ToolDog Documentation

Release

Kenzo-Hugo Hillion, Ivan Kuzmin, Hervé Ménager

Mar 02, 2018

Contents

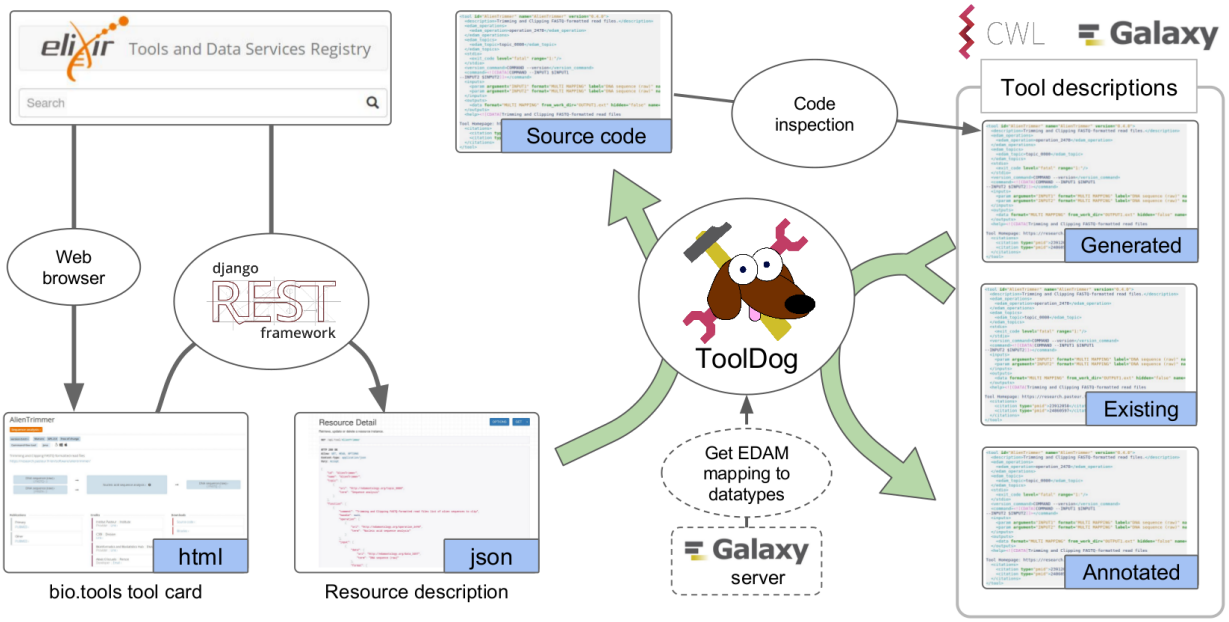
ToolDog is a program that generates tool description from a tool entry from https://bio.tools.

CHAPTER 1

ToolDog

1.1 Introduction

During the last years, integration of various tools has been eased by the use of workbench systems such as Galaxy or frameworks using Common Workflow Language. Still, it remains time consuming and not straightforward to adapt resources to such environments. **ToolDog** (Tool DescriptiOn Generator) is the main component of the Workbench Integration Enabler service of the ELIXIR bio.tools registry, that guides the integration of tools into workbench environments.



1.2 Installation

1.2.1 Requirements

ToolDog is built with Python 3.6.0 and uses the following Python libraries:

- galaxyxml (>=0.4.3)
- cwlgen (>=0.2.2)
- requests (>=2.13.0)
- rdflib (>=4.2.2)
- docker (==0.2.1)

Docker is also required in order to perform the code analysis part of the code.

Note: We highly recommend the use of a virtual environment with Python 3.6.0 using virtualenv or conda.

1.2.2 Installation procedure

Pip

You can use pip to install ToolDog of the latest stable version:

```
pip3 install tooldog
```

Manually

Note: This is particularly useful when you wish to install a version under development from any branches of the Github repository.

