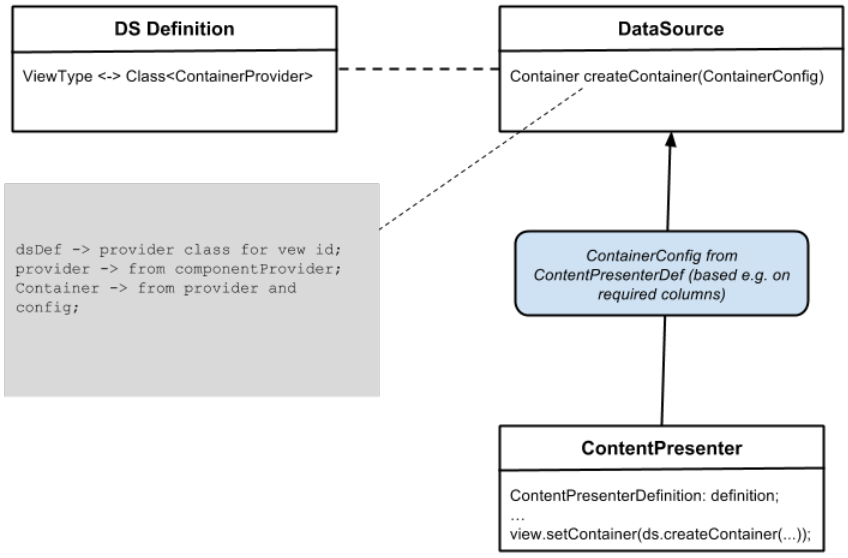
JCR-agnostic content apps - API Changelog

This note attempts to give an overview of topics covered during the JCR agnostic container spike, which required significant API changes - although many of them are currently work-in-progress.

- 1. ContentSubApp configuration.
- 2. ActionExecution
- 3. Availability Rules
- 4. ActionAvailability -> ActionDefinition/ActionbarSectionDefinition
- 5. ContentAppEvents
- 6. ImageProvider interface
- 7. Inplace-editing
- 8. Actionbar refactoring
- 9. To be done

1. ContentSubApp configuration.

- Introduced DataSource interface which allows for connecting different data domains to the sub-apps. (**Proposed** to rename into ***`ContentConnector`***).
- JcrDataSource operates with JcrItemId pojo (workspace + uuid) instead of just uuid.
- `DataSourceDefinition getDataSource()` method was added in order to be able to configure DS dynamically.
- DS implementation class is stored inside `DataSourceDefinition`.
- However, DS can be directly injected in sub-app scope by means of `DataSourceProvider` (Guice provider implementation)
- Creating container for a content view. Current state:



Good points:

- All the data is managed in DS.
- Stock content presenter can be used out of the box with other types of data.

Bad points:

- Quite complex: ContainerConfig/ContainerProvider is something to explain.

Proposal: we let the content presenters instantiate containers themselves and if some other container is needed - override the content presenter.

2. ActionExecution

- Using Vaadin Item instead of javax.jcr.Item in `ActionExecutor#isAvailable()` method.

3. Availability Rules

- Using Vaadin Item instead of javax.jcr.Item in `AvailabilityRule#isAvailable()` method.

4. ActionAvailability -> ActionDefinition/ActionbarSectionDefinition

- In order to get rid of JCR-dependent and very specific action/section availability verification we introduced the Voter-based mechanism.
- Some extra details can be found here: Permissions for UI availability.
- All the criteria that existed before (nodes, properties, root, multiple item support, availability rules, access etc) are now not used within ContentSubApp (BrowserSubApp) logic and handle universally with Voters.
- We though still support all the short-hand configuration properties we used before by means of generating corresponding Voters on the fly with a custom transformer.
- All in all what is now contained in availability is just a list of voters.
- Analogous to the AccessDefinition which is capable to check user access, availability is capable of invoking the Voting mechanism (in order to save some code lines and hide implementation logic from the user).
- The longish and multiple-nested-if-cluttered methods like `BrowserSubApp#isSectionVisible` are transformed into one-liners:

```
private boolean isSectionVisible(ActionbarSectionDefinition section, List<Item> items) {
    return section.getAvailability().isAvailableForItems(items);
}
```

5. ContentAppEvents

- `SelectionChangedEvent` no longer carries the the Items - just the ids

6. ImageProvider interface

- Got rid of path-based query methods (were just convenience methods duplicating UUID-based ones).
- Replaced `String` ids with arbitrary `Object`.

7. Inplace-editing

- Removed InplaceEditingTreeTable
- Replaced ItemEditedEvent with ActionEvent + SaveItemPropertyAction

8. Actionbar refactoring

- ActionbarView interface was moved from ui-vaadin-common-widgets to ui-actionbar
  - ✗ info.magnolia.ui.vaadin.actionbar.ActionbarView
  - ✓ info.magnolia.ui.actionbar.ActionbarView
- ActionbarPresenter #start now requires (ActionbarDefinition, Map<String, ActionDefinition>)

9. To be done

- Tests
- Update tasks
- ...