odk

ONTOLOGY DEVELOPMENT KIT

Setting up your own ODK ontology repository

TIB

NFDI4Chem

Philip Strömert | February 8th | **Ontology Summit 2023**

10.5281/zenodo.7623878

Prerequisites: Your Setup

You will need

- ODK installed in a Docker environment
  - `docker pull obolibrary/odkfull`
- Git (e.g. Github Desktop)
- a GitHub or GitLab account

Prerequisites: ODK seed wrapper script

- a convenient way
  - to start the ODK Docker container
  - to pass it your working directory, Git username & email
  - and to call the ODK seed method with all its parameters
- download it from the ODK repo
  - seed-via-docker.sh (Mac/Linux) or seed-via-docker.bat (Win)
- it must be in your working directory
  - a temp folder for the seed process output (e.g. `~/my_ODK_saplings`)

Creating an ODK based repository

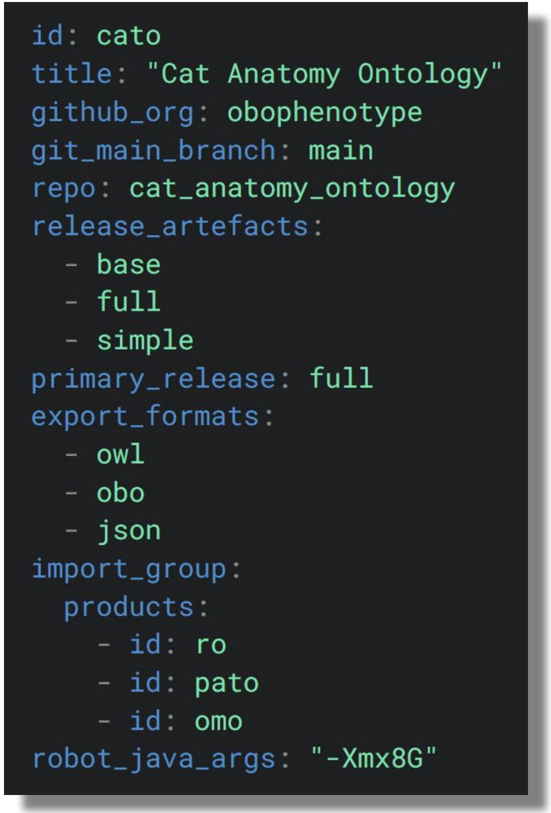
Step 1: Configure ODK parameters in a `project.yaml`

Step 2: Seed your repository with the wrapper script

Step 3: Upload the repo to GitHub / GitLab

→ Enjoy maintaining your ontology with ODK 😁

Step 1: Configure general metadata



```yaml
id: cato
title: "Cat Anatomy Ontology"
github_org: obophenotype
git_main_branch: main
repo: cat_anatomy_ontology
release_artefacts:
  - base
  - full
  - simple
primary_release: full
export_formats:
  - owl
  - obo
  - json
import_group:
  products:
    - id: ro
    - id: pato
    - id: omo
robot_java_args: "-Xmx8G"
```

id (required)

- your ontology acronym
  - used to build file names & term IDs
  - should be lower case, usually ~ 4 characters

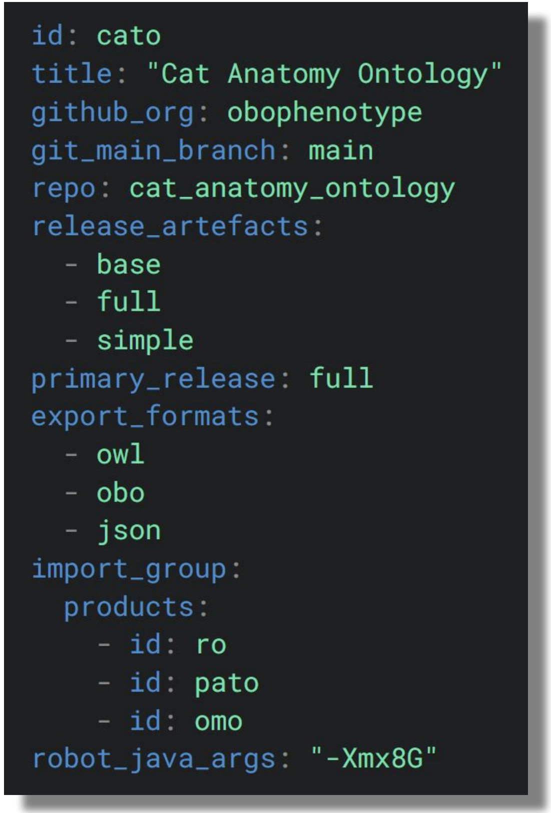
title (required)