Clone the repository and install ToolDog with the following commands:

```
git clone https://github.com/bio-tools/ToolDog.git
cd ToolDog
pip3 install .
```

1.2.3 Uninstallation procedure

Pip

You can remove ToolDog with the following command:

```
pip3 uninstall tooldog
```

Note: This will not uninstall dependencies. To do so you can make use of the pip-autoremove tool pip-autoremove.

1.3 How to use ToolDog ?

ToolDog can either generates a template from a bio.tools entry for Galaxy or CWL but also annotates exiting tool descriptors with missing metadata. ToolDog supports generation of XML files for Galaxy (*-g/–galaxy*) or CWL tool (*-c/–cwl*). It works in two main steps: **(1)** Analysis of the source code when possible (for the moment, only support Python code using argparse), **(2)** Addition of metadata to the tool description.

Note: If you find a bug, have any questions or suggestions, please create an issue on GitHub or contact us on Gitter.

1.3.1 Import from https://bio.tools entry

You can generate a XML for Galaxy from an online https://bio.tools description from the identifier using the following command:

```
tooldog -g/--galaxy id > outfile.xml
```

1.3.2 Import from JSON local file

To generate XML from a local file downloaded from http://bio.tools, use the following command:

```
tooldog -g/--galaxy file.json > outfile.xml
```

1.3.3 Annotation of existing files

You can also use ToolDog to add missing metadata from your tool descriptor if the tool is registered on https://bio.tools:

```
tooldog -g/--galaxy id --existing_desc your_xml.xml > annotated_xml.xml
```

Note: For the moment, only annotation of Galaxy XML is supported.

1.3.4 Advanced options

Please refer to the help section (`tooldog -h`) of ToolDog to see the full list of options.

Run parts independently

You can decide to run both parts of ToolDog in an independant manner using `tooldog -g id --analyse` and `tooldog -g id --annotate` options.

Use your own settings and local files

ToolDog offers the possibility to use your own settings for most of the step of the generation.

Options for source code analysis

- `--source_language`: specify the language of your source code.
- `--source_code`: specify the path to your source code directory.

Warning: For the moment, only analysis of Python code is available

Options for tool description annotation

- `--inout_biotools`: select this option to also add inputs and outputs found on the https://bio.tools description to your Galaxy XML or CWL tool description.

Options specific to Galaxy XML generation

The options below are used for the mapping between EDAM formats and data to Galaxy datatypes. As some Galaxy instances sometimes have their own defined datatypes, you can here specify the url:

- `--galaxy_url`: URL of the Galaxy instance (default is https://usegalaxy.org)
- `--edam_url`: URL or local path to EDAM.owl (default is http://edamontology.org/EDAM.owl)
- `--mapping_file`: this is a JSON file generated by ToolDog that you can keep once you have performed your own mapping.

1.4 References

1.4.1 Articles

- Hillion KH, Kuzmin I, Khodak A et al. Using bio.tools to generate and annotate workbench tool descriptions [version 1; referees: 2 approved]. F1000Research 2017, 6(ELIXIR):2074 doi: 10.12688/f1000research.12974.1
- Hervé Ménager, Matúš Kalaš, Kristoffer Rapacki and Jon Ison. Using registries to integrate bioinformatics tools and services into workbench environments. International Journal on Software Tools for Technology Transfer (2016) doi: 10.1007/s10009-015-0392-z

1.4.2 Posters and presentations

- Hillion KH, Kuzmin I, Peterson H et al. ToolDog – generating tool descriptors from the ELIXIR tool registry [version 1; not peer reviewed]. F1000Research 2017, 6(ISCB Comm J):1194 (**slides**) doi: 10.7490/f1000research.1114473.1
- Hillion KH, Kuzmin I, Peterson H et al. ToolDog – generating tool descriptors from the ELIXIR tool registry [version 1; not peer reviewed]. F1000Research 2017, 6(ISCB Comm J):1193 (**poster**) doi: 10.7490/f1000research.1114472.1

1.4.3 External libraries repositories

- galaxyxml by Eric Rasche
- argparse2tool by Eric Rasche and Anton Khodak
- python-cwlgen by Hervé Ménager and Kenzo-Hugo Hillion

CHAPTER 2

ToolDog API Documentation

2.1 API documentation

2.1.1 main.py

Main functions used by ToolDog.

tooldog.main.**analyse**(*biotool*, *args*)
Run analysis of the source code from bio.tools or given locally.

