INSPIRE-HEP Documentation

Release v20181214

INSPIRE-HEP Collaboration

Dec 14, 2018

Contents

CHAPTER 1

About

INSPIRE is the leading information platform for High Energy Physics (HEP) literature. It provides users with high quality, curated metadata covering the entire corpus of HEP and the fulltext of all such articles that are Open Access.

This repository contains the source code of the next version of INSPIRE, which is currently under development, but already available at https://labs.inspirehep.net. It is based on version 3 of the Invenio Digital Library Framework.

A preliminary version of the documentation is available on Read the Docs.

CHAPTER 2

Contents

2.1 Getting Started

2.1.1 About INSPIRE

About

Inspire is a set of services, the main one being a search engine for high energy physics papers, with some side services like authors profiles, conferences, journals, institutions, experiments and a small specialized job market. It's main purpose is to provide physicists worldwide with a source of information about high energy physics related topics.

Currently we have two main websites open to the public:

- The Legacy website, with the current production application.
- The QA website, running the latest inspire-next code (for test purposes only).

2.1.2 Installing INSPIRE

Docker (Linux)

Docker is an application that makes it simple and easy to run processes in a container, which are like virtual machines, but more resource-friendly. For a detailed introduction to the different components of a Docker container, you can follow this tutorial.

Inspire and Docker

Get the latest Docker appropriate to your operationg system, by visiting Docker's official web site and accessing the *Get Docker* section.

Note: If you are using Mac, please build a simple box with `docker-engine` above `1.10` and `docker-compose` above `1.6.0`.

Make sure you can run docker without `sudo`.

- `id $USER`

  If you are not in the `docker` group, run the following command and then restart `docker`. If this doesn't work, just restart your machine :)
- `newgrp docker` or `su - $USER`
- `sudo usermod -a -G docker $USER`

Get the latest docker-compose:

```
$ sudo pip install docker-compose
```

- Add `DOCKER_DATA` env variable in your `.bashrc` or `.zshrc`. In this directory you will have all the persistent data between Docker runs.

```
$ export DOCKER_DATA=~/inspirehep_docker_data/
$ mkdir -p "$DOCKER_DATA"
```

By default the virtualenv and everything else will be kept on `/tmp` and they will be available only until the next reboot.

- Install a host persistent venv and build assets

Note: From now on all the docker-compose commands must be run at the root of the inspire-next repository, you can get a local copy with:

```
$ git clone git://github.com/inspirehep/inspire-next
$ cd inspire-next
```

```
$ docker-compose pull
$ docker-compose -f docker-compose.deps.yml run --rm pip
```

Note: If you have trouble with internet connection inside docker probably you are facing known DNS issue. Please follow this solution with DNS: `--dns 137.138.17.5 --dns 137.138.16.5`.

```
$ docker-compose -f docker-compose.deps.yml run --rm assets
```

- Run the service locally

```
$ docker-compose up
```

- Populate database

```
$ docker-compose run --rm web scripts/recreate_records
```

Once you have the database populated with the tables and demo records, you can go to localhost:5000

- Run tests in an **isolated** environment.

Note: The tests use a different set of containers than the default `docker-compose up`, so if you run both at the same time you might start having ram/load issues, if so, you can stop all the containers started by `docker-compose up` with `docker-compose kill -f`

You can choose one of the following tests types:

- unit
- workflows
- integration
- acceptance-authors
- acceptance-literature

```
$ docker-compose -f docker-compose.test.yml run --rm <tests type>
$ docker-compose -f docker-compose.test.yml down
```

Tip:

- cleanup all the containers:

  `docker rm $(docker ps -qa)`
- cleanup all the images:

  `docker rmi $(docker images -q)`
- cleanup the virtualenv (careful, if docker_data is set to something you care about, it will be removed):

  `sudo rm -rf "${DOCKER_DATA?DOCKER_DATA was not set, ignoring}"`

Extra useful tips

- Run a random shell

```
$ docker-compose run --rm web inspirehep shell
```

- Run *virtualenv* bash shell for running scripts manually (e.g. recreating records or building documentation)

```
$ docker-compose run --rm web bash
```

- Reload code in a worker

```
$ docker-compose restart worker
```

- Quick and safe reindex

```
$ docker-compose restart worker && docker-compose run --rm web scripts/recreate_
↪records
```

- Recreate all static assets. Will download all dependencies from npm and copy all static files to `${DOCKER_DATA}/tmp/virtualenv/var/inspirehep-instance/static`.

```
$ docker-compose -f docker-compose.deps.yml run --rm assets
```

- Monitor the output from all the services (elasticsearch, web, celery workers, database, flower, rabbitmq, scrapyd, redis) via the following command:

```
$ docker-compose up
```

Native Install (CentOS - MacOS)

System prerequisites

This guide expects you to have installed in your system the following tools:

- git
- virtualenv
- virtualenvwrapper
- npm > 3.0
- postgresql + devel headers
- libxml2 + devel headers
- libxslt + devel headers
- ImageMagick
- redis
- elasticsearch

CentOS

```
$ sudo yum install python-virtualenv python-virtualenvwrapper \
    npm postgresql postgresql-devel libxml2-devel ImageMagick redis git \
    libxslt-devel
$ sudo npm -g install npm
```

For elasticsearch you can find the installation instructions on the elasticsearch install page, and, to run the development environment, you will need also to add the following workarounds:

```
$ sudo usermod -a -G $USER elasticsearch
$ newgrp elasticsearch  # or log out and in again
$ sudo ln -s /etc/elasticsearch /usr/share/elasticsearch/config
```

MacOS

```
$ brew install postgresql
$ brew install libxml2
$ brew install libxslt
$ brew install redis
$ brew cask install caskroom/versions/java8
$ brew install elasticsearch@2.4
$ brew install rabbitmq
$ brew install imagemagick@6
$ brew install libmagic
```

```
$ brew install ghostscript
$ brew install poppler
```

You might also need to link imagemagick:

```
$ brew link --force imagemagick@6
```

Add to ~/.bash_profile:

```
# ElasticSearch.
export PATH="/usr/local/opt/elasticsearch@2.4/bin:$PATH"
```

Create a virtual environment

Create a virtual environment and clone the INSPIRE source code using *git*:

```
$ mkvirtualenv --python=python2.7 inspirehep
$ workon inspirehep
(inspirehep)$ cdvirtualenv
(inspirehep)$ mkdir src
(inspirehep)$ git clone https://github.com/inspirehep/inspire-next.git src/inspirehep
```

Note: It is also possible (and more flexible) to do the above the other way around like this and clone the project into a folder of your choice:

```
$ git clone https://github.com/inspirehep/inspire-next.git inspirehep
$ cd inspirehep
$ mkvirtualenv --python=python2.7 inspirehep
$ workon inspirehep
```

This approach enables you to switch to a new virtual environment without having to clone the project again. You simply specify on which environment you want to `workon` using its name.

Just be careful to replace all `cdvirtualenv src/inspirehep` in the following with a `cd path_you_chose/inspirehep`.

Install requirements

Use *pip* to install all requirements, it's recommended to upgrade pip and setuptools to latest too:

```
(inspirehep)$ pip install --upgrade pip setuptools
(inspirehep)$ cdvirtualenv src/inspirehep
(inspirehep)$ pip install -r requirements.txt --pre --exists-action i
(inspirehep)$ pip install honcho
```

And for development:

```
(inspirehep)$ pip install -e .[development]
```

Custom configuration and debug mode

If you want to change the database url, or enable the debug mode for troubleshooting, you can do so in the *inspirehep.cfg* file under *var/inspirehep-instance*, you might need to create it:

```
(inspirehep)$ cdvirtualenv var/inspirehep-instance
(inspirehep)$ vim inspirehep.cfg
```

There you can change the value of any of the variables that are set under the file *src/inspirehep/inspirehep/config.py*, for example:

```
DEBUG = True
SQLALCHEMY_DATABASE_URI = "postgresql+psycopg2://someuser:somepass@my.postgres.
↪server:5432/inspirehep"
```

Note: Make sure that the configuration keys you override here have the same exact name as the ones in the config.py file, as it will not complain if you put a key that did not exist.

Build assets

We build assets using *npm*. Make sure you have installed it system wide.

```
(inspirehep)$ sudo npm update
(inspirehep)$ sudo npm install -g node-sass@3.8.0 clean-css@^3.4.24 requirejs uglify-
↪js
```

Note: If you don't want to use sudo to install the npm packages globally, you can still setup a per-user npm modules installation that will allow you to install/remove modules as normal user. You can find more info in the npm docs here.

In particular, if you want to install the `npm` packages directly in your `virtualenv`, just add `NPM_CONFIG_PREFIX=$VIRTUAL_ENV` in the `postactivate` file of your `virtualenv` folder and you will be able to run the above command from inside your virtual environment.

Then we build the INSPIRE assets:

```
(inspirehep)$ inspirehep npm
(inspirehep)$ cdvirtualenv var/inspirehep-instance/static
(inspirehep)$ npm install
(inspirehep)$ inspirehep collect -v
(inspirehep)$ inspirehep assets build
```

Note: Alternatively, run *sh scripts/clean_assets* to do the above in one command.

Create database

We will use *postgreSQL* as database. Make sure you have installed it system wide.

Then create the database and database tables if you haven't already done so:

```
(inspirehep)$ psql
# CREATE USER inspirehep WITH PASSWORD 'dbpass123';
# CREATE DATABASE inspirehep;
# GRANT ALL PRIVILEGES ON DATABASE inspirehep to inspirehep;
(inspirehep)$ inspirehep db init
(inspirehep)$ inspirehep db create
```

Start all services

Rabbitmq

You must have rabbitmq installed and running (and reachable) somewhere. To run it locally on a CentOS:

```
$ sudo yum install rabbitmq-server
$ sudo service rabbitmq-server start
$ sudo systemctl enable rabbitmq-server.service  # to start on system boot
```

Everything else: Honcho

We use honcho to manage our services and run the development server. See Procfile for details.

```
(inspirehep)$ cdvirtualenv src/inspirehep
(inspirehep)$ honcho start
```

In MacOS you still need to manually run rabbitmq and postgresql:

```
$ brew services start rabbitmq
$ brew services start postgresql
```

And the site is now available on http://localhost:5000.

Create ElasticSearch Indices and Aliases

Note: Remember that you'll need to have the elasticsearch bin directory in your $PATH or prepend the binaries executed with the path to the elasticsearch bin directory in your system.

First of all, we will need to install the *analysis-icu* elasticsearch plugin.

```
(inspirehep)$ plugin install analysis-icu
```

For MacOS the *plugin* command will probably not be available system wide, so:

```
$ /usr/local/Cellar/elasticsearch\@2.4/2.4.6/libexec/bin/plugin install analysis-icu
```

Now we are ready to create the indexes:

```
(inspirehep)$ inspirehep index init
```

If you are having troubles creating your indices, e.g. due to index name changes or existing legacy indices, try:

```
(inspirehep)$ inspirehep index destroy --force --yes-i-know
(inspirehep)$ inspirehep index init
```

Create admin user

Now you can create a sample admin user, for that we will use the fixtures:

```
(inspirehep)$ inspirehep fixtures init
```

Note: If you are not running in debug mode, remember to add the *local=1* HTTP GET parameter to the login url so it will show you the login form, for example:

> http://localhost:5000/login/?local=1

Add demo records

```
(inspirehep)$ cdvirtualenv src/inspirehep
(inspirehep)$ inspirehep migrate file --force --wait inspirehep/demosite/data/demo-
↪records.xml.gz
```

Note: Alternatively, run *sh scripts/recreate_records* to drop db/index/records and re-create them in one command, it will also create the admin user.

Warning: Remember to keep *honcho* running in a separate window.

Create regular user

Now you can create regular users (optional) with the command:

```
(inspirehep)$ inspirehep users create your@email.com -a
```

Access the records (web/rest)

While running *honcho* you can access the records at

```
$ firefox http://localhost:5000/literature/1
$ curl -i -H "Accept: application/json" http://localhost:5000/api/records/1
```

2.2 Developers Guide

2.2.1 Basic development flow

Git configuration

First of all we have to set up some basic git configuration values:

- Set up the user info that will be used by Git as author and committer for each commit.

```
git config --global user.name "name surname"
git config --global user.email "your@email.here"
```

- Configure git to add the *Signed-off-by* header on each commit:

```
git config --global format.signoff true
```

Recommended: configure your ssh key on GitHub

That will allow you to easily access the git repositories without having to enter your user and password every time in a secure manner.

If you don't have one already, create an ssh key:

```
ssh-keygen
```

It will ask for a path and a password, the password is optional.

Now go to the github settings page for keys and add the contents of the public key you just created, by default *~/.ssh/id_rsa.pub*.

Warning: Never share you private key with anybody! (by default ~/.ssh/id_rsa)

Recomended: install the hub tool for git-github integration

There's a tool created by github that adds some extra commands and better integration with github to the git command, you can download it from the hub tool git repo.

Throughout this guide you will see also some tips that use it.

Clone the code

Navigate to your work directory (or wherever you want to put the code) and clone the main repository from github:

```
cd ~/Work  # or wherever you want to store the repo
git clone git@github.com:inspirehep/inspire-next
cd inspire-next
```

You will need also to add your personal fork, to do so just:

```
git remote add <your_gh_user> git@github.com:<your_gh_user>/inspire-next
```

Replacing <*your_gh_user*> with your github username.

Now to make sure you have the correct remotes set up, you can run:

```
git remote -v
```

And that should show two, one called *origin* that points to the inspirehep repo, and one called <*your_gh_user*> that points to your fork.

If for any reason you messed up or want to change the url or add/remove a remote, check the commands:

```
git remote add <name> <url>
git remote remove <name>
git remote set-url <url>
```

Note: If you are using the hub tool, you can clone the inspire repo, fork it and setup the remotes with:

```
hub clone inspirehep/inspire-next
cd inspire-next
hub fork
```

Create your feature branch

Before starting to make changes, you should create a branch for them:

```
git checkout -b add_feature_x
```

It's a good habit to name your feature branch in a way that hints about what it is adding/fixing/removing, for example, instead of *my_changes* it's way better to have *adds_user_auth_to_workflows*.

Do your changes

Now you can start modifying, addin or removing files, try to create commits regularily, and avoid mixing up changes on the same commit. For example, commit any linting changes to existing code in a different commit to the addition of code, or the addition of the tests.

