How to Deprecate YAML Definitions

Introduction

Currently we have no possibility to mark an YAML definition as deprecated (not talking about the java classes).

DEV-713 - Jira project doesn't exist or you don't have permission to view it.

is created to investigate the problem at hand.

Potential Solutions

By YAML Tag

We could use a custom YAML tag to achieve this.

It'd look something similar to:

```
!deprecated:5.6:some-reason
```

Although it looks great for the job at hand, conceptually it's not really what we want to have. Because we won't be doing anything rather than communicating the deprecation to definitions app.

For other similar tags we have constructs were fine because we were actually inserting some information there and it was the correct approach for those tasks.

Pros

- Can potentially deprecate partial configuration
- Looks nice in YAML configuration
- We already have similar experiences (inherit, decorate and such)

Cons

- Construct
- No space is allowed not to corrupt the document (check Further research part for more detail)

By File Name

We could indicate that YAML files are deprecated by simply adding deprecated in the file name as a prefix.

e.g.

```
deprecated.yamlDefinition.yaml
```

Pros

- No need to change definition content

Cons

- Can only deprecate the whole file, hence no possibility to deprecate sub-definitions of the definition
- How to handle the name change of particular definition → system should do some resolution automatically which sounds nasty at the first place.

By YAML Property

Yet another idea would be to have default property for all definitions.

If we could introduce a property which will capture all the definitions, then it'd make sense to have one default property which is set to false but definable by the user via YAML.

Pros

- Users have to only define a property in their YAML definitions, would work like a construct but without constructing anything.
- Looks similar to what we will get from JCR

Cons

- Need to change java code
- Not sure if it's even cool to do it that way

Further findings

It turns out that we have an issue when one tries to define the description with empty spaces. Unfortunately we hit to a YAML limitation and it's not possible to parse the following:

```
!deprecated:5.6,some reason
```

To work it around we have two possible options:

Have predefined description separator

For instance we can say that our descriptions will contain '_' or '-' instead of spaces and we will simply replace them while we parse the YAML file:

```
!deprecated:5.6,some_reason
```

Use properties to describe since and description

With tag

We can still use !deprecated tag for marking the file as deprecated but the information of it will be stored in plain YAML properties like the following:

```
!deprecated
mgnl:deprecatedSince: 5.6
mgnl:deprecatedReason: Some reason
```

Without Tag

Or without the tag we could simply have it like:

```
mgnl:deprecated:
 since: 5.6
 description: some_reason
```

Decision

In the end we have decided to go with approach 2.b. This is due to the fact that It is a no-go to have description written without spaces, we simply can't ask that from customer or even from our devs. Also everything related to deprecation information is now in one place (node) and properties and are sub-properties of the node.