Parameters

- **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object.
- **args** (argparse.ArgumentParser) – Parsed arguments.

tooldog.main.**annotate**(*biotool*, *args*, *existing_desc=None*)
Run annotation (generated by analysis or existing_desc).

Parameters

- **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object.
- **args** (argparse.ArgumentParser) – Parsed arguments.
- **existing_desc** (*STRING*) – Existing tool descriptor path.

tooldog.main.**config_logger**(*write_logs*, *log_level*, *log_file*, *verbose*)
Initialize the logger for ToolDog. By default, only WARNING, ERROR and CRITICAL are written on STDERR. You can also write logs to a log file.

Parameters

- **write_logs** (*BOOLEAN*) – Decide to write logs to output log file.
- **log_level** (*STRING*) – Select the level of logs. 'debug', 'info' or 'warn'. Other value is considered as 'warn'.
- **log_file** (*STRING*) – path to output log file.

Returns Config dictionnary for logger.

Return type DICT

tooldog.main.**json_from_biotools**(*tool_id*, *tool_version='latest'*)
Import JSON of a tool from https://bio.tools.

Parameters

- **tool_id** (*STRING*) – ID of the tool.
- **tool_version** (*STRING*) – Version of the tool.

Returns dictionnary corresponding to the JSON from https://bio.tools.

Return type DICT

tooldog.main.**json_from_file**(*json_file*)
Import JSON of a tool from a local JSON file.

Parameters **json_file** (*STRING*) – path to the file

Returns dictionnary corresponding to the JSON.

Return type DICT

tooldog.main.**json_to_biotool**(*json_file*)
Takes JSON file from bio.tools description and loads its content to `tooldog.model.Biotool` object.

Parameters **json_file** (*DICT*) – dictionnary of JSON file from bio.tools description.

Returns Biotool object.

Return type *tooldog.biotool_model.Biotool*

tooldog.main.**parse_arguments**()
Defines parser for ToolDog.

tooldog.main.**run**()
Running function called by tooldog command line.

tooldog.main.**write_cwl**(*biotool*, *outfile=None*, *existing_tool=None*)
This function uses `tooldog.cwl.CwlToolGen` to write CWL using cwlgen. CWL is generated on STDOUT by default.

Parameters

- **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object.
- **outfile** (*STRING*) – path to output file to write the CWL.
- **existing_tool** (*STRING*) – local path to existing CWL tool description.

tooldog.main.**write_xml**(*biotool*, *outfile=None*, *galaxy_url=None*, *edam_url=None*, *mapping_json=None*, *existing_tool=None*, *inout_biotool=False*)
This function uses `tooldog.galaxy.GalaxyToolGen` to write XML using galaxyxml library.

Parameters

- **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object.
- **outfile** (*STRING*) – path to output file to write the XML.
- **galaxy_url** (*STRING*) – link to galaxy instance.
- **edam_url** (*STRING*) – link to EDAM owl.
- **mapping_json** (*STRING*) – local JSON mapping between EDAM and Galaxy datatypes.
- **existing_tool** (*STRING*) – local path to existing Galaxy XML tool description.
- **inout_biotool** (*BOOLEAN*) – add input and outputs description from https://bio.tools.

2.1.2 biotool_model.py

Model used to process information contained in JSON from https://bio.tools description.

The content of a description on https://bio.tools is contained in a JSON file and this model aims to store the different information.

class tooldog.biotool_model.**Biotool**(*name*, *tool_id*, *version*, *description*, *homepage*)
This class correspond to an entry from https://bio.tools.