- to generate various default values (e.g. in docs)

more is possible (optional)

- e.g. use `description`, `licence` or `creators` to generate dcterms ontology annotations

Step 1: Configure Git parameters



```
id: cato
title: "Cat Anatomy Ontology"
github_org: obophenotype
git_main_branch: main
repo: cat_anatomy_ontology
release_artefacts:
  - base
  - full
  - simple
primary_release: full
export_formats:
  - owl
  - obo
  - json
import_group:
  products:
    - id: ro
    - id: pato
    - id: omo
robot_java_args: "-Xmx8G"
```

github_org

- your GitHub or GitLab handle / organisation
    - used for some basic configs of the Git repo
    - defaults to your Git username

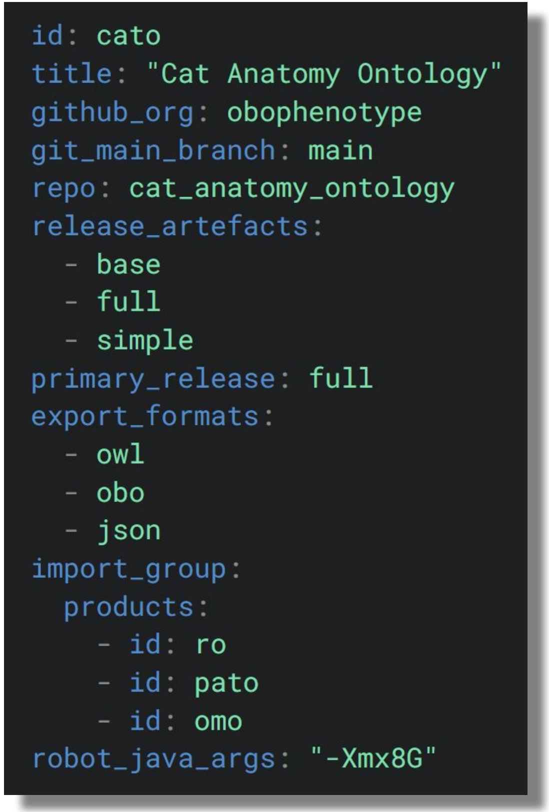
repo

- set the name of your repository

git_main_branch

- set the name of your main (master) branch

Step 1: Configure pipeline parameters



```
id: cato
title: "Cat Anatomy Ontology"
github_org: obophenotype
git_main_branch: main
repo: cat_anatomy_ontology
release_artefacts:
  - base
  - full
  - simple
primary_release: full
export_formats:
  - owl
  - obo
  - json
import_group:
  products:
    - id: ro
    - id: pato
    - id: omo
robot_java_args: "-Xmx8G"
```

release_artefacts

- six different logical types (sets of axioms) possible
- generated according to these OBO/ROBOT conventions