To commit the changes:

```
git add <modified_file>
git rm <file_to_delete>
git add <any_new_file>
git commit
```

About the commit message structure, we try to follow the Invenio commit guideline, but we put a strong emphasis in the content, specially:

- Describe why you did the change, not what the change is (the diff already shows the what).
- In the message body, add as many information as you need, it's better to be extra verbose than the alternative.
- If it adresses an issue, add the coment *closes #1234* to the description, where *#1234* is the issue number on github.

Create a pull request

As soon as you have worked some time doing changes, it's recommended to share them, even if they are not ready yet, so in case that there's a misundestanding on how to do the change, you don't find out after spending a lot of time on it.

To create the pull request, first you have to push your changes to your repositoy:

```
git push <your_gh_user> <add_feature_x> -f
```

Note: The *-f* flag is required if it's not the first time you push, and you rebased your changes in between.

Now you can go to your github repo page, and create a new pull request, that will ask you to specify a new message and description for it, if you had multiple commits, try to summarize them there, that will help with the review.

Note: If you are using the hub tool, you can create a pull request with: .. code-block:: console

hub pull-request

Warning: At this point, travis will test your changes and give you some feedback on github. To avoid ping-ponging with travis and save you some time, it's highly recommended to run the tests locally first, that will also allow you to debug any issues.

By default, your pull request will start with the flag *WIP*, while this is set, you can push to it as many times as you want. Once your changes are ready to be reviewed, add the *Need: Review* flag and remove the *WIP*. It's also recommended to request a review directly to someone if you know that she's good in the domain of the pull request.

Update your changes

Some pull requests might take some time to merge, and other changes get merged before to master. That might generate some code conflicts or make your tests fail (or force you to change some of your code).

To resolve that issue, you should rebase on the latest master branch periodically (try to do it at the very least once a day).

To do so: * Fetch changes from the remotes:

```
git fetch --all
```

- Rebase your code and edit, drop, squash, cherry-pick and/or reword commits. This step will force you to resolve any conflicts that might arise.

```
git rebase -i origin/master
```

- Run the tests again to make sure nothing got broken.

Documentation

Same as tests, documentation is part of the development process, so whenever you write code, you should keep this priorities in mind:

- Very readable code is best.

- Good comments is good.
- Extra documentation is ok.

Documentation will be required though for some parts of the code meant to be reused several times, like apis, utility functions, etc.

The format of the docstrings that we use is the Google style one defined in the Napoleon Sphinx extension page.

More details

Some useful links are listed bellow:

Official git documentation

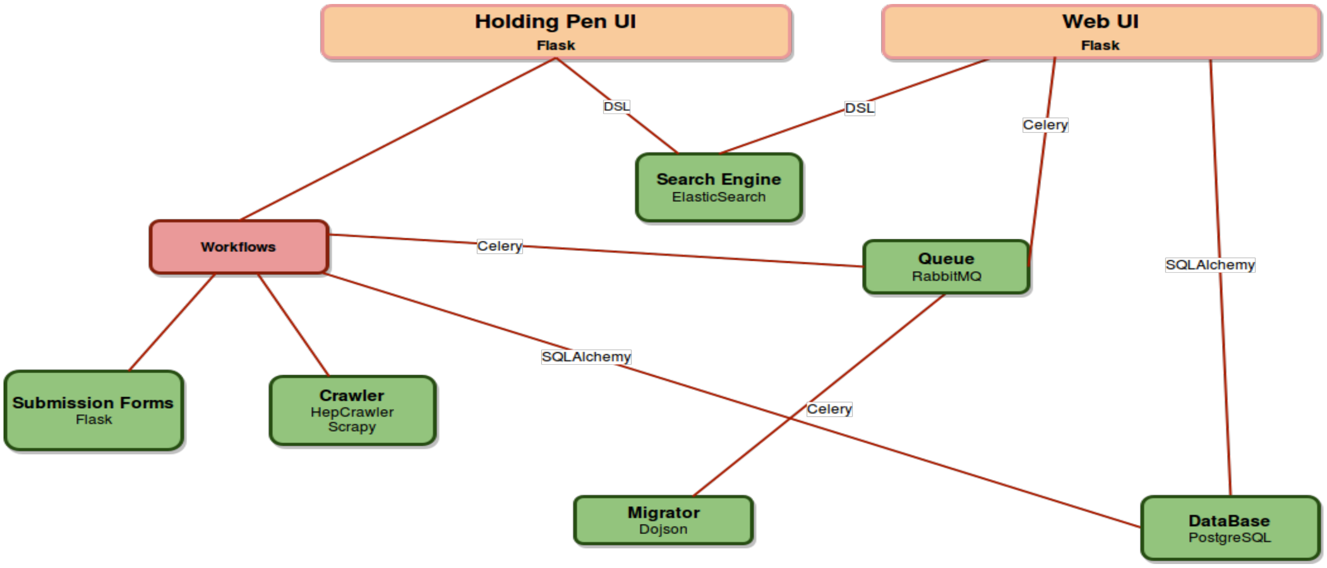
Git branching tutorial

General git tutorial

2.2.2 Technologies Used

High level overview

This is a very basic framework overview.



Invenio

INVENIO

Invenio is a free software suite enabling you to run your own digital library or document repository on the web. The technology offered by the software covers all aspects of digital library management, from document ingestion through classification, indexing, and curation up to document dissemination. Invenio complies with standards such as the Open Archives Initiative and uses MARC 21 as its underlying bibliographic format. The flexibility and performance of Invenio make it a comprehensive solution for management of document repositories of moderate to large sizes.

Invenio has been originally developed at CERN to run the CERN document server, managing over 1,000,000 bibliographic records in high-energy physics since 2002, covering articles, books, journals, photos, videos, and more. Invenio is nowadays co-developed by an international collaboration comprising institutes such as CERN, DESY, EPFL, FNAL, SLAC and is being used by many more scientific institutions worldwide.

INSPIRE is build on top of latest Invenio currently version is 3.0.

For a detailed description of how we use the different Invenio modules, see the Invenio modules section.

Flask

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions.

Official documentation for Flask.

Related tutorial.

Werkzeug

Werkzeug is a WSGI utility library for Python.

Official documentation for Werkzeug.

Jinja

Jinja2 is a modern and designer-friendly templating language for Python, modelled after Django's templates.

Official documentation for Jinja.

SQLAlchemy

SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL. It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.

Official documentation for SQLAlchemy.

Celery

Celery is a simple, flexible and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system. It's a task queue with focus on real-time processing, while also supporting task scheduling.

Official documentation for Celery.

ElasticSearch

Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.

In addition, Elasticsearch provides a full Query DSL based on JSON to define queries and it's used by INSPIRE.

Official documentation for ElasticSearch.

DSL documentation for ElasticSearch.

Angular js

(under construction)

2.2.3 Invenio modules

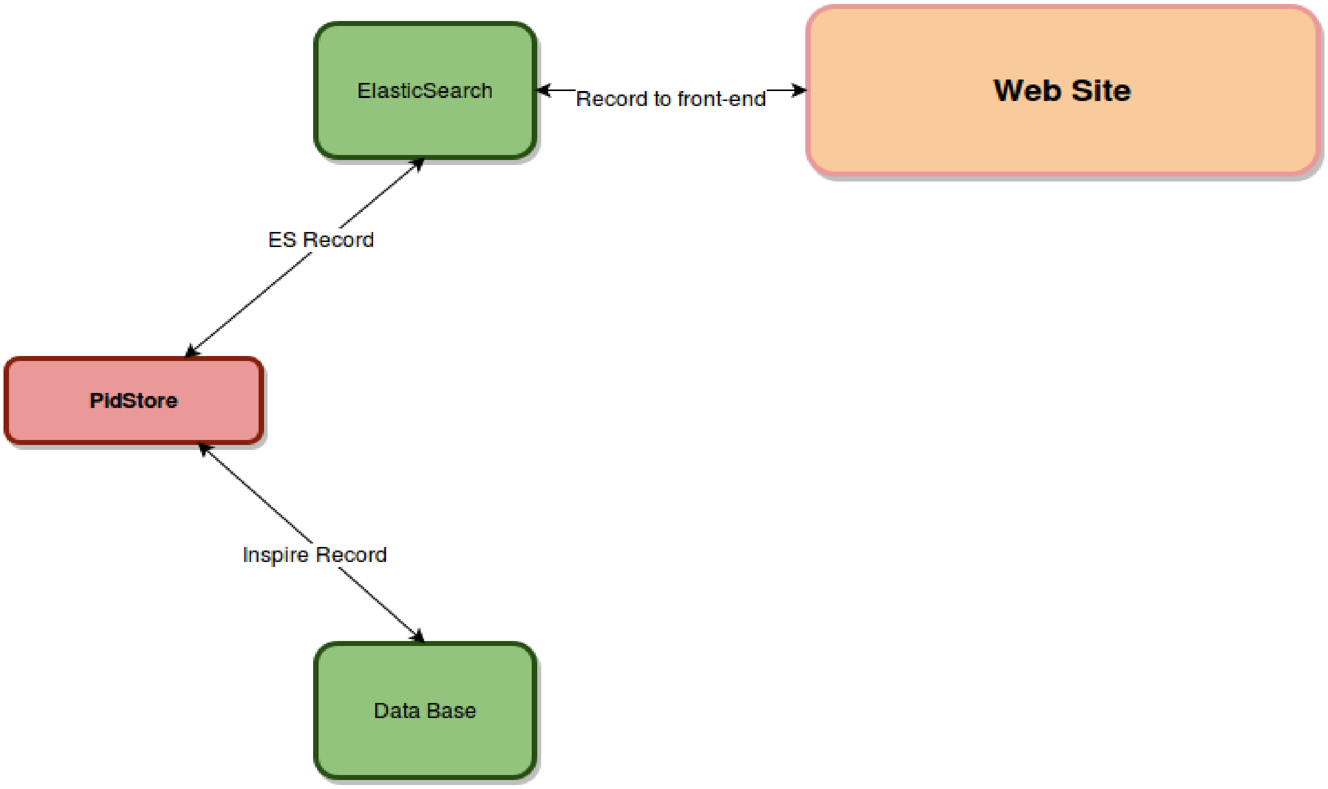
Invenio-PIDStore

Pidstore in Inspire-next

Pidstore is based on on the Invenio pidstore module that mints, stores, registers and resolves persistent identifiers. Pidstore has several uses in Inspire-next:

- Map record ids (UUIDs) between ElasticSearch and DataBase. In that way every record, that is stored in the Database, can be fetched and imported by ElasticSearch. Also, it's important to notice that the records for the front-end are inherited by `ES Record`, so they are coming from ElasticSearch.
- Pidstore also provide a unique identifier for every record that is the *known* id for the outer world.

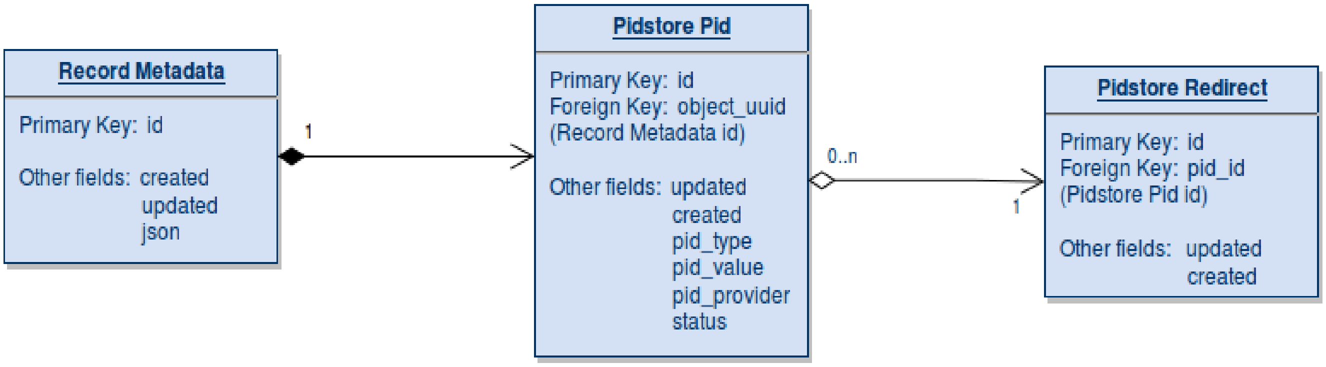
In the following you can find how pidstore is connected with Inspire-next.



Pidstore and Database

There are three basic tables:

- Record Metadata: is the table of the actual record stored in the database. The primary key (id) is foreign key (object_uuid) for the table `Pidstore Pid`. In that way record is mapped to the pidstore.
- Pidstore Pid: is the main table of pidstore in which are stored all the *known* ids called `pid_value` for the outer world. For example given url of a specific record `https://server_name.cern.ch/literature/1482866` , number `1482866` is the `pid_value` stored in `Pidstore Pid` table.
- Pidstore Redirect: is the table of pidstore that keeps the mapping of a record that is redirected to another record.



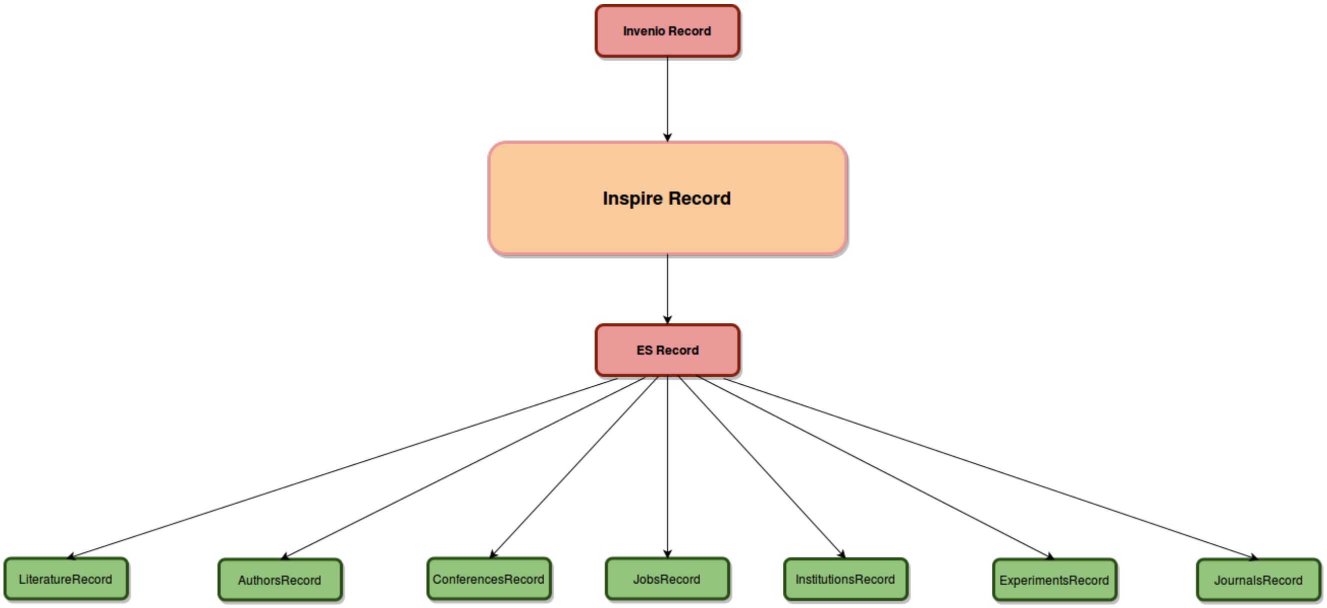
Invenio-Records

Inspire Record Class

- A record is the unit of information that we manage in inspire, from a literature record to a job record.
- This data is stored as a json object that must be compliant on a specific jsonschema.