__init__(*name*, *tool_id*, *version*, *description*, *homepage*)

Parameters

- **name** (*STRING*) – Name of the tool.
- **tool_id** (*STRING*) – ID of the tool entry.
- **version** (*STRING*) – Version of the tool entry.
- **description** (*STRING*) – Description of the tool entry.
- **homepage** (*STRING*) – URL to homepage.

tooldog.biotool_model.Biotool object is also initialized with two empty list of objects:

- functions: list of *tooldog.biotool_model.Function*
- topics: list of *tooldog.biotool_model.Topic*

More information (*tooldog.biotool_model.Informations* object) can be specified using *tooldog.biotool_model.Biotool.set_informations()*.

add_functions(*functions*)
Add *tooldog.biotool_model.Function* objects to the list of functions of the Biotool object.

Parameters **functions** (*LIST of DICT*) – list of functions description from https://bio.tools.

add_topics(*topics*)
Add *tooldog.biotool_model.Topic* objects to the list of topics of the Biotool object.

Parameters **topics** (*LIST of DICT*) – list of topics description from https://bio.tools.

generate_cwl_doc()
Generate a doc from the different informations found on the tool.

Returns a doc for CWL tool description.

Return type STRING

generate_galaxy_help()
Generate a help message from the different informations found on the tool.

Returns a help message for Galaxy XML.

Return type STRING

set_informations(*tool_credits*, *contacts*, *publications*, *docs*, *language*, *links*, *download*)
Add an *tooldog.biotool_model.Informations* object to the Biotool.

Parameters

- **tool_credits** (*LIST of DICT*) – list of different tool_credits.
- **contacts** (*LIST of DICT*) – list of different contacts.
- **publications** (*LIST of DICT*) – list of different IDs for publications.
- **doc** (*LIST of DICT*) – list of different documentations.

class tooldog.biotool_model.**Contact**(*contact*)

Class to store one contact information.

__init__(*contact*)

Parameters **contact** (*DICT*) – contact part of the JSON from http://bio.tools.

class tooldog.biotool_model.**Credit**(*credit*)

Class to store a credit information.

__init__(*credit*)

Parameters **credit** (*DICT*) – credit part of the JSON from http://bio.tools.

class tooldog.biotool_model.**Data**(*data_type*, *formats*, *description=None*)

Data described by EDAM ontology.

__init__(*data_type*, *formats*, *description=None*)

Parameters

- **data_type** (*DICT*) – EDAM ontology for the data type with uri and term.
- **formats** (*LIST of DICT*) – EDAM ontology for data formats with uri and term.
- **description** (*STRING*) – description of the data (DEPRECATED)

class tooldog.biotool_model.**DataType**(*edam*)

EDAM data associated to either input or output.

__init__(*edam*)

Parameters **edam** (*DICT*) – EDAM ontology with uri and term.

class tooldog.biotool_model.**Documentation**(*documentation*)

Class to store one documentation information.

__init__(*documentation*)

Parameters **documentation** (*DICT*) – documentation part of the JSON from http://bio.tools.

class tooldog.biotool_model.**Edam**(*edam*)
Edam annotation with the uri and its corresponding term.

__init__(*edam*)

Parameters **edam** (*DICT*) – EDAM ontology with uri and term.

get_edam_id()
Get the EDAM id from the uri.

Returns EDAM id from the uri.

Return type STRING

class tooldog.biotool_model.**Format**(*edam*)
EDAM format associated to either input or output.

__init__(*edam*)

Parameters **edam** (*DICT*) – EDAM ontology with uri and term.

class tooldog.biotool_model.**Function**(*edams*)
Correspond to one function of the entry with the corresponding inputs and outputs.