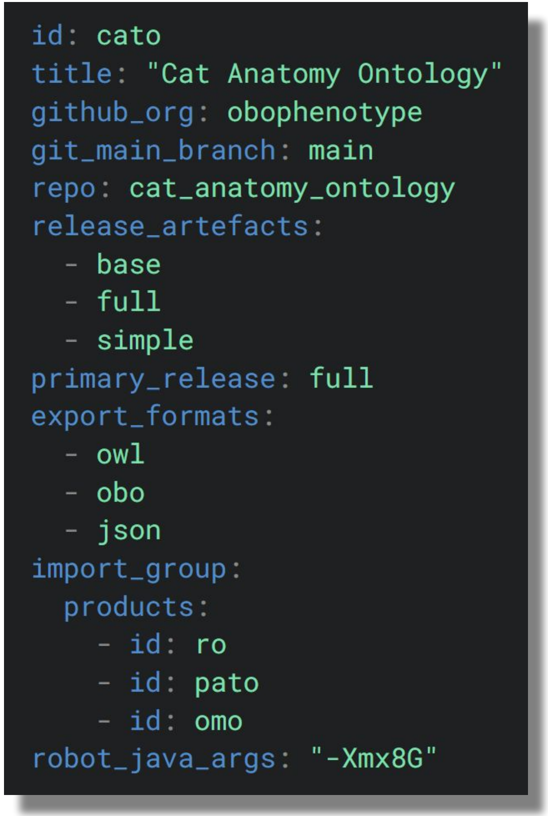
primary_release

- what logical type is to be your main release artefact

export_formats

- the file formats you want to provide, such as OWL, OBO, TTL or OBOGraphs JSON

Step 1: Configure import modules

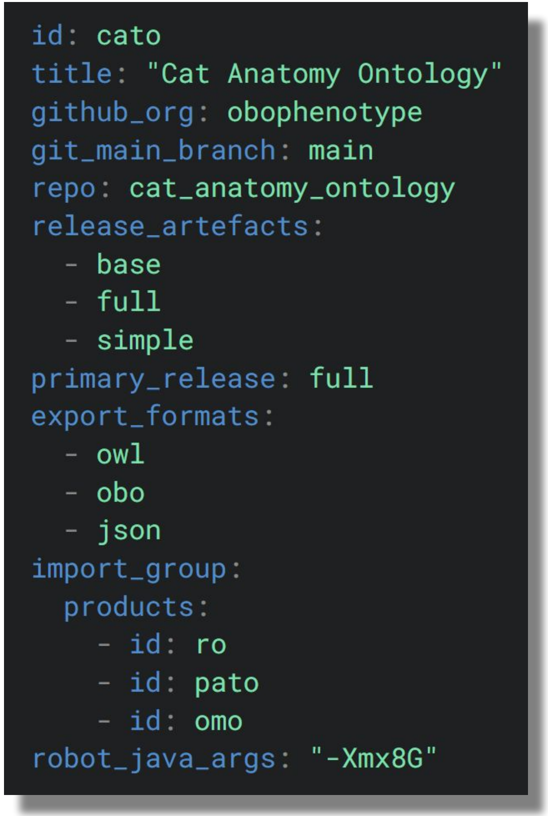


```yaml
id: cato
title: "Cat Anatomy Ontology"
github_org: obophenotype
git_main_branch: main
repo: cat_anatomy_ontology
release_artefacts:
  - base
  - full
  - simple
primary_release: full
export_formats:
  - owl
  - obo
  - json
import_group:
  products:
    - id: ro
    - id: pato
    - id: omo
robot_java_args: "-Xmx8G"
```

import_groups products

- lists all the ontologies you want to import
- for OBO ontologies `id` usually suffices
- but, you can configure much more, e.g.:
  - `mirror_from` → e.g. for non-OBO ontologies
  - `module_type` → to customize import modules
  - `is_large` & `use_gzipped` → for large ontologies

Step 1: Configure arguments passed to ROBOT



```
id: cato
title: "Cat Anatomy Ontology"
github_org: obophenotype
git_main_branch: main
repo: cat_anatomy_ontology
release_artefacts:
  - base
  - full
  - simple
primary_release: full
export_formats:
  - owl
  - obo
  - json
import_group:
  products:
    - id: ro
    - id: pato
    - id: omo
robot_java_args: "-Xmx8G"
```

robot_java_args

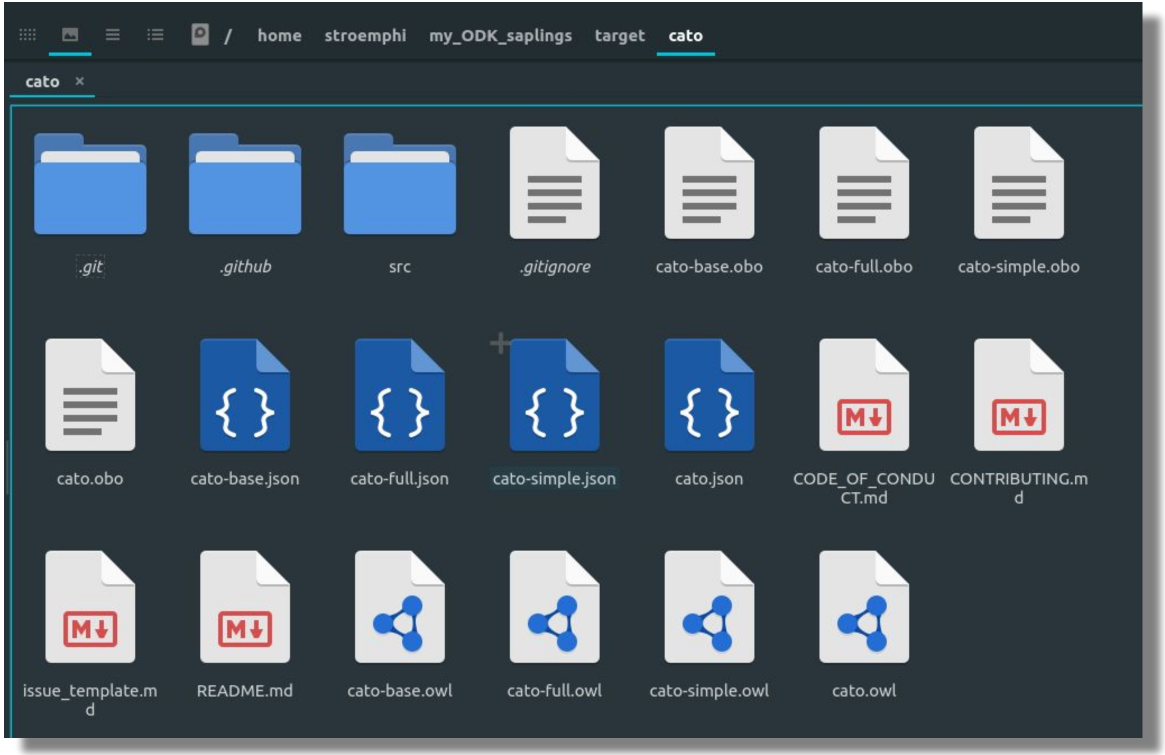
- e.g. set the max RAM used by ROBOT
  - CAVE: give Docker at least ~20% more