The Inpire record is derived by the base class of Invenio record. Inspire record is used mainly for the back-end processes and for the outer world is used the inherited classes of Inspire record. According to the bellow diagram, Inspire record is the base class and ES record (ElasticSearch) is the derived class. The data that are given to the front-end are inherited classes from ES record:

- AuthorsRecord
- LiteratureRecord
- JobsRecord
- ConferencesRecord
- InstitutionsRecord
- ExperimentsRecord
- JournalsRecord



Note: the above classes are written in the following files.

- `inspirehep/modules/records/wrappers.py`
- `inspirehep/modules/records/api.py`

Invenio-Query-Parser

(under construction)

Invenio-Search

(under construction)

2.2.4 Ingestion of records (Workflows)

Inspire-next retrieves new records every day from several sources, such as:

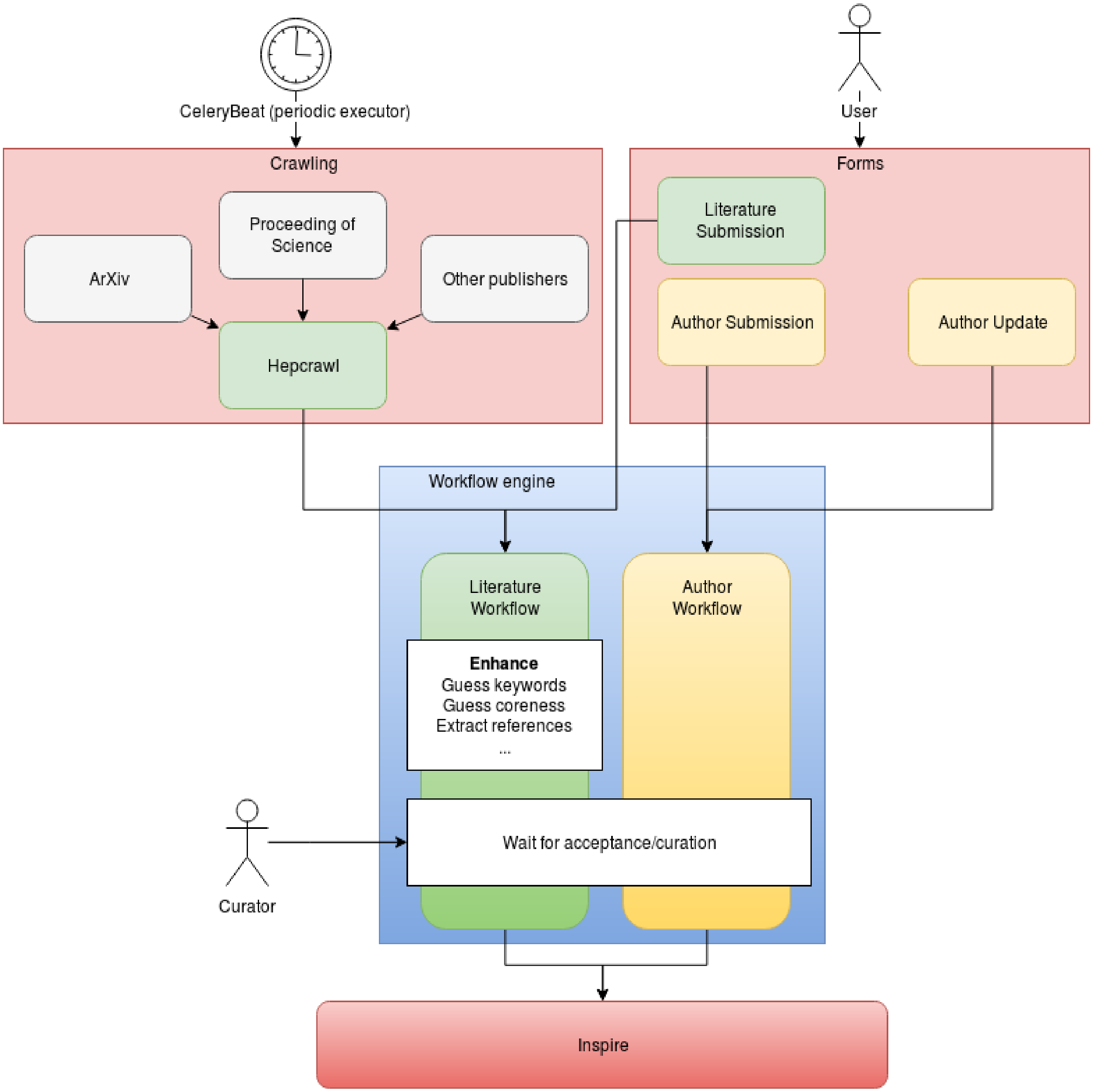
- External sites (arXiv, Proceedings of Science, ...).
- Users, through submission forms.

The records harvested from external sites are all pulled in by hepcrawl, that is periodically executed by a celery beat task.

The Users also suggest new records, both literature records and author records by using the submission forms.

One of the main goals of Inspire is the high quality of the information it provides, so in order to achieve that, every record is carefully and rigorously revised by our team of curators befor finally getting accepted inside the Inspire database.

Below there's a small diagram summarizing the process.



2.2.5 Handle workflows in error state

Via web interface

1. Visit Holding Pen list and filter for records in error state.

2. If any, you need to investigate why the record workflow failed, check the detailed page error report.
3. Sometimes the fix is simply to restart the task again if it is due to some circumstantial reasons.

   You can do that from the interface by clicking the "current task" button and hit restart.

Via shell

1. SSH into any worker machine (usually builder to avoid affecting the machines serving users)
2. Enter the shell and retrieve all records in error state:

```
inspirehep shell
```

```
from invenio_workflows import workflow_object_class, ObjectStatus
errors = workflow_object_class.query(status=ObjectStatus.ERROR)
```

3. Get a specific object:

```
from invenio_workflows import workflow_object_class
obj = workflow_object_class.get(1234)
obj.data  #  Check data
obj.extra_data   # Check extra data
obj.status  # Check status
obj.callback_pos  # Position in current workflow
```

4. See associated workflow definition:

```
from invenio_workflows import workflows
workflows[obj.workflow.name].workflow   # Associated workflow list of tasks
```

5. Manipulate position in the workflow

```
obj.callback_pos = [1, 2, 3]
obj.save()
# to persist the change in the db
from invenio_db import db
db.session.commit()
```

6. Restart workflow in various positions:

```
obj.restart_current()  # Restart from current task and continue workflow
obj.restart_next()  # Skip current task and continue workflow
obj.restart_previous()  # Redo task before current one and continue workflow

# If the workflow is in inital state, you can start it from scratch
from invenio_workflows import start
start('article', object_id=obj.id)
# or for an author workflow
start('author', object_id=obj.id)
```

2.2.6 Common Tasks

Caching

For caching we use Invenio-Cache. For example, to set a value in the cache:

```
>>> from invenio_cache import current_cache
>>> current_cache.set('test', [1, 2, 3], timeout=60)
```

And to retrieve the value from the cache:

```
>>> from invenio_cache import current_cache
>>> current_cache.get('test')
```

Profiling a Celery Task

To profile a Celery task we need to make sure that the task is executed by the same Python process in which we are collecting the profiling information. That is, the configuration must contain

```
CELERY_TASK_ALWAYS_EAGER = True
CELERY_RESULT_BACKEND = 'cache'
CELERY_CACHE_BACKEND = 'memory'
```

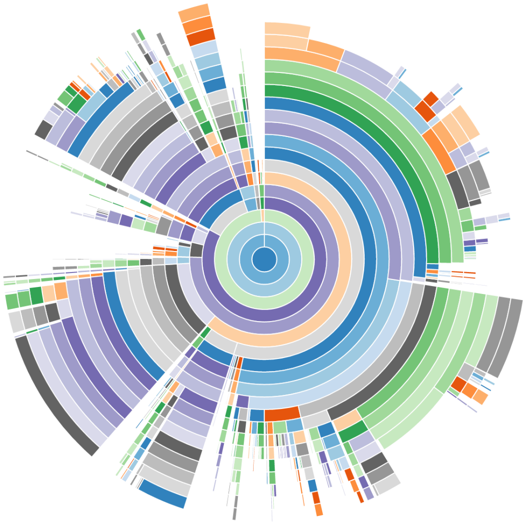
Then, in a Flask shell, we do

```
>>> import cProfile
>>> import pstats
>>> from path.to.our.task import task
>>> pr = cProfile.Profile()
>>> pr.runcall(task, *args, **kwargs)
```

where `*args` and `*kwargs` are the arguments and keyword arguments that we want to pass to `task`. Then

```
>>> ps = pstats.Stats(pr)
>>> ps.dump_stats('task.prof')
```

will create a binary file containing the desired profiling information. To read it we can use snakeviz, which will create a graph such as



Essentially each layer of the graph is a level of the call stack, and the size of the slice is the total time of the function call. For a complete explanation visit the documentation of snakeviz.

Profiling a Request

To profile a request we need to add the following variable to our configuration:

```
PROFILE = True
```

Then we need to attach the WSGI application profiler to our WSGI application. To do this, we need to add a few lines at the bottom of `inspirehep/wsgi.py`:

```
import os; os.mkdir('prof')
from werkzeug.contrib.profiler import ProfilerMiddleware
application = ProfilerMiddleware(application, profile_dir='prof')
```

Now, after we restart the application, a profile report will be created in the `prof` folder for each request that we make. These binary files can be visualized as above with snakeviz.

Rebuild the assets (js/css bundles)

From the root of the code repository, you can run the helper script:

```
$ workon inspire
(inspire)$ ./scripts/clean_assets
```

This will:

1. Remove all your static assets
2. Gather all the npm dependencies and write them in the file *package.json* in the instance static folder
3. Execute *npm install*
4. Execute *inspirehep collect* and *inspirehep assets build*

You should then find all your updated assets in the static folder of your inspire installation, if you are using virtualenv:

```
cdvirtualenv var/inspirehep-instance/static/
```

Rebuild the database, the elasticsearch indexes, and reupload the demo records

Same as the assets, from the root of the code repository, run the script:

```
$ workon inspire
(inspire)$ ./scripts/recreate_records
```

2.2.7 Alembic

Create an alembic revision

We use alembic as a migration tool integrated in *invenio-db*. If you want to create a new alembic revision in INSPIRE you should run the following command:

```
(inspirehep)$ inspirehep alembic revision 'Revision message' -p <parent-revision> --
↪path alembic
```

Consider that you should use as *parent-revision* the last head revision in order to keep a straightforward hierarchical history of alembic revisions. In order to find the last revision for *inspirehep* branch run:

```
(inspirehep)$ inspirehep alembic heads | grep inspirehep
```

and the output will be something similar to:

```
a82a46d12408 (a26f133d42a9, 9848d0149abd) -> fddb3cfe7a9c (inspirehep) (head), Create␣
↪inspirehep tables.
```

From the output we can see that *fddb3cfe7a9c* is the head revision, *a82a46d12408* is it's parent revision and depends on *(a26f133d42a9, 9848d0149abd)* revisions. For more explanatory output you can run:

```
(inspirehep)$ inspirehep alembic heads -vv
```

and search for *inspirehep* branch.

Upgrade to specific alembic revision

If you want to execute a specific alembic revision you should run the following command:

```
(inspirehep)$ inspirehep alembic upgrade <revision_id>
```

In a similar way if you want to revert a specific alembic revision run the following command:

```
(inspirehep)$ inspirehep alembic downgrade <revision_id>
```

Alembic stamp

Alembic stores information about every latest revision that has been applied in to an internal database table called *alembic_version_table*. When we run an upgrade to a specific revision, alembic will search this table and will apply all the revisions sequentially from the last applied until our own. When we run the following command:

```
(inspirehep)$ inspirehep alembic stamp
```

we tell alembic to update this table with all latest revisions that should have been applied without actually applying them. This command is useful when we want to make our migrations up-to-date without calling the migration scripts. For example, if we populate a alembic recipe for creating some new tables but these tables are already present we want to tell alembic to update the version table without applying the missing revisions because in that case will fail during the trial of recreating the already existing tables.

2.3 How to Connect to the PostgreSQL Database

2.3.1 1. About

In inspire-next stores all the data in a postgresql database. This document specifies how to connect and query the inspire's PostgreSQL database. We access it thorught the docker-containers.

2.3.2 2. Run the web container

The first step is run the web container, in order to start our database.

```
$ docker-compose run --rm web
```

2.3.3 3. Connect to the PostgresSQL Database

When all the containers are up you have to open a new console and run the following command line:

```
$ docker-compose exec database psql -U inspirehep
```

```
psql (9.2.18, server 9.4.5)
WARNING: psql version 9.2, server version 9.4.
         Some psql features might not work.
Type "help" for help.

inspirehep=#
```

Now you have an interactive console to query the inspire SQL database. In case PostgreSQL requires the authentication credentials the password for the inspirehep database is *dbpass123*

2.3.4 4. PostgreSQL useful commands

A list of useful commands:

- `\h` this lists all the sql commands that you can run
- `\dt` this lists all the tables
- `\l` this lists all the databases
- `\e` this will open the editor, where you can edit the queries and save it. By doing so the query will get executed.
- `\?` this shows the PSQL command prompt help

2.3.5 5. Search a record with the uuid

Given the uuid of a record you can obtain the record running this query:

```
select * from records_metadata where id = YOUR_UUID;
```

2.3.6 6. Search a record with the pid

Given the pid of a record you can obtain the record running this query:

```
select * from pidstore_pid, records_metadata where records_metadata.id = pidstore_pid.
↪object_uuid where pidstore_pid.id = YOUR_PID_ID;
```

2.4 Operations

INSPIRE operations manual.