__init__(*edams*)

Parameters **edams** (*LIST of DICT*) – EDAM ontology for operation(s) with uri and term.

tooldog.biotool_model.Function object is initialized with two empty list of objects:

- inputs: list of *tooldog.biotool_model.Input*
- outputs: list of *tooldog.biotool_model.Output*

add_inputs(*inputs*)
Add inputs to the *tooldog.biotool_model.Function* object.

Parameters **inputs** (*LIST of DICT*) – inputs part of one function from http://bio.tools.

add_outputs(*outputs*)
Add outputs to the *tooldog.biotool_model.Function* object.

Parameters **outputs** (*LIST of DICT*) – inputs part of one function from http://bio.tools.

class tooldog.biotool_model.**Informations**
Class to describe different information concerning a bio.tool entry.

__init__()
tooldog.biotool_model.Informations object is initialized with four empty list of objects:

- publications: list of *tooldog.biotool_model.Publication*
- documentations: list of *tooldog.biotool_model.Documentation*
- contacts: list of *tooldog.biotool_model.Contact*
- tool_credits: list of *tooldog.biotool_model.Credit*
- language: list of coding language
- link: list of *tooldog.biotool_model.Link*

class tooldog.biotool_model.**Input**(*data_type*, *formats*, *description=None*)
Input of a described function.

__init__(*data_type*, *formats*, *description=None*)

Parameters

- **data_type** (*DICT*) – EDAM ontology for the data type with uri and term.
- **formats** (*LIST of DICT*) – EDAM ontology for data formats with uri and term.
- **description** (*STRING*) – description of the data (DEPRECATED)

class tooldog.biotool_model.**Link**(*link*)
Class to store download and links content.

__init__(*link*)

Parameters **link** (*DICT*) – links or download content of the JSON from http://bio.tools.

class tooldog.biotool_model.**Operation**(*edam*)
EDAM operation associated to a function.

__init__(*edam*)

Parameters **edam** (*DICT*) – EDAM ontology with uri and term.

class tooldog.biotool_model.**Output**(*data_type*, *formats*, *description=None*)
Output of a described function.

__init__(*data_type*, *formats*, *description=None*)

Parameters

- **data_type** (*DICT*) – EDAM ontology for the data type with uri and term.
- **formats** (*LIST of DICT*) – EDAM ontology for data formats with uri and term.

- **description** (*STRING*) – description of the data (DEPRECATED)

class tooldog.biotool_model.**Publication**(*publication*)
Class to store one publication information.

__init__(*publication*)

Parameters **publication** (*DICT*) – publication part of the JSON from http://bio.tools.

class tooldog.biotool_model.**Topic**(*edam*)
EDAM topic associated to the entry.

__init__(*edam*)

Parameters **edam** (*DICT*) – EDAM ontology with uri and term.

2.1.3 analyse/code_collector.py

class tooldog.analyse.code_collector.**CodeCollector**(*biotool*)
Class to download source code from a https://bio.tools entry

__init__(*biotool*)

Parameters **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object

get_source()
Retrieve source code of the tool using links provided in https://bio.tools

2.1.4 analyse/container.py

Wrapper for docker-py low-level client API. Allow creating container and using it within *with* statement.

class tooldog.analyse.container.**Container**(*image*, *command*, *environment=None*)
Class to represent docker container and expose simple API to it.

__enter__()
Start the container and make a *with* context.

__exit__(*exc_type*, *exc_val*, *exc_tb*)
Stop the container and clean up.

__init__(*image*, *command*, *environment=None*)
Create the container. Not the same as *docker run*, need to be started after the creation.

Parameters

- **image** (*STRING*) – the image to run

- **command** (*STRING or LIST*) – the command to be run in the container
- **environment** (*DICT or LIST*) – A dictionary or a list of strings in the following format {"TEST": "123"} or ["TEST=123"].
- **mount** (*STRING*) – absolute path to file to mount

exec(*command*)
Execute the command inside container cid

Parameters **command** (*str*) – String representation of bash command

Returns Returns a generator of output of the result of running bash command in bytes

Return type iter

inspect()

Returns Description of the container ans its status

Return type DICT

kill()
Kill the container.

logs()

Returns Returns a stream of the STDOUT and STDERR

Return type iter

pull(*image*)
Pull image :param image: the image to pull :type image: STRING

remove()
Remove the container and associated volumes.

run(*image*, *command*, *environment=None*)
Do real work of __init__.

start()
Start the container.

stop()
Stop the container.