for other possible options

- many examples to learn from

Step 2: Let ODK seed your repo

```
sh seed-via-docker.sh -C project.yaml
```



.git

.github

src

.gitignore

cato-base.obo

cato-full.obo

cato-simple.obo

cato.obo

cato-base.json

cato-full.json

cato-simple.json

cato.json

CODE_OF_CONDUCT.md

CONTRIBUTING.md

issue_template.md

README.md

cato-base.owl

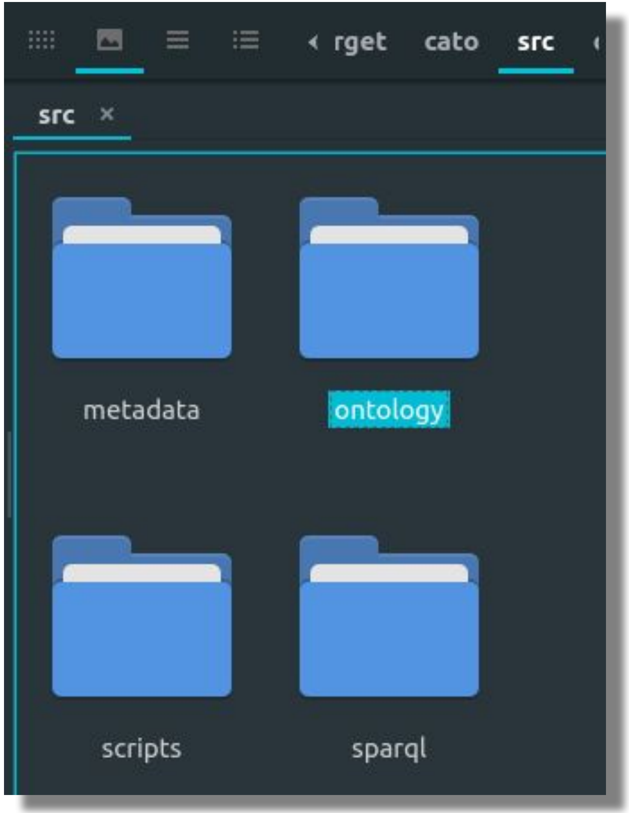
cato-full.owl

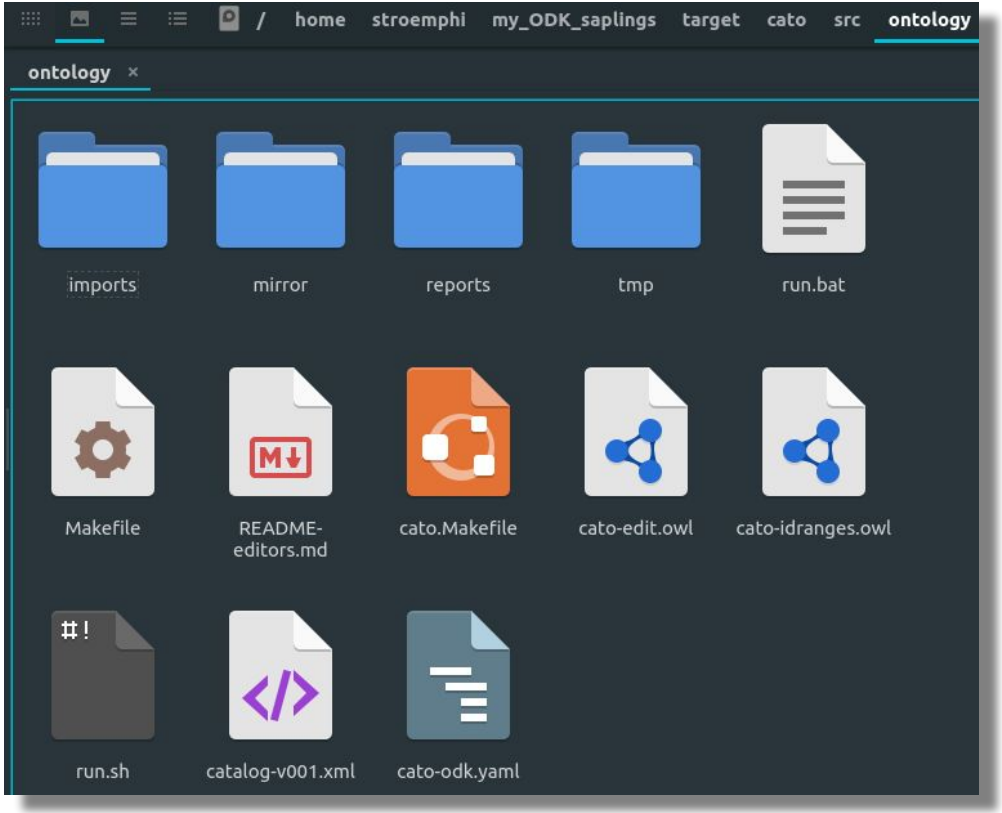
cato-simple.owl

cato.owl

- complete git repo
- release artefacts
- README
- templates
  - NTRs
  - contributing
  - code of conduct

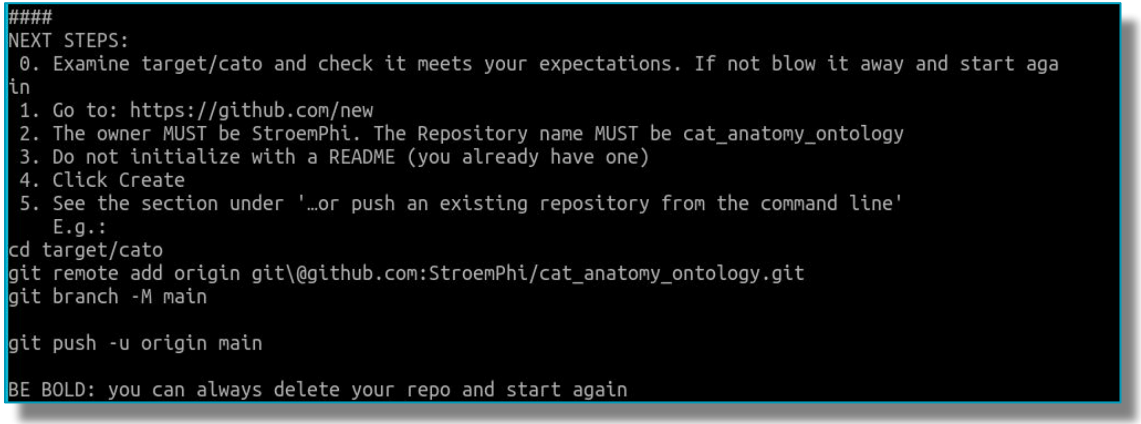
Your ODK repository workspace





Step 3: Upload the repo to GitHub / GitLab

- copy the generated dir to desired place (e.g. $HOME)
- in GitHub Desktop
  - `File → Add local repository`
  - `Publish the repository`
- on the command line → instructions provided in seed script output



```
####
NEXT STEPS:
 0. Examine target/cato and check it meets your expectations. If not blow it away and start aga
in
 1. Go to: https://github.com/new
 2. The owner MUST be StroemPhi. The Repository name MUST be cat_anatomy_ontology
 3. Do not initialize with a README (you already have one)
 4. Click Create
 5. See the section under '…or push an existing repository from the command line'
    E.g.:
cd target/cato
git remote add origin git\@github.com:StroemPhi/cat_anatomy_ontology.git
git branch -M main

git push -u origin main

BE BOLD: you can always delete your repo and start again
```

