Support for cascading Yaml configuration update

- ~~MAGNOLIA-6451~~ - YAML configuration: changes in included files don't trigger a reload of the registered item CLOSED , why it happens?
  - we do not track the fact of inclusion
  - included files may be located anywhere
  - no notion of config dependency from data-structure point of view:
    - each config file potentially could have an inclusion tree: included files also may have includes
- **dynamic definition providers**
  - The way population of a definition provider happens now:
    - **Yaml —> YamlCfgSrc \ / —> DefinitionProvider** (*static, i.e. everything is pre-resolved*)

      \YamlReader ——> M2B - *(produces the bean)/*

      \

      (*resolves all the includes recursively and populates a map; from this point on the inclusions are lost*)
- - Proposed way:
    - **Yaml —> YamlCfgSrc —> YamlDefinitionProvider** *(no conversion happens right away, it only has reference to a resource and tools like YamlReader and M2B).*
    - **YamlReader** would still produce a map with converted Yaml,
      - + it would populate a set of such inclusions with references to respective files.
    - Upon each **provider#get()** call the last modified dates of the dependencies are checked, the definition bean is re-resolved if needed
    - Cons:
      - small overhead upon definition retrieval (**DefinitionProvider#get()**)
    - Pros:
      - easy to implement (https://git.magnolia-cms.com/projects/FORGE/repos/light-yaml/browse)
      - easy to map onto other cases of dependencies (JCR, Registries etc)
      - each def knows exactly what it relies on, the changes of the non-config related resources are essentially ignored
      - easy to make prominent in definitions app
- alternative way
  - we could track the dependencies outside of the definition provider and still use the static definition providers
    - resolve the dependencies as soon as a config file is registered, store them (on the YamlCfgSrc level)
    - => there would be a set of dependencies per config file
    - upon any resource change - check the sets of deps of each config file on the presence of the changed one
    - Cons:
      - cumbersome
      - would cause tons of useless checks
      - hard to make extensible
    - Pros:
      - DefProviders are still dumb and simple
      - no overhead for **DefinitionProvider#get()**