Permissions for UI availability

This concept describes the current state of permission-based configuration - or lack thereof - for enabling/disabling actions (or any other UI element) based on specific access restrictions.

Decisions are marked with a ✅ icon.

Concept is ready for implementation, both for 5.2.x and for a future major version.

Problem

One can restrict (action) availability based on user roles, but not based on user permissions at given workspace / path.

Concrete case

As of Magnolia 5.2.1, actions are not disabled if user has no permission to act on selected node. This is an issue with e.g. read-only pages, as captured by the following Jira ticket:

~~MGNLUI-2510~~ - UI shouldn't enable actions for which the user has no permissions CLOSED

Current configuration

Let's take the deactivation action as an example, with following base path /modules/pages/apps/pages/subApps/browser/actions.

- actions
  - deactivate (`ActionDefinition`)
    - availability (`AvailabilityDefinition`)
      - access (AccessDefinition)
        - roles
          - demo-publisher = demo-project-publisher
          - superuser = superuser
      - ruleClass = info.magnolia.ui.api.availability.IsNotDeletedRule (`AvailabilityRule`)
    - class = info.magnolia.ui.framework.action.DeactivationActionDefinition
    - ...

As a side note, the AccessDefinition property is named *access* in the case of actions but it is generally named *permissions*, as in AppDescriptor or AppLauncherGroupDefinition.
This has been reported to be misleading, in particular in documentation, and should be inlined whenever is appropriate.

Decisions

1. Configuring permission checks in availability

- ~~Add a *requiredPermissions* property under AvailabilityDefinition or AccessDefinition~~
  - comma separated list of JCR permissions (aka action strings)
    - add_node, set_property, remove, read
    - we should rather use Magnolia permissions
    - ⛔ doesn't fit for upcoming custom permissions
  - naming is debatable (*permissions* > *requiredPermissions*)
- ✅ **For 5.2.x we add a *writePermissionRequired* boolean property under AvailabilityDefinition**
  - simply checks for Magnolia Permission.WRITE permission
  - we add this to the availability evaluation sequence
    - as of 5.2.2 this is in AbstractActionExecutor (action availability) and in BrowserSubApp (section availability)
    - use PermissionUtil when processing
  - for custom permissions, people need to implement AvailabilityRule
  - ⚠️ After another round of reviews, **we ultimately decided against using voters for availability**
    - We found important to instantiate whatever criteria (voters or rules) on subapp scope
    - In order to make it possible to inject components in these rules - which are then also resolved on subapp scope

        - as opposed to voters which are instantiated by n2b on global webapp startup
        - similarly as we have now *ruleClass* configured in `ActionAvailability`, which is instantiated on the fly by subapp's *componentProvider*.
      - We did not want to revise voters or introduce voter definitions at this time
      - We also chose the flag approach for 5.2.x so that we don't introduce any new mechanism and leave the door open to finalize the proposal for 5.3.
  - For 5.3 we now aim at improving use and flexibility of the `AvailabilityRules`.
    - by configuring a collection of such *rules* in `AvailabilityDefinition`, instead of one single *ruleClass*, so that we can compose multiple rules
    - by introducing an `AvailabilityRuleDefinition` to make these rules configurable
    - yet to be decided
- ~~We unify availability's access, ruleClass and other criteria using voters, in a future major version~~
  - supports custom permissions (forum), even non-JCR based, using dedicated voters
  - Do we keep availability's "shorthands"?
    - *nodeTypes*, *root*, *properties*...
    - yet update underlying implementation to work with voters
    - **Proposal:** how about maintaining all the shorthands we have and also providing a rather smooth mechanism of moving from old impl to the new one by implementing a custom `Node2BeanTransformerImpl`
      that would build voters based on the properties from `AvailabilityDefinition` (e.g. once the property name is ***nodes*** - we generate a corresponding voter)?
- ~~For 5.2.x, we introduce a delegating AvailabilityRule which helps us already start working with voters~~
  - getting well prepared for migrating to the next approach

2. The *add_node* permission with subnodes-only ACLs

- with /A readonly and /A/B/* read-write, add_node is not granted on /A/B
- JCR spec is a bit unclear as to what absPath means in that case
  - adding a node *at* absPath VS. adding a node *under* absPath
- 4.5 behaves the same in similar subnodes-only permissions
- **Current behavior is actually correct against JCR permissions**
  - add, move, reorder all require write permission on parent node

3. ActionExecutor is responsible for availability checks

- Currently hooking in AbstractActionExecutor#isAvailableForItem
- item is null when root is selected, no way to assess permissions then
- #isAvailable() is (the sole) JCR Item dependent api in ActionExecutor interface and doesn't belong here
- **We keep this as a separate topic, not for 5.2.x anyway**
  - **We may cover that for 5.3** (by e.g. having a single AvailabilityChecker component).

Forward thinking

- Availability / AccessDefinition is a broad concept meant to be reused across several UI components (e.g. fields, tabs, templates).
- Can one configure custom permissions for an action? e.g. forum moderator can only perform moderation at specific path
- Can one plug basic permission rules for non-JCR datasources (no ACLs)?
- ActionExecutor is probably not where availability / permission checks belong.