2.4.1 Elasticsearch tasks

Simple index remapping

This procedure does not take into account the current database, it acts only on elasticsearch, so any missing records on elasticsearch will not be added, and any modifications made to the db will not be propagated to elasticsearch.

1. Install es-cli:

```
pip install es-cli
```

1. Run the remap command:

```
es-cli remap -m path/to/the/new/mapping.json 'https://user:pass@my.es.instan.ce/
↪myindex'
```

Things to have into account:

- There's no nicer way yet to pass the user/pass
- You can pass more than one '-m–mapping' option if you are using multiple mappings for the same index.
- It creates the new indices with the same aliases that the original had.
- It creates a temporary index in the ES instance, so you will need extra space to allocate it.

Note: It's recommended to create a dump/backup of the index prior to the remapping, just in case.

Dumping an index

This procedure will create a set of json files in a directory containing batches of the index data, including the index metadata (mappings and similar).

```
es-cli dump_index -o backup_dir 'https://user:pass@my.es.instan.ce/myindex'
```

This will create a directory called 'backup_dir' that contains two types of json files, a 'myingex-metadat.json' with the index metadata, and one or more 'myindex-N.json' with the batches of data.

Loading the dump of an index

If you already have dumped an index and you want to load it again, you can run this:

```
es-cli load_index_dump 'https://user:pass@my.es.instan.ce/myindex' backup_dir
```

Where 'backup_dir' is the path to the directory where the index dump was created.

2.4.2 Harvesting and Holding Pen

Handle records in error state

Via web interface

1. Visit Holding Pen list and filter for records in error state.

2. If any, you need to investigate why the record workflow failed, check the detailed page error report.
3. Sometimes the fix is simply to restart the task again if it is due to some circumstantial reasons.

   You can do that from the interface by clicking the "current task" button and hit restart.

Via shell

1. SSH into any worker machine (usually builder to avoid affecting the machines serving users)
2. Enter the shell and retrieve all records in error state:

```
inspirehep shell
```

```
from invenio_workflows import workflow_object_class, ObjectStatus
errors = workflows_object_class.query(status=ObjectStatus.ERROR)
```

3. Get a specific object:

```
from invenio_workflows import workflow_object_class
obj = workflow_object_class.get(1234)
obj.data  #  Check data
obj.extra_data   # Check extra data
obj.status  # Check status
obj.callback_pos  # Position in current workflow
```

4. See associated workflow definition:

```
from invenio_workflows import workflows
workflows[obj.workflow.name].workflow   # Associated workflow list of tasks
```

5. Manipulate position in the workflow

```
obj.callback_pos = [1, 2, 3]
obj.save()
```

6. Restart workflow in various positions:

```
obj.restart_current()  # Restart from current task and continue workflow
obj.restart_next()  # Skip current task and continue workflow
obj.restart_previous()  # Redo task before current one and continue workflow
```

Debug harvested workflows

Note: Added in inspire-crawler => 0.4.0

Sometimes you want to track down the origin of one of the harvest workflows, to do so you can now use the cli tool to get the log of the crawl, and the bare result that the crawler outputted:

```
$ # To get the crawl logs of the workflow 1234
$ inspirehep crawler workflow get_job_logs 1234

$ # To get the crawl result of the workflow 1234
$ inspirehep crawler workflow get_job_result 1234
```

You can also list the crawl jobs, and workflows they started with the commands:

```
$ inspirehep crawler workflow list --tail 50

$ inspirehep crawler job list --tail 50
```

There are also a few more options/commands, you can explore them passing the help flag:

```
$ inspirehep crawler workflow --help

$ inspirehep crawler job --help
```

2.4.3 Operations in QA

Migrate records in QA

The labs database contains a full copy of the legacy records in MARCXML format, called the mirror. Migrating records from legacy involves connecting to the right machine and setting up the work environment, populating the mirror from the file and migrating the records from the mirror, and finally updating the state of the legacy test database.

Setting up the environment

1. First of all establish a Kerberos authentication (this can be helpful: http://linux.web.cern.ch/linux/docs/kerberos-access.shtml)
2. After you have run the `kinit` command and have successfully authenticated you should be able to connect to the builder machine:

```
localhost$ ssh username@inspire-qa-worker3-build1.cern.ch
```

3. Get root access:

```
build1$ sudo -s
```

4. At this point it's a good idea to initialize a screen so you have something to connect to and reestablish your session if something happens to your connection while working remotely to a machine. You can use `byobu`, which is a more user-friendly alternative to `tmux` or `screen`:

```
# This will also reconnect to a running session if any
build1$ byobu
```

5. To finish the setup, you need to get into the Inspire virtual environment:

```
build1# workon inspire
```

Perform the record migration

1. Make sure you have access to the dump of the records on the local machine, for example in your local directory or in `/tmp` (otherwise transfer it there via scp). You can use either a single `.xml.gz` file corresponding to a single legacy dump, or a whole `prodsync.tar` which besides a full first dump contains daily incremental dumps of modified records.

3. Now you can migrate the records, which will be done using the `inspirehep migrate` command:

Note: You shouldn't drop the database or destroy the es index as the existing records will be overwritten with the ones introduced.

```
build1$ inspirehep migrate file --wait filename
```

Note: Instead of doing a full migration from file, it is possible to only populate the mirror or migrate from the mirror. See `inspirhep migrate --help` for more information.

---

4. After migrating the records since we are getting the initial incrementation value for our database records from the legacy test database, you should set the total number of records migrated to the legacy test incrementation table, otherwise every further submission will generate an already existing recid, thus failing:

```
#connect to the legacy qa web node
build1$ ssh inspirevm16.cern.ch

#connect to the legacy qa db
legacy_node$ /opt/cds-invenio/bin/dbexec -i

# to check the autoincrement:
mysql> SHOW CREATE TABLE bibrec;

#to set the new value:
mysql> ALTER TABLE bibrec AUTO_INCREMENT=XXXX;
```

2.5 Harvesting

2.5.1 1. About

This document specifies how to harvest records into your system.

2.5.2 2. Prerequisites (optional)

If you are going to run harvesting workflows which needs prediction models such as the CORE guessing, keyword extraction, and plot extraction you may need to install some extra packages.

Warning: Those additional services (i.e. Beard and Magpie) are not Dockerized, so you will have to do that yourself if the need arises. Instructions below are only applicable if you're running inspire locally, without Docker.

For example, on Ubuntu/Debian you could execute:

```
(inspire)$ sudo aptitude install -y libblas-dev liblapack-dev gfortran imagemagick
```

For guessing, you need to point to a Beard Web service with the config variable `BEARD_API_URL`.

For keyword extraction using Magpie, you need to point to a Magpie Web service with the config variable `MAGPIE_API_URL`.

For hepcrawl crawling of sources via scrapy, you need to point to a scrapyd web service running *hepcrawl* project.

More info at http://pythonhosted.org/hepcrawl/

2.5.3 3. Quick start

All harvesting of scientific articles (hereafter "records") into INSPIRE consist of two steps:

1. Downloading meta-data/files of articles from source and generating INSPIRE style meta-data.
2. Each meta-data record is then taken through an ingestion workflow for pre- and post-processing.

Many records require human acceptance in order to be uploaded into the system. This is done via the Holding Pen web interface located at http://localhost:5000/holdingpen

3.1. Getting records from arXiv.org

Firstly, in order to start harvesting records you will need to deploy the spiders, if you are using docker:

```
docker-compose -f docker-compose.deps.yml run --rm scrapyd-deploy
```

The simplest way to get records into your system is to harvest from arXiv.org using OAI-PMH.

To do this we use *inspire-crawler* CLI tool `inspirehep crawler`.

See the diagram in hepcrawl documentation to see what happens behind the scenes.

Single records like this (if you are running docker, you first will need to open bash and get into the virtual environment in one of the workers, e.g. `docker-compose run --rm web bash`, read the *3.2. Getting records from other sources (no Docker)* section if you aren't using docker):

```
(inspire)$ inspirehep crawler schedule arXiv_single article \
    --kwarg 'identifier=oai:arXiv.org:1604.05726'
```

Range of records like so:

```
(inspire)$ inspirehep crawler schedule arXiv article \
    --kwarg 'from_date=2016-06-24' \
    --kwarg 'until_date=2016-06-26' \
    --kwarg 'sets=physics:hep-th'
```

You can now see from your Celery logs that tasks are started and workflows are executed. Visit the Holding Pen interface, at http://localhost:5000/holdingpen to find the records and to approve/reject them. Once approved, they are queued for upload into the system.

3.2. Getting records from other sources (no Docker)

Example above shows in the simplest case how you can use *hepcrawl* to harvest Arxiv, however hepcrawl can harvest any source so long as it has a spider for that source.

It works by scheduling crawls via certain triggers in *inspirehep* to a *scrapyd* service which then returns harvested records and ingestion workflows are triggered.

First make sure you have setup a scrapyd service running hepcrawl (http://pythonhosted.org/hepcrawl/operations.html) and flower (workermon) running (done automatic with *honcho*).

In your local config (*${VIRTUAL_ENV}/var/inspirehep-instance/inspirehep.cfg*) add the following configuration:

```
CRAWLER_HOST_URL = "http://localhost:6800"   # replace with your scrapyd service
CRAWLER_SETTINGS = {
    "API_PIPELINE_URL": "http://localhost:5555/api/task/async-apply",   # URL to your␣
↪flower instance
    "API_PIPELINE_TASK_ENDPOINT_DEFAULT": "inspire_crawler.tasks.submit_results"
}
```

Now you are ready to trigger harvests. There are two options on how to trigger harvests, from the CLI or code.

Via shell:

```
from inspire_crawler.tasks import schedule_crawl
schedule_crawl(spider, workflow, **kwargs)
```

Via inspirehep cli:

```
(inspire)$ inspirehep crawler schedule --kwarg 'sets=hep-ph,math-ph' --kwarg 'from_
↪date=2018-01-01' arXiv article
```

If your scrapyd service is running you should see output appear from it shortly after harvesting. You can also see from your Celery logs that tasks are started and workflows are executed. Visit the Holding Pen interface, at http://localhost:5000/holdingpen to find the records and to approve/reject them. Once approved, they are queued for upload into the system.

3.2. Getting records from other sources (with Docker)

It works by scheduling crawls via certain triggers in *inspirehep* to a *scrapyd* service which then returns harvested records and ingestion workflows are triggered.

Scrapyd service and configuration for inspire-next will be automatically set up by docker-compose, so you don't have to worry about it.

If you have not previously deployed your spiders, you will have to do it like so:

```
docker-compose -f docker-compose.deps.yml run --rm scrapyd-deploy
```

Afterwards you can schedule a harvest from the CLI or shell:

```
from inspire_crawler.tasks import schedule_crawl
schedule_crawl(spider, workflow, **kwargs)
```

Via inspirehep cli:

```
(inspire docker)$ inspirehep crawler schedule arXiv article --kwarg 'sets=hep-ph,math-
↪ph' --kwarg 'from_date=2018-01-01'
```

Where *arXiv* is any spider in hepcrawl/spiders/ and each of the kwarg``s is a parameter to the spiders ``__init__.

2.6 GROBID

2.6.1 1. About

This document specifies how to train and use GROBID.

2.6.2 2. Prerequisites

GROBID uses Maven as its build system. To install it on Debian/Ubuntu systems we just have to type:

```
$ sudo apt-get install maven
```

Note that this will also install Java, the language GROBID is written in. Similar commands apply to other distributions. In particular for OS X we have:

```
$ brew install maven
```

2.6.3 3. Quick start

To install GROBID we first need to clone its code:

```
$ git clone https://github.com/inspirehep/grobid
```

Note that we are fetching it from our fork instead of the main repository because our HEP training data has not yet been merged inside of it. Now we move inside its `grobid-service` folder and start the service:

```
$ cd grobid/grobid-service
$ mvn jetty:run-war
```

This will run the tests, load the modules and start a service available at `localhost:8080`.

2.6.4 4. Training

The models available after cloning are not using the new available training data. To generate the new ones we need to go inside of the root folder and call:

```
$ cd grobid
$ java -Xmx1024m -jar grobid-trainer/target/grobid-trainer-0.3.4-SNAPSHOT.one-jar.jar␣
↪0 $MODEL -gH grobid-home
```

where `$MODEL` is the model we want to train. Note that there's new data only for the `segmentation` and `header` models.

Moreover, note that the `0` parameter instructs GROBID to only train the models. A value of `1` will only evaluate the trained model on a random subset of the data, while a value of `2` requires an additional parameter:

```
$ java -Xmx1024m -jar grobid-trainer/target/grobid-trainer-0.3.4-SNAPSHOT.one-jar.jar␣
↪0 $MODEL -gH grobid-home -s$SPLIT
```

where `$SPLIT` is a float between 0 and 1 that represents the ratio of data to be used for training.

