NRP MODEL CHEAT SHEET

LAYOUT - YAML

```
price:
  type: float
```

— or —

```
price: float
```

SIMPLE TYPES

```
integer
boolean
float
double
date (iso)
time (iso)
datetime (iso)
edtf (level 0)
edtf-interval (level 0)
```

TEXT TYPES

```
fulltext
keyword
fulltext+keyword
```

Examples:

```
title: fulltext
published: edtf
```

CONSTRAINTS

```
price:
  type: float
  minimum: 0
  required: true
```

— or —

```
price: float{minimum:0,
required: true}
```

Supported:
https://t.ly/4Met

OBJECTS

```
person:
  properties:
    firstName: keyword
```

— or —

```
person{}:
  firstName: keyword
```

EXTRAS ON OBJECTS

```
person:
  label.en: “Person”
  properties:
    firstName: …
```

— or —

```
person{}:
  ^label.en: Person
  firstName: …
```

everything starting with ^ goes to person element, not person.properties

NESTED

Why to use nested? See …

```
person:
  type: nested
  properties:
    firstName: keyword
```

— or —

```
person{nested}:
  firstName: keyword
```

EXTRAS ON NESTED

same as on objects

CUSTOM TYPES

Custom datatypes are supported, see …. for details

ARRAYS

```
prices:
  type: array
  items:
    type: float
```

— or —

```
prices: array{items:float}
```

— or —

```
prices[]:
  type: float
```

— or —

```
prices[]: float
```

EXTRAS ON ARRAYS

```
prices[]:
  type: float
  ^label.en: Prices
  ^minItems: 5
```

everything starting with ^ maps to array, not item

EXTRAS ON ARRAY ITEMS

```
prices[]:
  type: float
  minimumExclusive: 0
```

everything without starting ^ goes to item

ARRAYS OF OBJECTS

```
people{}[]:
  firstName: keyword
  lastName: keyword
```

do not use this as it is not readable, better way is $defs or importing the definition

INCLUDES

```
use: person.yaml
use: types.yaml#/Person
```

— or —

```
$ref: person.yaml
```

I18N USER INTERFACE

supported with oarepo-model-builder-ui

```
label.en: Person
label.key: metadata.person
title.en: Person tooltip
hint.en: Hint in deposit form
```

.key → generated to .po files as msgid, if not specified, json path within the model with removed “properies” and “items”, with added “.label”, “.title” and “.hint”

I18N VALUES

supported with oarepo-model-builder-multilingual

```
translatedTitle: i18nStr
```

— is —

```
translatedTitle{}:
  lang:  keyword
  value: keyword
```

OS mapping is the same, no special handling for different languages

```
translatedTitle: multilingual
```

— is —

```
translatedTitle:
  type: array
  items{}:
    lang:  keyword
    value: keyword
```

OS mapping per language, supported languages must be defined on root element, see docs at

NRP MODEL CHEAT SHEET

JSONSCHEMA OVERRIDES

```
title:
  jsonschema:
    minLength: 5
```

whatever is in jsonschema is merged with an automatic jsonschema generator (and overrides its settings). Marshmallow is not generated for such a section

MAPPING SETTINGS

```
model.yaml

model:
  metadata: …

  mapping:
    settings:
      analysis: …
```

everything defined on model root in “mapping” section goes to the top of the OS index definition file

MAPPING OVERRIDES

```
compound:
  type: keyword
  mapping:
    analyzer: chemcompound
```

everything defined inside “mapping” goes directly to index definition. Make sure the analyzer is defined in the mapping settings.

MARSHMALLOW

```
prop:
  type: object
  marshmallow:
   read: true
   write: true
   field-name: prop
   field: <if defined, use it
as it is>
   field-class: <if defined,
use it as a field class on
prop’s parent
   arguments: [“key=value”]
   schema-class:
<package>.PropSchema
   base-classes: [“ma.Schema”]
   generate: true
   imports: [{import:””,
alias: “”}]
```

VOCABULARIES (INVENIO FLAVOUR)

```
affiliation:
  type: vocabulary
  vocabulary-type:
affiliations
  fields: [id, file]
```

— or —

the following is on one line

```
affiliation:
vocabulary{type=affiliations,f
ields=[id,title]}
```

id and title are default, default model is “vocabulary”

```
language:
vocabulary{type=languages}
```

HIERARCHY VOCABULARIES (TAXONOMIES)

the same as above, but “taxonomy” is used

```
degree-grantor:
taxonomy{type=degree-
grantors,fields=[id,title]}
```

CUSTOM VOCABULARIES

vocabularies with custom properties can be created, see docs at …

MODEL DEFAULTS

model defaults are defined directly on the model element. The list below shows just the basic ones, to see all of them generate an empty model and look at the model.json file. Alternatively, consult the schema at …

```
model:
  package:
  record-class:
  record-base-classes: []
  properties:
    … model properties go here
```