# FakeMagnoliaUI future

## What to do with the JcrFormEvironment

- Option1: Migrate JcrFormEnv → FMUI
  - no need to do extra work there right now, the technical debt isn't big
- ✅**Option2: Do not migrate anything yet → migrate everything to JUnit5 setup**
  - would save time
  - when proper JUnit 5 extensions land - we move all the test together
  - JcrFormEnv and FMUI aren't so different, should be easy to take care of them at once

## How to proceed extracting the FMUI from the fw-jcr artifact

- Where we want the FMUI to be?
  - FMUI extension will be in UI reactor itself
  - What to do with the dependency on the framework itself?
    - FMUI depends heavily on UI.fw parts
    - it would only be possible to write the FMUI-involving tests for the `ui.fw-core` if FMUI was in the `fw-core/test`
    - or we (realistically) can concentrate the end-2-end tests not in the UI.fw core, but in reference implementations (JCR/REST) and in concrete modules
- We need repo support [in JUnit 5] (from core)
  - most of the times we're good with mock sessions, sometimes - with real ones
    - ideally - it's the test setup (annotations/extensions etc) drive the flavour of JCR support (mock/real)
  - currently there's a PoC by Evzen

- not integrated, cannot be a part of test fw (yet), because of strong dependencies to core codebase
- will probably reside in core reactor for the time being (maybe a separate artifact)

## What to do to turn FMUI to JUnit 5 setup

- How do we even write the tests and populate the mock environment
- Breaking down the DSL:

```
magnoliaUI
  .withLightModuleName("fooModule") // similar things are done for UI tests with
 @LightModule annotation, can be separate ext
  .withModule("ui-framework-jcr") // same here probably (???)
  .readContentTypes()
  .withI18(Locale.GERMAN)
  .withJcrWorkSpaces("config") // could be chipped out in a separate ext
  .start();
```

- IoC
  - connect JUnit 5 param resolution with the Mgnl Component Provider for seamless injection of components in test vs `magnoliaUI.getComponentProvider().getComponent(...class)`
- Jcr repo support
  - to be provided as part of the effort in the core (take Evzens PoC further)
- Admincentral setup
  - primitives like app/sub-app ctx's and AppController/LocationControllers etc to be provision-able with the IoC effort (☝)

## Concerns

- calculating coverage might be a tad tedious with end-to-end tests
  - FMUI implicitly increases the coverage but that is not reflected in numbers

- Simple JUnit tests still [should] provide the sufficient coverage on their own (and we're less concerned by the coverage brought in by the end-to-end tests)?
- Extraction of functionality should encourage test coverage distribution somewhat (although the current tests will probably remain in the fw-jcr)
- Still the point is valid - how do we quantify the test coverage
  - TODO: check the possibilities with clover maybe, although the hope is slim there
- Point: the fact that the upstream code is used in the test doesn't mean that the test actually tests it (it uses it merely) and probably shouldn't be considered in coverage?
- Very similar issue with UI tests (even worse so)

## Ticketising strategy

- Use https://jira.magnolia-cms.com/browse/BUILD-408 as starting point
- Start with the outline of basic extensions that supersede the FMUI, just a skeleton
  - consider the maven coordinates (separate artifact probably? in core and UI, would be cleaner and would let us avoid inter-deps and such)
  - we do not touch the JUnit 4 based facilities (FMUI/FormEnv)
  - do not port the test
- Go step by step
  - ensure that we can provision the UI/Sessions
  - outline extensions for IoC
  - finalisation of core efforts re: JCR repo support and such
  - porting the tests (effort shouldn't be dramatic, it should be *much* easier to update those)