2.7 Inspire Tests

2.7.1 How to Run the Selenium Tests

Via Docker

1. If you have not installed `docker` and `docker-compose`, install them now.

2. Run `docker`:

```
$ docker-compose -f docker-compose.test.yml run --rm acceptance
```

Via Docker with a graphical instance of Firefox (Linux)

1. Check the first step in the *Via Docker* section.
2. Add the root user to the list allowed by **X11**:

```
$ xhost local:root
non-network local connections being added to access control list
```

3. Run `docker`:

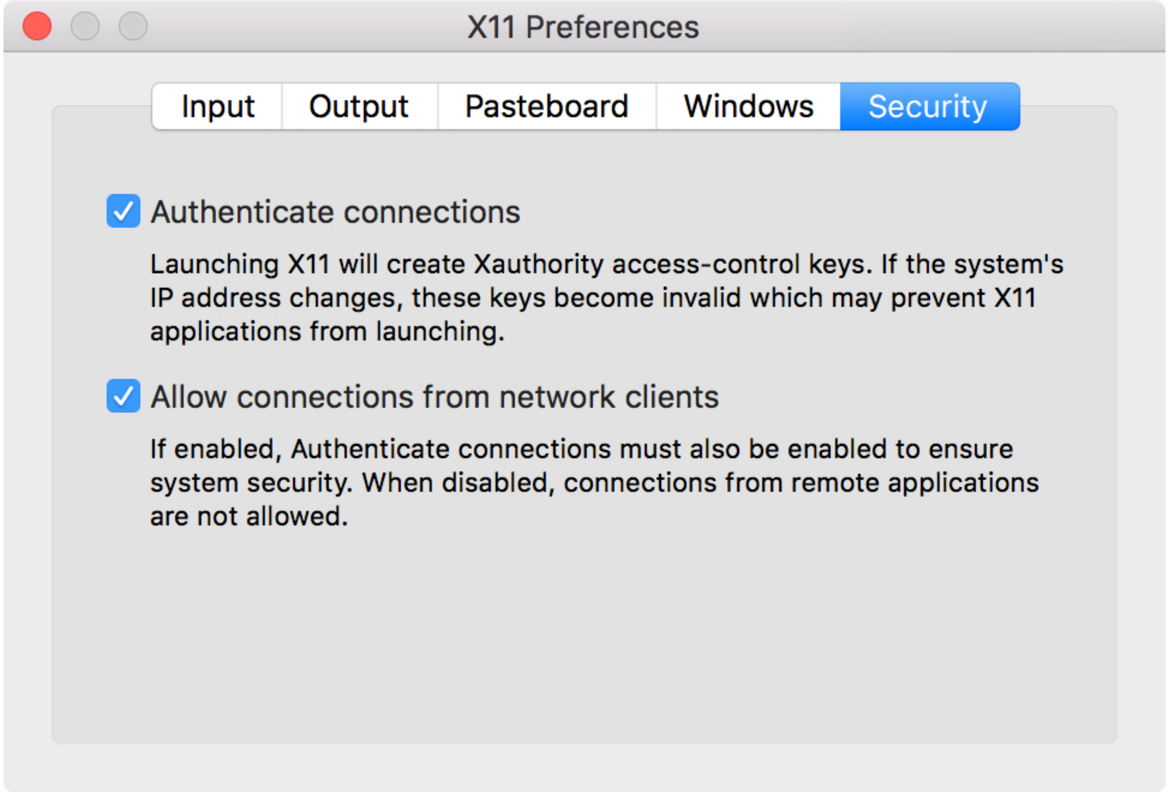
```
$ docker-compose -f docker-compose.test.yml run --rm visible_acceptance
```

Via Docker with a graphical instance of Firefox (macOS)

1. Check the first step in the *Via Docker* section.
2. Install **XQuartz**: go to the XQuartz website and install the latest version. In alternative, run:

```
$ brew cask install xquartz
```

3. Having installed **XQuartz**, run it and open the **XQuartz** -> **Preferences** menu from the bar. Go to the last tab, **Security**, enable both the **“Authenticate connections”** and **“Allow connections from network clients”** checkboxes, then restart your computer.



☑ Authenticate connections

Launching X11 will create Xauthority access-control keys. If the system's IP address changes, these keys become invalid which may prevent X11 applications from launching.

☑ Allow connections from network clients

If enabled, Authenticate connections must also be enabled to ensure system security. When disabled, connections from remote applications are not allowed.

4. Write down the IP address of your computer because you will need it later:

```
$ ifconfig en0 | grep inet | awk '$1=="inet" {print $2}'
123.456.7.890
```

5. Add the IP address of your computer to the list allowed by **XQuartz**:

```
$ xhost + 123.456.7.890
123.456.7.890 being added to access control list
```

6. Set the `$DISPLAY` environment variable to the same IP address, followed by the id of your display (in this case, `:0`):

```
$ export DISPLAY=123.456.7.890:0
```

7. Run `docker`:

```
$ docker-compose -f docker-compose.test.yml run --rm visible_acceptance
```

2.7.2 How to Write the Selenium Tests

Selenium Test Framework

INSPIRE's Selenium tests are written using an in-house framework called BAT (`inspirehep/bat`). The framework is made of four main components:

- *Tests*
- *Pages*
- *Arsenic*
- *ArsenicResponse*

Tests

Tests don't call directly Selenium methods, but call methods on *Pages*, which are eventually translated to Selenium calls.

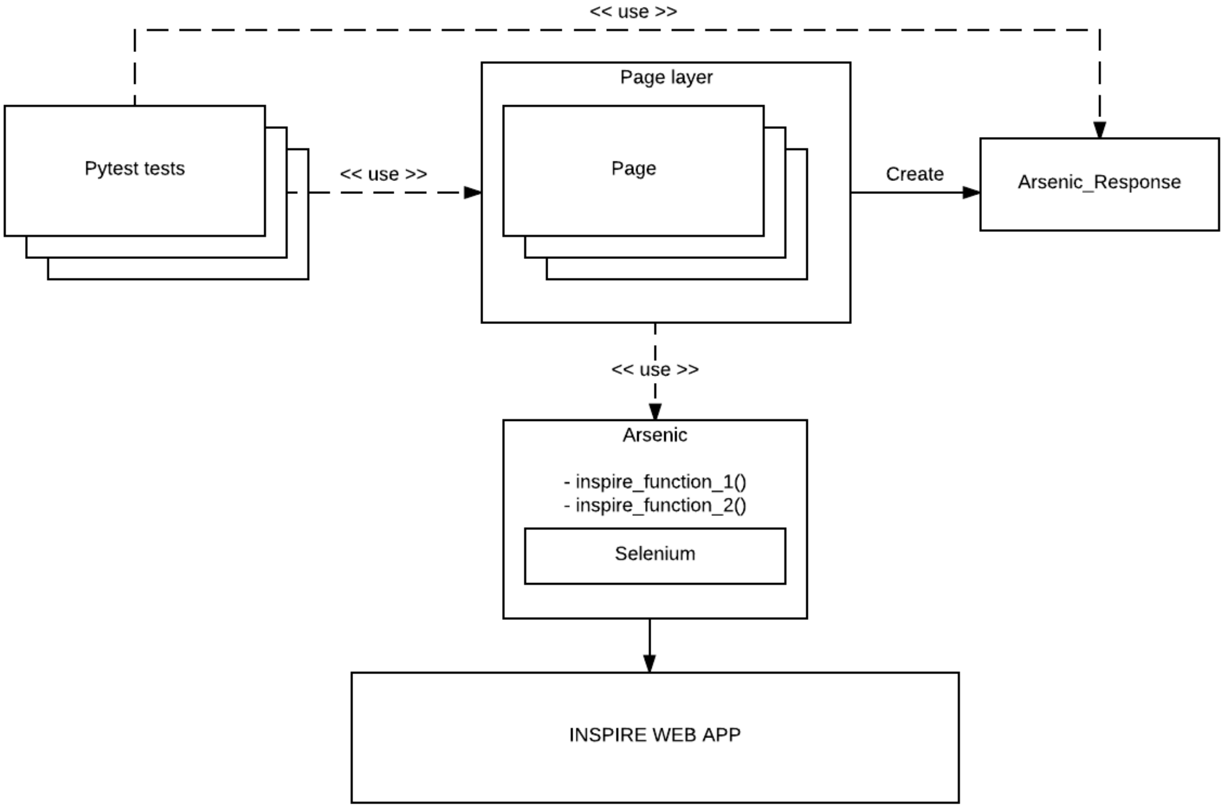
Tests are intended to be imperative descriptions of what the user does and what they expect to see. For example

```
def test_mail_format(login):
    author_submission_form.go_to()
    author_submission_form.write_mail('wrong mail').assert_has_error()
    author_submission_form.write_mail('me@me.com').assert_has_no_error()
```

asserts that, when the user visits the "Create Author" page and writes `wrong mail`, they see an error, while when they visit the same page but write a valid email, they don't see it.

Pages

Pages are abstractions of web pages served by INSPIRE. Concretely, a page is a collection of methods in a module that implement the various action that a user can take when interacting with that page. For example the



<< use >>
Pytest tests
<< use >>
Page layer
Page
Create
Arsenic_Response
<< use >>
Arsenic
- inspire_function_1()
- inspire_function_2()
Selenium
INSPIRE WEB APP

```
def go_to():
    Arsenic().get(os.environ['SERVER_NAME'] + '/authors/new')
```

method in `inspirehep/bat/pages/author_submission_form.py` represents the action of visiting the "Create Author" page, while

```
def write_institution(institution, expected_data):
    def _assert_has_error():
        assert expected_data in Arsenic().write_in_autocomplete_field(
            'institution_history-0-name', institution)

    return ArsenicResponse(assert_has_error=_assert_has_error)
```

in the same module represents the action of filling the autocomplete field of id `institution_history-0-name` with the content of the `institution` variable.

Note that the latter method returns a closure over `expected_data` and `institution` which is going to be used by an `assert_has_error` call to determine if the action was successful or not.

Arsenic

The `Arsenic` class is a proxy to the Selenium object, plus some INSPIRE-specific methods added on top.

ArsenicResponse

As mentioned above, an `ArsenicResponse` wraps a closure that is going to be used by an `assert_has_error` or `assert_has_no_error` call to determine if the action executed successfully or not.

2.7.3 How to Debug the Selenium Tests

Unlike the other test suites, the container that is running the test code of the `acceptance` test suite is different from the one running the application code. Therefore, in order to debug a test failure, we must connect remotely to this other container. The tool to achieve this is called remote-pdb. This section explains how to use it.

1. First we install it in the container:

```
$ docker-compose run --rm web pip install remote-pdb
```

2. Then we insert the following code where we want to start tracing:

```
from remote_pdb import RemotePdb
RemotePdb('0.0.0.0', 4444).set_trace()
```

3. Now we run the `acceptance` test suite:

```
$ docker-compose -f docker-compose.test.yml run --rm acceptance
```

4. At some point the test suite will stop: it means that we have hit the tracing call. We discover the IP of the `web` container with:

```
$ docker inspect inspirenext_test-web_1 | grep IPAddress
[...]
"IPAddress": "172.18.0.6"
```

5. Finally, we connect to it with:

```
$ telnet 172.18.0.6 4444
```

2.8 E2E Test Writing Tutorial

For the tutorial we will try to test the first part of the harvest. We will try to harvest arXiv and then assert that a holdingpen entry for the harvested record appears.

2.8.1 Fixtures

Let's create a test file *tests/e2e/test_arxiv_in_hp.py* in INSPIRE-Next. To run our tests we will need to import a few things and set up some fixtures:

```
import os
import pytest
import time

from inspirehep.testlib.api import InspireApiClient
from inspirehep.testlib.api.mitm_client import MITMClient, with_mitmproxy

@pytest.fixture
def inspire_client():
    # INSPIRE_API_URL is set by k8s when running the test in Jenkins
    inspire_url = os.environ.get('INSPIRE_API_URL', 'http://test-web-e2e.local:5000')
    return InspireApiClient(base_url=inspire_url)


@pytest.fixture
def mitm_client():
    mitmproxy_url = os.environ.get('MITMPROXY_HOST', 'http://mitm-manager.local')
    return MITMClient(mitmproxy_url)
```

`InspireApiClient` is used to interact with INSPIRE through the API. Using it we can for example trigger a harvest, or request holdingpen entries. `MITMClient` is a similar client for the proxy, with it we can swap scenarios, enable recording of interactions, or make assertions based on what happened during the test. `with_mitmproxy` is a helper decorator, that will automatically set up the scenario for you (scenario name will match the test name) and optionally, if you specify `record=True`, enable recording for the duration of the test.

We will also need the following fixture to set up all of the dummy fixtures and records in the test instance of INSPIRE. Most likely when writing a real test this fixture will already be present, as it is needed for virtually any test:

```
@pytest.fixture(autouse=True, scope='function')
def init_environment(inspire_client):
    inspire_client.e2e.init_db()
    inspire_client.e2e.init_es()
    inspire_client.e2e.init_fixtures()
    # refresh login session, giving a bit of time
    time.sleep(1)
    inspire_client.login_local()
```

2.8.2 Interaction Recording

Now that we have set up all of the necessary fixtures, we can attempt to start writing our test. We add a wait (for now, we will improve it later in the tutorial) at the end as to give time for INSPIRE to harvest, pull the pdf and the eprint, etc. Without this, the test would finish immediately after scheduling the crawl, which would deregister the scenario and disable recording. Later on, we will add actual polling to see if the articles were harvested.

```
@with_mitmproxy(should_record=True)
def test_arxiv_in_hp(inspire_client, mitm_client):
    inspire_client.e2e.schedule_crawl(
        spider='arXiv_single',
        workflow='article',
        url='http://export.arxiv.org/oai2',
        identifier='oai:arXiv.org:1806.04664',  # Non-core, will halt
    )

    time.sleep(60)  # Let's wait for INSPIRE to harvest the records
```

Let us now run this “test” and see what happens:

```
docker-compose -f docker-compose.test.yml run --rm e2e pytest tests/e2e/test_arxiv_in_
↪hp.py
```

2.8.3 Proxy Web UI

After the test started running we can use the proxy’s web interface to look at the requests that are happening during the test session. The proxy exposes its web interface on port 8081, so open your browser and navigate to `http://127.0.0.1:8081`.

There you will see initial requests to RT, ElasticSearch and so on, logging in to INSPIRE. These are followed by requests to the `mitm-manager.local` that set up the test scenario (`PUT /config`) and and recording (`POST /record`).

After this all the requests (until disabling recording and/or switching the scenario) belong to the current test session. Many of them (`test-indexer`, `test-web-e2e.local`) are whitelisted and not recorded. You might notice a few requests to ArXiv like so:

- `GET http://export.arxiv.org/oai2?verb=GetRecord&metadataPrefix=arXiv&identifier=oai...`
- `GET http://export.arxiv.org/pdf/1806.04664`
- `GET http://export.arxiv.org/e-print/1806.04664`

These are live interactions that are recorded, you can find them in `tests/e2e/scenarios/arxiv_in_hp/ArxivService/`. If you need to re-record an interaction, simply remove the file you want to overwrite or rename it in such a way that it doesn’t have a *yaml* extension.

Tip: Since the responses from ArXiv come compressed, in order to preserve the original test data, this is also the way they are stored. If you need to look inside, you can copy the body from the yaml, and assuming it’s pasted in another file called `gzip.txt` run:

```
cat gzip.txt | base64 -di | gzip -d > plain.txt
```

Similarily to compress it back:

```
cat plain.txt | gzip | base64 > gzip.txt
```

2.8.4 Querying the Holdingpen

Now that our interactions are recorded we can go ahead and finish our test, by making assertions on the holdingpen records. We can also remove the `should_record=True` option from the `@with_mitmproxy` decorator, as our interactions are now recorded.