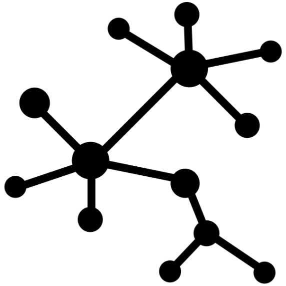
Enjoy maintaining your ontology with ODK

- build import modules automatically
  - add needed terms to the TXT files in ./src/ontology/imports
  - refresh with `sh run.sh make refresh-imports`
  - customize the module build process via your custom Makefile
- run the release pipeline
  - `sh run.sh make prepare_release`
- run all QC checks or check the OWL2 DL profile validity
  - `sh run.sh make test`
  - `sh run.sh make validate_profile_cato-edit.owl`

Continuous Integration Testing

Developer

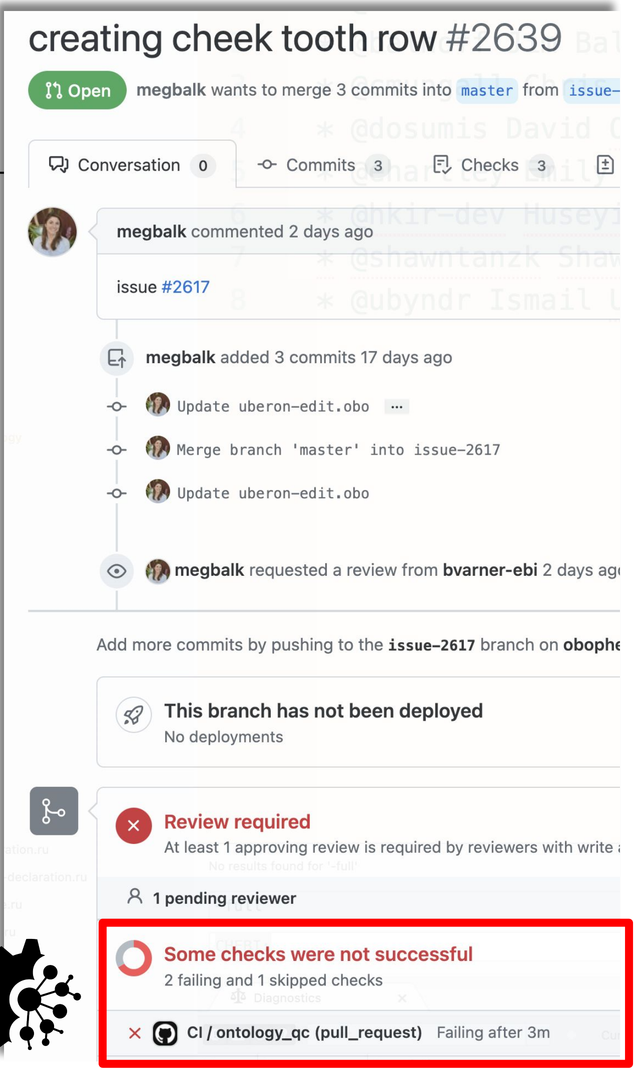
edit locally



Make pull request

CI System (GH actions) runs ODK checks

- No more broken ontologies on “main”
- No more fear you might “break stuff”
- Rich set of checks:
    - OWL profile checking
    - ROBOT report (incl. many best practices)
    - Customisable with SPARQL-based unit testing
    - Logical consistency



creating cheek tooth row #2639

Open megbalk wants to merge 3 commits into master from issue-

Conversation 0 · Commits 3 · Checks 3

megbalk commented 2 days ago

issue #2617

megbalk added 3 commits 17 days ago

```
Update uberon-edit.obo
Merge branch 'master' into issue-2617
Update uberon-edit.obo
```

megbalk requested a review from bvarner-ebi 2 days ag

Add more commits by pushing to the issue-2617 branch on obophe

This branch has not been deployed
No deployments

Review required
At least 1 approving review is required by reviewers with write a

1 pending reviewer

Some checks were not successful
2 failing and 1 skipped checks

× CI / ontology_qc (pull_request) Failing after 3m

Want to do it yourself?

Check the official documentation in the OBOOK

- how-to: set up the ODK Docker environment
- tutorial: set up an ODK GitHub repository
- tutorial: a complete walk through the core ODK workflows
  - go here to learn more about what you can do with ODK
- reference: frequently used ODK commands

Thank You

For Your Attention