2.1.5 analyse/tool_analyzer.py

class tooldog.analyse.tool_analyzer.**ToolAnalyzer**(*biotool*, *gen_format*, *language=None*, *source_code=None*)

Class to perform appropriate source code analysis of a tool.

__init__(*biotool*, *gen_format*, *language=None*, *source_code=None*)

Parameters

- **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object
- **gen_format** (*STRING*) – tool descriptor language (Galaxy XML or CWL)
- **language** (*STRING*) – language of the tool
- **source_code** (*STRING*) – path to source code

get_source()

Get source code to give to analyzer.

run_analysis()

Method to run analysis of source code of the entry.

set_language()

Set the language attribute of the object based on the https://bio.tools description.

2.1.6 analyse/language_analyzer.py

class tooldog.analyse.language_analyzer.**LanguageAnalyzer**(*biotool*)

This should be the abstract class for all analyzer.

__init__(*biotool*)

Parameters **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object

analyse()

Run source code analysis.

class tooldog.analyse.language_analyzer.**PythonAnalyzer**(*gen_format*, *source_code*)

Object to specifically analyze Python source code.

__init__(*gen_format*, *source_code*)

Parameters

- **gen_format** (*STRING*) – tool description language (Galaxy XML or CWL)
- **source_code** (*STRING*) – path to source code.

analyse()
Run source code analysis.

2.1.7 annotate/galaxy.py

Generation of XML for Galaxy from https://bio.tools based on the Tooldog model using galaxyxml library.

class tooldog.annotate.galaxy.**GalaxyToolGen**(*biotool*, *galaxy_url=None*, *edam_url=None*, *mapping_json=None*, *existing_tool=None*)
Class to support generation of XML from *tooldog.biotool_model.Biotool* object.

__init__(*biotool*, *galaxy_url=None*, *edam_url=None*, *mapping_json=None*, *existing_tool=None*)
Initialize a [Tool] object from galaxyxml with the minimal information (a name, an id, a version, a description, the command, the command version and a help).

Parameters **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object of an entry from https://bio.tools.

add_citation(*publication*)
Add publication(s) to the tool (XML: <citations>).

Parameters **publication** (*tooldog.biotool_model.Publication*) – Publication object.

add_edam_operation(*operation*)
Add the EDAM operation to the tool (XML: <edam_operations>).

Parameters **topic** (*tooldog.biotool_model.Operation*) – Operation object.

add_edam_topic(*topic*)
Add the EDAM topic to the tool (XML: <edam_topics>).

Parameters **topic** (*tooldog.biotool_model.Topic*) – Topic object.

add_input_file(*input_obj*)
Add an input to the tool (XML: <inputs>).

Parameters **input_obj** (*tooldog.biotool_model.Input*) – Input object.

add_output_file(*output*)
Add an output to the tool (XML: <outputs>).

Parameters **output** (*tooldog.biotool_model.Output*) – Output object.

write_xml(*out_file=None*, *index=None*, *keep_old_command=False*)
Write CWL to STDOUT or out_file(s).

Parameters

- **out_file** (*STRING*) – path to output file.
- **index** (*INT*) – Index in case more than one function is described.

2.1.8 annotate/cwl.py

Generation of CWL tool from https://bio.tools based on the ToolDog model using cwlgen library.

class tooldog.annotate.cwl.**CwlToolGen**(*biotool*, *existing_tool=None*)
Class to support generation of CWL from *tooldog.biotool_model.Biotool* object.