To make assertions we can use the `inspire_client` and more precisely its `holdingpen` module:

```
@with_mitmproxy
def test_arxiv_in_hp(inspire_client, mitm_client):
    inspire_client.e2e.schedule_crawl(
        spider='arXiv_single',
        workflow='article',
        url='http://export.arxiv.org/oai2',
        identifier='oai:arXiv.org:1806.04664',
    )

    time.sleep(60)

    holdingpen_entries = inspire_client.holdingpen.get_list_entries()

    assert len(holdingpen_entries) == 1

    holdingpen_entry = holdingpen_entries[0]

    assert holdingpen_entry.status == 'HALTED'
    assert holdingpen_entry.core is None
    assert holdingpen_entry.arxiv_eprint == '1806.04664'
```

This test needs to be refactored to not use a “simple” `time.sleep`, but actual polling, but already it should work.

2.8.5 Further Improvements

As mentioned before, we can introduce a fixture which will enable us to poll until harvest was finished, instead of having a simple `time.sleep` (snippet taken from `tests/e2e/test_arxiv_harvest.py`):

```
def wait_for(func, *args, **kwargs):
    max_time = kwargs.pop('max_time', 200)
    interval = kwargs.pop('interval', 2)

    decorator = backoff.on_exception(
        backoff.constant,
        AssertionError,
        interval=interval,
        max_time=max_time,
    )
    decorated = decorator(func)
    return decorated(*args, **kwargs)
```

We can then use the fixture in our test:

```
@with_mitmproxy
def test_arxiv_in_hp(inspire_client, mitm_client):
    inspire_client.e2e.schedule_crawl(
        spider='arXiv_single',
        workflow='article',
        url='http://export.arxiv.org/oai2',
```

```
        identifier='oai:arXiv.org:1806.04664',
    )

    def _in_holdinpen():
        holdingpen_entries = inspire_client.holdingpen.get_list_entries()
        assert len(holdingpen_entries) > 0
        assert holdingpen_entries[0].status == 'HALTED'
        return holdingpen_entries

    # Will poll every two seconds and timeout after 200 seconds
    holdingpen_entries = wait_for(_in_holdinpen)

    assert len(holdingpen_entries) == 1

    holdingpen_entry = holdingpen_entries[0]

    assert holdingpen_entry.core is None
    assert holdingpen_entry.arxiv_eprint == '1806.04664'
```

We can also use the mitmproxy client to make assertions on the interactions with external services that happened during our test:

```
@with_mitmproxy
def test_arxiv_in_hp(inspire_client, mitm_client):
    # ... ...
    mitm_client.assert_interaction_used('ArxivService', 'interaction_0', times=1)
```

Above will fail if the interaction `scenarios/arxiv_in_hp/ArxivService/interaction_0.yaml` has not been used exactly one time. You can leave off the `times` parameter if you want to assert that the interaction happened at least once, instead of specifying exactly the number of times. Names of interactions are not important so you can rename them if you like. Naming only matters if two interactions can match the same request: in such case the lexicographically first one is chosen for consistency.

2.8.6 Troubleshooting/Tips

Accessing web node in browser

If for any reason you need to access the web interface of INSPIRE, you can add an entry to your `/etc/hosts` file with the IP of the web container:

```
$ docker inspect inspirenext_test-web-e2e.local_1 | grep '"IPAddress"'

            "IPAddress": "",
                "IPAddress": "172.20.0.9",

$ sudo vim /etc/hosts
```

And add a line at the bottom:

```
172.20.0.9 test-web-e2e.local
```

Now you can visit http://test-web-e2e.local:5000 in your browser, provided the container is running.

Docker cheatsheet

In order to start the web container (don't forget the `.local` at the end!):

```
docker-compose -f docker-compose.test.yml up test-web-e2e.local
```

For any other container, change the `test-web-e2e.local` to the suitable name; other containers don't end in `.local`, this is needed only for inspire-next node as it has to be a domain name.

Similarily substitute `up` for `stop` or `kill` to bring it down, and `rm` to remove the container (e.g. so that the new updated image can be used).

To view the logs of a container:

```
docker-compose -f docker-compose.test.yml logs test-worker-e2e
```

In order to run a shell in an already running container (e.g. to investigate errors):

```
# E.g. for INSPIRE
docker-compose -f docker-compose.test.yml exec test-web-e2e.local bash

# For MITM-Proxy we use `ash`, as it runs on Alpine Linux base, which doesn't ship␣
↪with `bash`
docker-compose -f docker-compose.test.yml exec mitm-proxy ash
```

2.9 Building this docs page

Sometimes when you modify the docs it's convenient to generate them locally in order to check them before sending a pull request, to do so, you'll have to install some extra dependencies:

Note: Remember that you'll need a relatively newer version of setuptools and pip, so if you just created a virtualenv for the docs, you might have to run:

```
(inspirehep_docs)$ pip install --upgrade setuptools pip
```

Also keep in mind that you need all the inspire system dependecies installed too, if you don't have them, go to Installation

```
(inspirehep_docs)$ pip install -e .[all]
```

And then, you can generate the html docs pages with:

```
(inspirehep_docs)$ make -C docs html
```

And to view them, you can just open them in your favourite browser:

```
(enspirehep_docs)$ firefox docs/_build/html/index.html
```

2.10 inspirehep package

2.10.1 Subpackages

inspirehep.bat package

Subpackages

inspirehep.bat.pages package

Submodules

inspirehep.bat.pages.author_submission_form module

`inspirehep.bat.pages.author_submission_form.`**`go_to`**`()`

`inspirehep.bat.pages.author_submission_form.`**`submit_author`**`(`*input_data*`)`

`inspirehep.bat.pages.author_submission_form.`**`submit_empty_form`**`(`*expected_data*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_advisor`**`(`*advisor*, *expected_data*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_experiment`**`(`*experiment*, *expected_data*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_institution`**`(`*institution*, *expected_data*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_mail`**`(`*mail*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_orcid`**`(`*orcid*`)`

`inspirehep.bat.pages.author_submission_form.`**`write_year`**`(`*input_id*, *error_message_id*, *year*`)`

inspirehep.bat.pages.holdingpen_author_detail module

`inspirehep.bat.pages.holdingpen_author_detail.`**`accept_record`**`()`

`inspirehep.bat.pages.holdingpen_author_detail.`**`curation_record`**`()`

`inspirehep.bat.pages.holdingpen_author_detail.`**`go_to`**`()`

`inspirehep.bat.pages.holdingpen_author_detail.`**`load_submitted_record`**`(`*input_data*`)`

`inspirehep.bat.pages.holdingpen_author_detail.`**`reject_record`**`()`

`inspirehep.bat.pages.holdingpen_author_detail.`**`review_record`**`(`*input_data*`)`

inspirehep.bat.pages.holdingpen_author_list module

`inspirehep.bat.pages.holdingpen_author_list.`**`click_first_record`**`()`

`inspirehep.bat.pages.holdingpen_author_list.`**`go_to`**`()`

`inspirehep.bat.pages.holdingpen_author_list.`**`load_submission_record`**`(`*input_data*`)`

inspirehep.bat.pages.holdingpen_literature_detail module

inspirehep.bat.pages.holdingpen_literature_detail.**accept_record**()

inspirehep.bat.pages.holdingpen_literature_detail.**assert_first_record_matches**(*input_data*, *try_count=0*)

inspirehep.bat.pages.holdingpen_literature_detail.**go_to**()

inspirehep.bat.pages.holdingpen_literature_list module

inspirehep.bat.pages.holdingpen_literature_list.**assert_first_record_completed**()

inspirehep.bat.pages.holdingpen_literature_list.**assert_first_record_matches**(*input_data*)

inspirehep.bat.pages.holdingpen_literature_list.**click_first_record**()

inspirehep.bat.pages.holdingpen_literature_list.**get_first_record_info**(*try_count=0*)

inspirehep.bat.pages.holdingpen_literature_list.**go_to**()

inspirehep.bat.pages.literature_submission_form module

class inspirehep.bat.pages.literature_submission_form.**InputData**(*data=None*)

Bases: object

add_basic_info(*abstract*, *title*, *language*, *title_translation*, *collaboration*, *experiment*, *authors=()*, *report_numbers=()*, *subjects=()*)

add_book_chapter_info(*book_title*, *page_start*, *page_end*)

add_book_info(*book_title*, *book_volume*, *publication_date*, *publication_place*, *publisher_name*)

add_journal_info(*journal_title*, *volume*, *issue*, *year*, *page_range*, *conf_name*)

add_links(*pdf_url*)

add_proceedings(*nonpublic_note*)

add_references_comments(*references*, *extra_comments*)

add_thesis_info(*defense_date*, *degree_type*, *institution*, *supervisor_affiliation*, *supervisor_name*, *thesis_date*)

get(**args*, ***kwargs*)

inspirehep.bat.pages.literature_submission_form.**go_to**()

inspirehep.bat.pages.literature_submission_form.**submit_article**(*input_data*)

inspirehep.bat.pages.literature_submission_form.**submit_arxiv_id**(*arxiv_id*, *expected_data*)

inspirehep.bat.pages.literature_submission_form.**submit_book**(*input_data*)

inspirehep.bat.pages.literature_submission_form.**submit_chapter**(*input_data*)

inspirehep.bat.pages.literature_submission_form.**submit_doi_id**(*doi_id*, *expected_data*)

inspirehep.bat.pages.literature_submission_form.**submit_journal_article**(*input_data*)

inspirehep.bat.pages.literature_submission_form.**submit_journal_article_with_proceeding**(*input*

inspirehep.bat.pages.literature_submission_form.**submit_thesis**(*input_data*) `inspirehep.bat.pages.literature_submission_form.`**`write_affiliation`**(*affiliation*, *expected_data*)

`inspirehep.bat.pages.literature_submission_form.`**`write_conference`**(*conference_title*, *expected_data*)

`inspirehep.bat.pages.literature_submission_form.`**`write_date_thesis`**(*date_field*, *error_message_id*, *date*)

`inspirehep.bat.pages.literature_submission_form.`**`write_institution_thesis`**(*institution*, *expected_data*)

`inspirehep.bat.pages.literature_submission_form.`**`write_journal_title`**(*journal_title*, *expected_data*)

`inspirehep.bat.pages.literature_submission_form.`**`write_pdf_link`**(*pdf_link*)

inspirehep.bat.pages.top_navigation_page module

`inspirehep.bat.pages.top_navigation_page.`**`am_i_logged`**()

`inspirehep.bat.pages.top_navigation_page.`**`log_in`**(*user_id*, *password*)

`inspirehep.bat.pages.top_navigation_page.`**`log_out`**()

Module contents

BAT framework pages.

Submodules

inspirehep.bat.EC module

Module for custom selenium ‘Expected Conditions’.

See also:

http://selenium-python.readthedocs.io/waits.html

class `inspirehep.bat.EC.`**`GetText`**(*locator*)

Bases: `object`

An Expectation that waits until an element has text.

Todo: Better filter out the *WebDriverException* s .

class `inspirehep.bat.EC.`**`TryClick`**(*locator*)

Bases: `object`

An Expectation that tries to click an element.

Is very similar to *EC.element_to_be_clickable*, but actually works.

Todo

Better filter out the *WebDriverException* s .

---

inspirehep.bat.actions module

`inspirehep.bat.actions.`**`click`**`(`*_id=None*, *xpath=None*, *link_text=None*`)`

`inspirehep.bat.actions.`**`get_text_of`**`(`*_id=None*, *xpath=None*, *link_text=None*`)`

`inspirehep.bat.actions.`**`get_value_of`**`(`*_id=None*, *xpath=None*, *link_text=None*`)`

`inspirehep.bat.actions.`**`select`**`(`*value*, *_id=None*, *xpath=None*, *link_text=None*`)`

`inspirehep.bat.actions.`**`wait_for`**`(`*_id=None*, *xpath=None*, *link_text=None*`)`

`inspirehep.bat.actions.`**`write`**`(`*data*, *_id=None*, *xpath=None*, *link_text=None*`)`

inspirehep.bat.arsenic module

class `inspirehep.bat.arsenic.`**`Arsenic`**`(`**args*`)`

Bases: `object`

`click_with_coordinates``(`*element_id*, *x*, *y*`)`

`hide_title_bar``()`

`show_title_bar``()`

`write_in_autocomplete_field``(`*field_id*, *field_value*`)`

class `inspirehep.bat.arsenic.`**`ArsenicResponse`**`(`*assert_has_no_errors_func=None*, *assert_has_errors_func=None*`)`

Bases: `object`

`assert_has_errors``()`

`assert_has_no_errors``()`

Module contents

INSPIRE BAT framework.

inspirehep.modules package

Subpackages

inspirehep.modules.accounts package

Subpackages

inspirehep.modules.accounts.views package

Submodules

inspirehep.modules.accounts.views.login module

`inspirehep.modules.accounts.views.login.`**`login`**`()`

Module contents

Submodules

inspirehep.modules.accounts.ext module

Accounts extension.

class `inspirehep.modules.accounts.ext.`**`InspireAccounts`**(*app=None*)
Bases: `object`

`init_app`(*app*)

Module contents

INSPIRE Accounts module.

inspirehep.modules.api package

Subpackages

inspirehep.modules.api.v1 package

Submodules

inspirehep.modules.api.v1.common_serializers module

Common (to all collections) API of INSPIRE.

class `inspirehep.modules.api.v1.common_serializers.`**`APIRecidsSerializer`**
Bases: `object`

Recids serializer.

serialize_search(*pid_fetcher*, *search_result*, *item_links_factory=None*, *links=None*)

Module contents

Version 1 of the API of INSPIRE.

Module contents

API of INSPIRE.

inspirehep.modules.arxiv package

Submodules

inspirehep.modules.arxiv.config module

ArXiv configuration.

inspirehep.modules.arxiv.core module

ArXiv Core.

`inspirehep.modules.arxiv.core.`**get_json**(*arxiv_id*)

`inspirehep.modules.arxiv.core.`**get_response**(*arxiv_id*)

inspirehep.modules.arxiv.ext module

ArXiv extension.

class `inspirehep.modules.arxiv.ext.`**InspireArXiv**(*app=None*)

Bases: `object`

init_app(*app*)

init_config(*app*)

inspirehep.modules.arxiv.utils module

`inspirehep.modules.arxiv.utils.`**etree_to_dict**(*tree*)

Translate etree into dictionary.

Parameters **tree** (<http://lxml.de/api/lxml.etree-module.html>) – etree dictionary object

inspirehep.modules.arxiv.views module

ArXiv blueprints.

