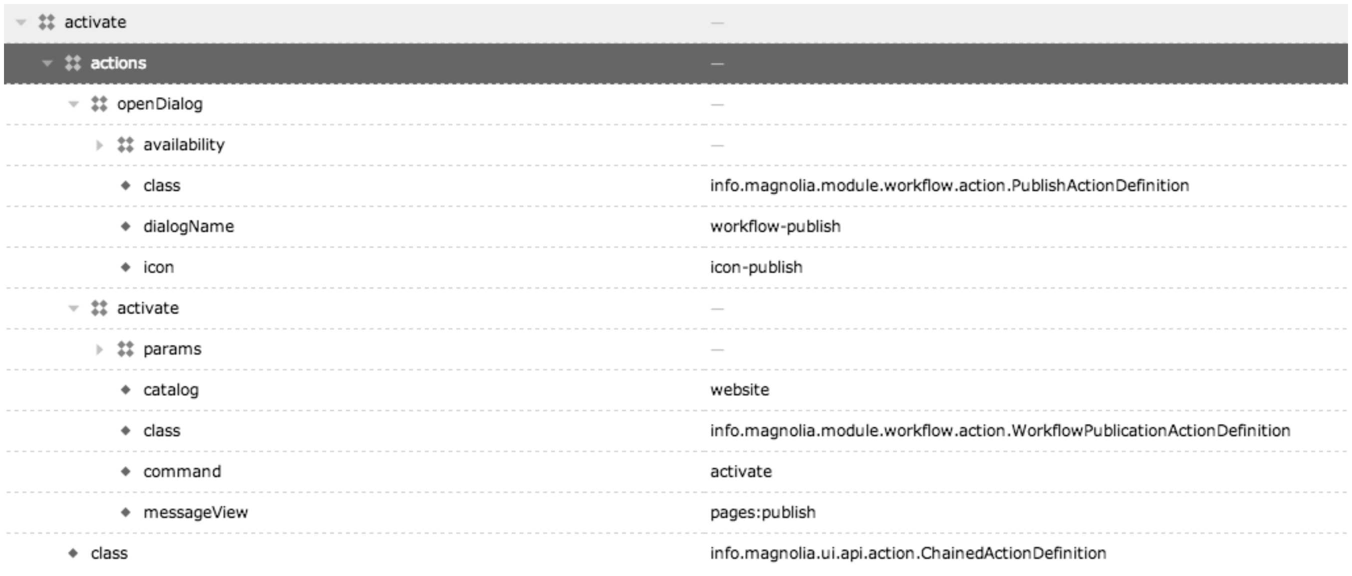
I don't think this is all wrong, but it is a total mess in the configuration tree and very hard to explain to customers and ends up in total craziness in the MVHs.

Proposals

Chained Actions

We could solve this by reusing the concept used for chaining commands. Inside an action you can define a chain of actions to be executed. This approach has been prototyped and does not need API changes, as it was done.

Here's an example of the configuration:



```
activate                      —
  actions                     —
    openDialog                —
      availability            —
      class                   info.magnolia.module.workflow.action.PublishActionDefinition
      dialogName              workflow-publish
      icon                    icon-publish
    activate                  —
      params                  —
      catalog                 website
      class                   info.magnolia.module.workflow.action.WorkflowPublicationActionDefinition
      command                 activate
      messageView             pages:publish
    class                     info.magnolia.ui.api.action.ChainedActionDefinition
```

The reason why I prefer this one is because it is generic and can be reused for confirmation dialogs. You want to open some sort of dialog before doing any action, just chain it.

Here's a code snippet from the ChainedAction prototype. One problem I ran into is that I somehow have to stop or pause the chain execution, when waiting for user input. I solved this by introducing two interfaces `ChainedActionCallback` and `DeferredAction`. If one of the action sin the chain is a DeferredAction we sto pthe execution and the Action will have to call proceed on the callback to resume execution.

ChainedAction Prototype

```
public class ChainedAction<D extends ChainedActionDefinition> extends AbstractAction<ChainedActionDefinition>
implements ChainedActionCallback {
    private final Item item;
    private final Context context;
    private final ComponentProvider componentProvider;
    private Iterator<Action> it;
    public ChainedAction(D definition, final Item item, ComponentProvider componentProvider) {
        super(definition);
        this.item = item;
        this.componentProvider = componentProvider;
        this.context = new SimpleContext();
    }
    private void prepareExecution() throws ActionExecutionException {
        Collection<Action> actions = new LinkedList<Action>();
        for (ActionDefinition actionDefinition : getDefinition().getActions()) {
            Action action = createAction(actionDefinition, item, context, this);
            actions.add(action);
        }
        this.it = actions.iterator();
    }
    @Override
    public void execute() throws ActionExecutionException {
        prepareExecution();
        executeNext();
    }
    private void executeNext() throws ActionExecutionException {
        while(it.hasNext()) {
            Action action = it.next();
            action.execute();
            if (action instanceof DeferredAction) {
                break;
            }
        }
    }

    private Action createAction(ActionDefinition actionDefinition, Object... args) throws
ActionExecutionException {
        Class<? extends Action> implementationClass = actionDefinition.getImplementationClass();
        if (implementationClass == null) {
            throw new ActionExecutionException("No action class set for action: " + actionDefinition.getName());
        }
        Object[] combinedParameters = new Object[args.length + 1];
        combinedParameters[0] = actionDefinition;
        System.arraycopy(args, 0, combinedParameters, 1, args.length);
        try {
            return componentProvider.newInstance(implementationClass, combinedParameters);
        } catch (MgnlInstantiationException e) {
            throw new ActionExecutionException("Could not instantiate action class for action: " +
actionDefinition.getName(), e);
        }
    }

    @Override
    public void proceed() {
        try {
            executeNext();
        } catch (ActionExecutionException e) {
            e.printStackTrace();  //To change body of catch statement use File | Settings | File Templates.
        }
    }
    @Override
    public void abort() {
    }
}
```

Action Composition

Meaning building more complex code and the current Action API is not flexible enough to do it the right way IMO.