__init__(*biotool*, *existing_tool=None*)
Initialize a [CommandLineTool] object from cwlgen.

Parameters **biotool** (*tooldog.biotool_model.Biotool*) – Biotool object of an entry from https://bio.tools.

add_edam_operation(*operation*)
Add the EDAM operation to the tool (CWL: s:operation).

Parameters **operation** (*tooldog.biotool_model.Operation*) – Operation object.

add_edam_topic(*topic*)
Add the EDAM topic to the tool (CWL: s:topic).

Parameters **topic** (*tooldog.biotool_model.Topic*) – Topic object.

add_input_file(*input_obj*)
Add an input to the CWL tool.

Parameters **input_obj** (*tooldog.biotool_model.Input*) – Input object.

add_output_file(*output*)
Add an output to the CWL tool.

Parameters **output** (*tooldog.biotool_model.Output*) – Output object.

add_publication(*publication*)
Add publication to the tool (CWL: s:publication).

Parameters **publication** (*tooldog.biotool_model.Publication*) – Publication object.

write_cwl(*out_file=None*, *index=None*)
Write CWL to STDOUT or out_file(s).

Parameters

- **out_file** (*STRING*) – path to output file.
- **index** (*INT*) – Index in case more than one function is described.

2.1.9 annotate/edam_to_galaxy.py

Gather different information from a Galaxy server (by default https://usegalaxy.org) and EDAM ontology (by default from http://edamontology.org/EDAM.owl)

class tooldog.annotate.edam_to_galaxy.**EdamInfo**(*edam_url*)
Contains the given EDAM ontology.

It is also possible to generate several dictionnaries to help interrogating the ontology for a faster access.

__init__(*edam_url*)

Parameters **edam_url** (*STRING*) – path to EDAM.owl file

All the EDAM ontology will be contained in a dictionnary (self.edam_ontology).

generate_hierarchy()
Generates two dictionnaries of the EDAM hierarchy (format and data) with the following structure:

DICT[edam_uri] -> LIST of edam_uri from parents

The dictionnary can be accessed via self.edam_format_hierarchy

class tooldog.annotate.edam_to_galaxy.**EdamToGalaxy**(*galaxy_url=None*, *edam_url=None*, *mapping_json=None*)
Class to make the link between EDAM ontology terms (edam_format and edam_data) and Galaxy datatypes.

__init__(*galaxy_url=None*, *edam_url=None*, *mapping_json=None*)

Parameters

- **galaxy_url** (*STRING*) – URL of the galaxy instance.
- **edam_url** (*STRING*) – path to EDAM.owl file (URL or local path).
- **mapping_json** (*STRING*) – path to personnalized EDAM mapping to Galaxy.

export_info(*export_file*)

Method to export mapping of this object to a JSON file.

Parameters **export_file** (*STRING*) – path to the file.

generate_mapping()

Generates mapping between edam_format and edam_data to Galaxy datatypes based on the information of the Galaxy instance and the EDAM ontology.

Every edam_format and edam_data will be given a datatype.

get_datatype(*edam_data=None*, *edam_format=None*)

Get datatype from EDAM terms. :param edam_data: EDAM data term. :type edam_data: STRING :param edam_format: EDAM format term. :type edam_format: STRING :return: datatype corresponding to given EDAM ontologies. :rtype: STRING

load_local_mapping(*local_file*)

Method to load (from JSON file) mapping previously generated and exported in the *local_file*.

Parameters **local_file** (*STRING*) – path to the mapping local file.

class tooldog.annotate.edam_to_galaxy.**GalaxyInfo**(*galaxy_url*)

Class to gather different information about a Galaxy instance.

By default, if the galaxy_url is None, information is loaded from local files located in the *data/* folder corresponding to https://usegalaxy.org.

__init__(*galaxy_url*)

Parameters **galaxy_url** (*STRING*) – URL of the Galaxy instance.

tooldog.edam_to_galaxy.GalaxyInfo object is initialized with several information from the given Galaxy instance. It contains:

Parameters

- **self.version** (*STRING*) – version of the Galaxy instance.
- **self.edam_formats** (*DICT*) – mapping edam_format to LIST of extension of datatypes.
- **self.edam_data** (*DICT*) – mapping edam_data to LIST of extension of datatypes.