`inspirehep.modules.arxiv.views.`**search**(**args*, ***kwargs*)

Module contents

INSPIRE arXiv module.

inspirehep.modules.authors package

Subpackages

inspirehep.modules.authors.dojson package

Subpackages

inspirehep.modules.authors.dojson.fields package

Submodules

inspirehep.modules.authors.dojson.fields.updateform module

Author update/addition form JSON conversion.

Converts keys in the user form to the keys needed by the HepNames data model in order to produce MARCXML.

`inspirehep.modules.authors.dojson.fields.updateform.`**`advisors`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`arxiv_categories`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`bai`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`blog_url`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`control_number`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`display_name`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`experiments`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`inspireid`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`institution_history`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`linkedin_url`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`native_name`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`orcid`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`public_email`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`status`**(*self*, *key*, *value*) `inspirehep.modules.authors.dojson.fields.updateform.`**`twitter_url`**(*self*, *key*, *value*)

`inspirehep.modules.authors.dojson.fields.updateform.`**`websites`**(*self*, *key*, *value*)

Module contents

Submodules

inspirehep.modules.authors.dojson.model module

Models related to INSPIRE depositions.

Module contents

inspirehep.modules.authors.rest package

Submodules

inspirehep.modules.authors.rest.citations module

class `inspirehep.modules.authors.rest.citations.`**`AuthorAPICitations`**

Bases: `object`

API endpoint for author collection returning citations.

`serialize`(*pid*, *record*, *links_factory=None*)

Return a list of citations for a given author recid.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for the link generation, which are added to the response.

inspirehep.modules.authors.rest.coauthors module

class `inspirehep.modules.authors.rest.coauthors.`**`AuthorAPICoauthors`**

Bases: `object`

API endpoint for author collection returning co-authors.

`serialize`(*pid*, *record*, *links_factory=None*)

Return a list of co-authors for a given author recid.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for the link generation, which are added to the response.

inspirehep.modules.authors.rest.publications module

class `inspirehep.modules.authors.rest.publications.`**`AuthorAPIPublications`**

Bases: `object`

API endpoint for author collection returning publications.

`serialize`(*pid*, *record*, *links_factory=None*)

Return a list of publications for a given author recid.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for the link generation, which are added to the response.

inspirehep.modules.authors.rest.stats module

class `inspirehep.modules.authors.rest.stats.`**`AuthorAPIStats`**

Bases: `object`

API endpoint for author collection returning statistics.

`serialize`(*pid*, *record*, *links_factory=None*)

Return a different metrics for a given author recid.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for the link generation, which are added to the response.

Module contents

Record serialization.

Submodules

inspirehep.modules.authors.bundles module

Bundles for author forms.

inspirehep.modules.authors.ext module

Authors extension.

class `inspirehep.modules.authors.ext.`**`InspireAuthors`**(*app=None*)

Bases: `object`

`init_app`(*app*)

inspirehep.modules.authors.forms module

class `inspirehep.modules.authors.forms.`**`AdvisorsInlineForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Advisors inline form.

`degree_type` = <UnboundField(SelectField, (), {‘default’: ‘phd’, ‘choices’: [(u’bachelor’, u’Bachelor’), (u’diploma’, u’I

`degree_type_options` = [(u’bachelor’, u’Bachelor’), (u’diploma’, u’Diploma’), (u’habilitation’, u’Habilitation’), (u’la

`degree_types_schema` = {u’minLength’: 1, u’enum’: [u’other’, u’diploma’, u’bachelor’, u’laurea’, u’master’, u’phd’

`name` = <UnboundField(TextField, (), {‘autocomplete’: ‘author’, ‘placeholder’: ‘Name. Type for suggestions’, ‘widget’: <

`val` = u’habilitation’

class `inspirehep.modules.authors.forms.`**`AuthorUpdateForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Author update form.

`advisors` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.modules.authors.forms.

`arxiv_categories_schema` = {u’minLength’: 1, u’enum’: [u’astro-ph’, u’astro-ph.CO’, u’astro-ph.EP’, u’astro-ph.

`bai` = <UnboundField(StringField, (), {‘label’: ‘Bai’, ‘widget’: <wtforms.widgets.core.HiddenInput object>, ‘widget_clas

`blog_url` = <UnboundField(StringField, (), {‘widget_classes’: ‘form-control’, ‘label’: ‘Blog’, ‘placeholder’: ‘http://www

`control_number` = <UnboundField(IntegerField, (), {‘widget’: <wtforms.widgets.core.HiddenInput object>, ‘validator

`display_name` = <UnboundField(StringField, (), {‘label’: ‘Display Name’, ‘validators’: [<wtforms.validators.DataRequ

`experiments` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.modules.authors.fo

`extra_comments` = <UnboundField(TextAreaField, (), {‘label’: ‘Comments’, ‘description’: u’Send us any comments yo

`family_name` = <UnboundField(StringField, (), {‘label’: ‘Family Name’, ‘description’: u’e.g. Mart\xednez Santos’, ‘wid

`given_names` = <UnboundField(StringField, (), {‘label’: ‘Given Names’, ‘validators’: [<wtforms.validators.DataRequir

`groups` = [(‘Personal Information’, [‘given_names’, ‘family_name’, ‘display_name’, ‘native_name’, ‘email’, ‘public_ema

`inspireid` = <UnboundField(StringField, (), {‘label’: ‘Inspireid’, ‘widget’: <wtforms.widgets.core.HiddenInput object>

`institution_history` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.module

`linkedin_url` = <UnboundField(StringField, (), {‘widget_classes’: ‘form-control’, ‘label’: ‘Linkedin’, ‘placeholder’: ‘

`native_name` = <UnboundField(StringField, (), {‘label’: ‘Native Name’, ‘description’: u’For non-Latin names e.g. \u9ea

`orcid` = <UnboundField(StringField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.WrappedInput object>, ‘de

`public_emails` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.modules.author

`research_field` = <UnboundField(SelectMultipleField, (), {‘choices’: [(u’astro-ph’, u’astro-ph’), (u’cond-mat’, u’con

`research_field_options` = [(u’astro-ph’, u’astro-ph’), (u’cond-mat’, u’cond-mat’), (u’cs’, u’cs’), (u’econ’, u’econ’),

`status` = <UnboundField(SelectField, (), {‘default’: ‘active’, ‘choices’: [(‘active’, ‘Active’), (‘retired’, ‘Retired’), (‘depar

`status_options` = [(‘active’, ‘Active’), (‘retired’, ‘Retired’), (‘departed’, ‘Departed’), (‘deceased’, ‘Deceased’)]

`twitter_url` = <UnboundField(StringField, (), {‘widget_classes’: ‘form-control’, ‘label’: ‘Twitter’, ‘placeholder’: ‘http

`val` = u’stat.TH’

`websites` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.modules.authors.forms.

class `inspirehep.modules.authors.forms.`**`ColumnSelect`**`(`*widget=None*, *wrapper=None*, ***kwargs*`)`

Bases: *`inspirehep.modules.authors.forms.WrappedSelect`*

Specialized column wrapped input.

`wrapper`

Wrapper template with description support.

class `inspirehep.modules.authors.forms.`**`DynamicUnsortedItemWidget`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_widgets.DynamicItemWidget`*

class `inspirehep.modules.authors.forms.`**`DynamicUnsortedNonRemoveItemWidget`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_widgets.DynamicItemWidget`*

class `inspirehep.modules.authors.forms.`**`DynamicUnsortedNonRemoveWidget`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_widgets.DynamicListWidget`*

class `inspirehep.modules.authors.forms.`**`DynamicUnsortedWidget`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_widgets.DynamicListWidget`*

class `inspirehep.modules.authors.forms.`**`EmailInlineForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Public emails inline form.

`email` **= <UnboundField(StringField, (), {‘widget_classes’: ‘form-control’, ‘validators’: [<wtforms.validators.Optional ob**

`original_email` **= <UnboundField(HiddenField, (), {})>**

class `inspirehep.modules.authors.forms.`**`ExperimentsInlineForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Experiments inline form.

`current` **= <UnboundField(BooleanField, (), {‘widget’: <function currentCheckboxWidget>})>**

`end_year` **= <UnboundField(StringField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.WrappedInput object>,**

`name` **= <UnboundField(StringField, (), {‘autocomplete’: ‘experiment’, ‘widget’: <inspirehep.modules.forms.field_widget**

`start_year` **= <UnboundField(StringField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.WrappedInput objec**

class `inspirehep.modules.authors.forms.`**`InstitutionInlineForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Institution inline form.

`current` **= <UnboundField(BooleanField, (), {‘widget’: <function currentCheckboxWidget>})>**

`emails` **= <UnboundField(FieldList, (<UnboundField(HiddenField, (), {‘label’: ‘’})>,), {‘widget_classes’: ‘hidden-list’})>**

`end_year` **= <UnboundField(StringField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.WrappedInput object>,**

`name` **= <UnboundField(StringField, (), {‘autocomplete’: ‘affiliation’, ‘widget’: <inspirehep.modules.forms.field_widgets.**

`old_emails` **= <UnboundField(FieldList, (<UnboundField(HiddenField, (), {‘label’: ‘’})>,), {‘widget_classes’: ‘hidden-li**

`rank` **= <UnboundField(SelectField, (), {‘default’: ‘rank’, ‘choices’: [(‘rank’, ‘Rank’), (‘SENIOR’, ‘Senior (permanent)’),**

`rank_options` **= [(‘rank’, ‘Rank’), (‘SENIOR’, ‘Senior (permanent)’), (‘JUNIOR’, ‘Junior (leads to Senior)’), (‘STAFF**

`start_year` **= <UnboundField(StringField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.WrappedInput objec**

class `inspirehep.modules.authors.forms.`**`WebpageInlineForm`**`(`**args*, ***kwargs*`)`

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

URL inline form.

webpage = <UnboundField(StringField, (), {'widget_classes': 'form-control', 'widget': <inspirehep.modules.forms.field_

class `inspirehep.modules.authors.forms.`**`WrappedSelect`**(*widget=None*, *wrapper=None*, ***kwargs*)

Bases: `wtforms.widgets.core.Select`

Widget to wrap select input in further markup.

`wrapped_widget` = <wtforms.widgets.core.Select object>

`wrapper` = '<div>%(field)s</div>'

`inspirehep.modules.authors.forms.`**`currentCheckboxWidget`**(*field*, ***kwargs*)
Current institution checkbox widget.

inspirehep.modules.authors.permissions module

inspirehep.modules.authors.utils module

Helper functions for authors.

`inspirehep.modules.authors.utils.`**`bai`**(*name*)

`inspirehep.modules.authors.utils.`**`phonetic_blocks`**(*full_names*, *phonetic_algorithm='nysiis'*)
Create a dictionary of phonetic blocks for a given list of names.

inspirehep.modules.authors.views module

INSPIRE authors views.

`inspirehep.modules.authors.views.`**`holdingpenreview`**(**args*, ***kwargs*)
Deprecated Handler for approval or rejection of new authors in Holding Pen.

`inspirehep.modules.authors.views.`**`new`**()
Deprecated View for INSPIRE author new form.

`inspirehep.modules.authors.views.`**`newreview`**(**args*, ***kwargs*)
Deprecated View for INSPIRE author new form review by a cataloger.

`inspirehep.modules.authors.views.`**`reviewhandler`**(**args*, ***kwargs*)
Deprecated Form handler when a cataloger accepts an author review.

`inspirehep.modules.authors.views.`**`submitnew`**()
Deprecated Form action handler for INSPIRE author new form.

`inspirehep.modules.authors.views.`**`submitupdate`**()
Deprecated Form action handler for INSPIRE author update form.

`inspirehep.modules.authors.views.`**`update`**(*recid*)
Deprecated View for INSPIRE author update form.

`inspirehep.modules.authors.views.`**`validate`**()
Validate form and return validation errors.

FIXME: move to forms module as a generic /validate where we can pass the for class to validate.

Module contents

Authors module.

inspirehep.modules.crossref package

Submodules

inspirehep.modules.crossref.config module

Crossref configuration.

inspirehep.modules.crossref.core module

Crossref core.

`inspirehep.modules.crossref.core.`**`get_json`**`(`*doi*`)`

`inspirehep.modules.crossref.core.`**`get_response`**`(`*crossref_doi*`)`

inspirehep.modules.crossref.ext module

Crossref extension.

class `inspirehep.modules.crossref.ext.`**`InspireCrossref`**`(`*app=None*`)`
Bases: `object`

`init_app``(`*app*`)`

`init_config``(`*app*`)`

inspirehep.modules.crossref.views module

Crossref blueprints.

`inspirehep.modules.crossref.views.`**`search`**`(`**args, **kwargs*`)`

Module contents

INSPIRE Crossref module.

inspirehep.modules.disambiguation package

Subpackages

inspirehep.modules.disambiguation.core package

Subpackages

inspirehep.modules.disambiguation.core.db package

Submodules

inspirehep.modules.disambiguation.core.db.readers module

Disambiguation core DB readers.

`inspirehep.modules.disambiguation.core.db.readers.`**`get_all_curated_signatures`**`()`

Get all curated signatures from the DB.

Walks through all Literature records and collects all signatures that were marked as curated in order to build the training set for `BEARD`.

Yields *dict* – a curated signature.

`inspirehep.modules.disambiguation.core.db.readers.`**`get_all_publications`**`()`

Get all publications from the DB.

Walks through all Literature records and collects all information that will be useful for `BEARD` during training and prediction.

Yields *dict* – a publication.

`inspirehep.modules.disambiguation.core.db.readers.`**`get_all_signatures`**`()`

Get all signatures from the DB.

Walks through all Literature records and collects all signatures in order to build the running set for `BEARD`.

Yields *dict* – a signature.

`inspirehep.modules.disambiguation.core.db.readers.`**`get_signatures_matching_a_phonetic_encod`**

Get all signatures matching a phonetic encoding from ES.

Parameters **`phonetic_encodings`** (*`str`*) – a phonetic encoding.

Yields *dict* – a signature matching the phonetic encoding.