- **self.hierarchy** – class_to_classes part of the /api/mapping.json which maps

the parental classes of each classes. :type self.hierarchy: DICT :param self.class_names: ext_to_class_name part of the /api/mapping.json which maps the extension of a datatype to its class in Galaxy. :type self.class_names: DICT

select_root(*datatypes*)

Select the root datatype from all given datatypes.

Parameters **datatypes** (*list of STRING*) – list of different datatypes.

Returns root datatype.

Return type STRING

CHAPTER 3

Hangouts and Changelogs

3.1 Hangouts

3.1.1 23 May 2017

Ivan Kuzmin and Kenzo-Hugo Hillion

Discussion about the following points:

- Deliverable by the end of June
  - Link analysis part into ToolDog
  - Identify good example from https://bio.tools for demo
- Discussion about possible evolution of Tooldog and the library it uses
  - Evolution of the library galaxyxml and cwlgen
  - Build a similar model for bio.tools entries

3.1.2 24–28 April 2017, Paris

The meeting was to set up the collaboration between ELIXIR France (Hervé Menager, Kenzo-Hugo Hillion) and ELIXIR Estonia (Hedi Peterson, Ivan Kuzmin) nodes on the development of the workbench integration enabler.

Currently the tool generates Galaxy XML or CWL directly from the bio.tools tool description file in JSON as shown in the following figure.

After discussing the design of the tool an idea for a new architecture has emerged. ToolDog will not simply be monodirectional, but instead would allow to go from any given tool descriptor to another one as illustrated in the next figure.

Therefore, work is going to be first focused on both galaxyxml and cwlgen libraries to cover all different fields from corresponding tool descriptors. Then this libraries need to allow accurate import of existing files into the corresponding model. After that the new model for ToolDog can be built.

3.2 Roadmap

A brief summary of planned development for ToolDog.

3.2.1 2017 Q2

- Create environment and run argparse2tool to analyse python tool using argparse and annotate its output with metadata from bio.tools.
- Import and annotate Galaxy XML and CWL tools.

3.2.2 2017 Q4

- Change the model of ToolDog to have its own model :
  - model that can be extracted in JSON for bio.tools, Galaxy XML or CWL Tool.
  - build appropriate translator/generator objects.

3.3 Changelogs

Summary of developments of ToolDog software.

3.3.1 v0.3

v0.3.1

- DOI are not fetched when only PMID or PMCID is given on bio.tools through this API

- Addition of `--inout_biotools` to also write inputs and outputs from https://bio.tools in the tool description
- Namespaces have been added to cwlgen library so more information can be written in the CWL tool description
- Better errors and warnings handling for code analysis part
- ToolDog is not asking for `id/version` anymore but only `id` instead

v0.3.0

- Addition of source code analysis feature:
  - use argparse2tool in a docker container
  - only cover python tools using argparse
- Both part of ToolDog can be run independently:
  - *tooldog –analyse tool_id/version*
  - *tooldog –annotate tool_id/version*
- Options are available to specify language of the tool manually, as well as a path to access source code locally

3.3.2 v0.2

v0.2.2

- Add import feature from cwlgen to the workflow

v0.2.1

- Modify architecture of ToolDog
- add *–analyse* (feature not available yet) and *–annotate* arguments

v0.2.0

This is the first release of Tooldog:

- Import bio.tools description from online or local JSON file
- Generation of Galaxy XML:
  - Generates skeleton from bio.tools description (metadata)

  - Possibility to add EDAM annotation and citations to existing Galaxy XML
- Generation CWL tool:
  - Generates skeleton from bio.tools description (metadata)

Python Module Index

t

Index

Symbols

A

B

C

D

E

F

G

I

J

K

L

O

P

R

S

T

W