Module contents

Disambiguation core DB.

inspirehep.modules.disambiguation.core.ml package

Submodules

inspirehep.modules.disambiguation.core.ml.models module

Disambiguation core ML models.

class `inspirehep.modules.disambiguation.core.ml.models.`**`DistanceEstimator`**(*ethnicity_estimator*)

Bases: `object`

`fit`()

`load_data`(*signatures_path*, *pairs_path*, *pairs_size*, *publications_path*)

`load_model`(*input_filename*)

`save_model`(*output_filename*)

class `inspirehep.modules.disambiguation.core.ml.models.`**`EthnicityEstimator`**(*C=4.0*)

Bases: `object`

`fit`()

`load_data`(*input_filename*)

`load_model`(*input_filename*)

`predict`(*X*)

`save_model`(*output_filename*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_abstract`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_author_affiliation`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_author_full_name`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_author_other_names`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_coauthors_neighborhood`**(*signature*, *radius=10*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_collaborations`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_first_given_name`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_first_initial`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_keywords`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_second_given_name`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_second_initial`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_title`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`get_topics`**(*signature*)

`inspirehep.modules.disambiguation.core.ml.models.`**`group_by_signature`**(*signatures*)

inspirehep.modules.disambiguation.core.ml.sampling module

Disambiguation core ML sampling.

`inspirehep.modules.disambiguation.core.ml.sampling.`**`sample_signature_pairs`**(*signatures_path*, *clusters_path*, *pairs_size*)

Sample signature pairs to generate less training data.

Since INSPIRE contains ~3M curated signatures it would take too much time to train on all possible pairs, so we sample a subset in such a way that they are representative of the known cluster structure.

This is accomplished in three steps:

1. First we read all the clusters and signatures and build in-memory data structures to perform fast lookups of the id of the cluster to which a signature belongs as well as lookups of the name of the author associated with the signature.

   At the same time we partition the signatures in blocks according to the phonetic encoding of the name. Note that two signatures pointing to two distinct authors might end up in the same block.

2. Then we classify signature pairs that belong to the same block according to whether they belong to same cluster and whether they share the same author name.

   The former is because we want to have both examples of pairs of signatures in the same block pointing to the same author and different authors, while the latter is to avoid oversampling the typical case of signatures with exactly the same author name.

3. Finally we sample from each of the non-empty resulting categories an equal portion of the desired number of pairs. Note that this requires that it must be divisible by 12, the LCM of the possible number of non-empty categories, to make sure that we will sample the same number of pairs from each category.

Yields *dict* – a signature pair.

Module contents

Disambiguation core ML.

Module contents

Disambiguation core.

Submodules

inspirehep.modules.disambiguation.api module

Disambiguation API.

`inspirehep.modules.disambiguation.api.`**`save_curated_signatures_and_input_clusters`**()

Save curated signatures and input clusters to disk.

Saves two files to disk called (by default) `input_clusters.jsonl` and `curated_signatures.jsonl`. The former contains one line per each cluster initially present in INSPIRE, while the latter contains one line per each curated signature that will be used as ground truth by `BEARD`.

`inspirehep.modules.disambiguation.api.`**`save_publications`**`()`

Save publications to disk.

Saves a file to disk called (by default) `publications.jsonl`, which contains one line per record in INSPIRE with information that will be useful for `BEARD` during training and prediction.

`inspirehep.modules.disambiguation.api.`**`save_sampled_pairs`**`()`

Save sampled signature pairs to disk.

Save a file to disk called (by default) `sampled_pairs.jsonl`, which contains one line per each pair of signatures sampled from INSPIRE that will be used by `BEARD` during training.

`inspirehep.modules.disambiguation.api.`**`train_and_save_distance_model`**`()`

Train the distance estimator model and save it to disk.

`inspirehep.modules.disambiguation.api.`**`train_and_save_ethnicity_model`**`()`

Train the ethnicity estimator model and save it to disk.

inspirehep.modules.disambiguation.config module

Disambiguation configuration.

`inspirehep.modules.disambiguation.config.`**`DISAMBIGUATION_SAMPLED_PAIRS_SIZE`** **= 1200000**

The number of signature pairs we use during training.

Since INSPIRE has ~3M curated signatures it would take too much time to train on all possible pairs, so we sample ~1M pairs in such a way that they are representative of the known clusters structure.

Note: It MUST be a multiple of 12 for the reason explained in *`inspirehep.modules.disambiguation.core.ml.sampling`*.

inspirehep.modules.disambiguation.ext module

Disambiguation extension.

class `inspirehep.modules.disambiguation.ext.`**`InspireDisambiguation`**`(`*app=None*`)`

Bases: `object`

`init_app``(`*app*`)`

`init_config``(`*app*`)`

inspirehep.modules.disambiguation.utils module

Disambiguation utils.

`inspirehep.modules.disambiguation.utils.`**`open_file_in_folder`**`(`**args*, ***kwds*`)`

Open a file in a folder, creating the folder if it does not exist.

Module contents

Disambiguation module.

inspirehep.modules.editor package

Submodules

inspirehep.modules.editor.api module

Editor api views.

`inspirehep.modules.editor.api.`**`authorlist_text`**(**args*, ***kw*)
Run authorlist on a piece of text.

`inspirehep.modules.editor.api.`**`check_permission`**(*endpoint*, *pid_value*, ***kwargs*)
Check if logged in user has permission to open the given record.

Used by record-editor on startup.

`inspirehep.modules.editor.api.`**`create_rt_ticket`**(*endpoint*, *pid_value*, ***kwargs*)
View to create an rt ticket

`inspirehep.modules.editor.api.`**`get_linked_refs`**()

`inspirehep.modules.editor.api.`**`get_revision`**(*endpoint*, *pid_value*, ***kwargs*)
Get the revision of given record (uuid)

`inspirehep.modules.editor.api.`**`get_revisions`**(*endpoint*, *pid_value*, ***kwargs*)
Get revisions of given record

`inspirehep.modules.editor.api.`**`get_rt_queues`**(**args*, ***kw*)
View to get all rt queues

`inspirehep.modules.editor.api.`**`get_rt_users`**(**args*, ***kw*)
View to get all rt users

`inspirehep.modules.editor.api.`**`get_tickets_for_record`**(*endpoint*, *pid_value*, ***kwargs*)
View to get rt ticket belongs to given record

`inspirehep.modules.editor.api.`**`manual_merge`**(**args*, ***kw*)
Start a manual merge workflow on two records.

Todo

The following two assertions must be replaced with proper access control checks, as currently any curator who has access to the editor API can merge any two records, even if they are not among those who can see or edit them.

`inspirehep.modules.editor.api.`**`refextract_text`**(**args*, ***kw*)
Run refextract on a piece of text.

`inspirehep.modules.editor.api.`**`refextract_url`**(**args*, ***kw*)
Run refextract on a URL.

`inspirehep.modules.editor.api.`**`resolve_rt_ticket`**(*endpoint*, *pid_value*, ***kwargs*)
View to resolve an rt ticket

`inspirehep.modules.editor.api.`**`revert_to_revision`**(*endpoint*, *pid_value*, ***kwargs*)
Revert given record to given revision

`inspirehep.modules.editor.api.`**`upload_files`**(**args*, ***kw*)

inspirehep.modules.editor.bundles module

Bundle definition for record editor.

inspirehep.modules.editor.permissions module

`inspirehep.modules.editor.permissions.`**`editor_permission`**(*fn*)

inspirehep.modules.editor.views module

Invenio module for editing JSON records.

`inspirehep.modules.editor.views.`**`index`**(**args*, ***kwargs*)
Render base view.

`inspirehep.modules.editor.views.`**`preview`**(**args*, ***kwargs*)
Preview the record being edited.

Module contents

INSPIRE editor.

inspirehep.modules.fixtures package

Submodules

inspirehep.modules.fixtures.cli module

Manage fixtures for INSPIRE site.

inspirehep.modules.fixtures.ext module

Fixtures extension.

class `inspirehep.modules.fixtures.ext.`**`InspireFixtures`**(*app=None*)
Bases: `object`

`init_app`(*app*)

inspirehep.modules.fixtures.files module

Functions for searching ES and returning the results.

`inspirehep.modules.fixtures.files.`**`init_all_storage_paths`**()
Init all storage paths.

`inspirehep.modules.fixtures.files.`**`init_default_storage_path`**()
Init default file store location.

`inspirehep.modules.fixtures.files.`**`init_records_files_storage_path`**(*default=False*)
Init records file store location.

`inspirehep.modules.fixtures.files.`**`init_workflows_storage_path`**`(`*default=False*`)`

Init workflows file store location.

inspirehep.modules.fixtures.users module

Fixtures for users, roles and actions.

`inspirehep.modules.fixtures.users.`**`init_cataloger_permissions`**`()`

`inspirehep.modules.fixtures.users.`**`init_hermes_permissions`**`()`

`inspirehep.modules.fixtures.users.`**`init_jlab_permissions`**`()`

`inspirehep.modules.fixtures.users.`**`init_permissions`**`()`

`inspirehep.modules.fixtures.users.`**`init_roles`**`()`

`inspirehep.modules.fixtures.users.`**`init_superuser_permissions`**`()`

`inspirehep.modules.fixtures.users.`**`init_users`**`()`

Sample users, not to be used in production.

`inspirehep.modules.fixtures.users.`**`init_users_and_permissions`**`()`

Module contents

Fixtures module

inspirehep.modules.forms package

Subpackages

inspirehep.modules.forms.fields package

Submodules

inspirehep.modules.forms.fields.arxiv_id module

class `inspirehep.modules.forms.fields.arxiv_id.`**`ArXivField`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.simple.TextField`

inspirehep.modules.forms.fields.doi module

DOI field.

class `inspirehep.modules.forms.fields.doi.`**`DOIField`**`(`***kwargs*`)`

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.core.StringField`

DOIField.

inspirehep.modules.forms.fields.language module

class `inspirehep.modules.forms.fields.language.`**`LanguageField`**(***kwargs*)

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.core.SelectField`

inspirehep.modules.forms.fields.title module

Deprecated.

class `inspirehep.modules.forms.fields.title.`**`TitleField`**(***kwargs*)

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.core.StringField`

Deprecated.

inspirehep.modules.forms.fields.wtformsext module

This module makes all WTForms fields available in WebDeposit.

This module makes all WTForms fields available in WebDeposit, and ensure that they subclass INSPIREField for added functionality

The code is basically identical to importing all the WTForm fields and for each field make a subclass according to the pattern (using FloatField as an example):

```
class FloatField(INSPIREField, wtforms.FloatField):
    pass
```

class `inspirehep.modules.forms.fields.wtformsext.`**`FormField`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.core.FormField`

Deposition form field.

`flags`

Get flags in form of a proxy.

This proxy accumulats flags stored in this object and all children fields.

`get_flags`(*filter_func=None*)

Get flags.

`json_data`

JSON data property.

`messages`

Message property.

`perform_autocomplete`(*form*, *name*, *term*, *limit=50*)

Run auto-complete method for field.

This method should not be called directly, instead use Form.autocomplete().

`post_process`(*form=None*, *formfields=[]*, *extra_processors=[]*, *submit=False*)

Run post process on each subfield.

Run post process on each subfield as well as extra processors defined on form.

process (*formdata*, *data=<unset value>*)
Preprocess formdata in case we are passed a JSON data structure.

reset_field_data (*exclude=[]*)
Reset the `fields.data` value to that of `field.object_data`.

Usually not called directly, but rather through `Form.reset_field_data()`.

Parameters **exclude** – List of formfield names to exclude.

set_flags (*flags*)
Set flags.

class `inspirehep.modules.forms.fields.wtformsext.`**FieldList** (**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.FieldList`

Deposition field list.

bound_field (*idx*)
Create a bound field for index.

data
Adapted to use self.get_entries() instead of self.entries.

get_entries ()
Get entries.

get_flags (*filter_func=None*)
Get flags.

json_data
JSON data property.

messages
Message.

perform_autocomplete (*form*, *name*, *term*, *limit=50*)
Run auto-complete method for field.

This method should not be called directly, instead use `Form.autocomplete()`.

post_process (*form=None*, *formfields=[]*, *extra_processors=[]*, *submit=False*)
Run post process on each subfield.

Run post process on each subfield as well as extra processors defined on form.

process (**args*, ***kwargs*)
Process.

reset_field_data (*exclude=[]*)
Reset the fields.data value to that of field.object_data.

Usually not called directly, but rather through Form.reset_field_data()

Parameters **exclude** – List of formfield names to exclude.

set_flags (*flags*)
Set flags.

validate (*form*, *extra_validators=()*)
Adapted to use self.get_entries() instead of self.entries.

class `inspirehep.modules.forms.fields.wtformsext.`**DynamicFieldList** (**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.fields.wtformsext.FieldList*

Encapsulate an ordered list of multiple instances of the same field type.

Encapsulate an ordered list of multiple instances of the same field type, keeping data as a list.

Extends WTForm FieldList field to allow dynamic add/remove of enclosed fields.

bound_field(*idx*, *force=False*)
Create a bound subfield for this list.

get_entries()
Filter out empty index entry.

process(*formdata*, *data=<unset value>*)
Adapted from wtforms.FieldList.

Adapted from wtforms.FieldList to allow merging content formdata and draft data properly.

class `inspirehep.modules.forms.fields.wtformsext.`**BooleanField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.BooleanField`

class `inspirehep.modules.forms.fields.wtformsext.`**DateField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.DateField`

class `inspirehep.modules.forms.fields.wtformsext.`**DateTimeField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.DateTimeField`

class `inspirehep.modules.forms.fields.wtformsext.`**DecimalField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.DecimalField`

class `inspirehep.modules.forms.fields.wtformsext.`**Field**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.Field`

class `inspirehep.modules.forms.fields.wtformsext.`**FieldList**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.FieldList`

Deposition field list.

bound_field(*idx*)
Create a bound field for index.

data
Adapted to use self.get_entries() instead of self.entries.

get_entries()
Get entries.

get_flags(*filter_func=None*)
Get flags.

json_data
JSON data property.

messages
Message.

perform_autocomplete(*form*, *name*, *term*, *limit=50*)
Run auto-complete method for field.

This method should not be called directly, instead use `Form.autocomplete()`.

post_process (*form=None*, *formfields=[]*, *extra_processors=[]*, *submit=False*)
Run post process on each subfield.

Run post process on each subfield as well as extra processors defined on form.

process (**args*, ***kwargs*)
Process.

reset_field_data (*exclude=[]*)
Reset the fields.data value to that of field.object_data.

Usually not called directly, but rather through Form.reset_field_data()

Parameters **exclude** – List of formfield names to exclude.

set_flags (*flags*)
Set flags.

validate (*form*, *extra_validators=()*)
Adapted to use self.get_entries() instead of self.entries.

class `inspirehep.modules.forms.fields.wtformsext.`**FileField** (**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.FileField`

class `inspirehep.modules.forms.fields.wtformsext.`**FloatField** (**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.FloatField`

class `inspirehep.modules.forms.fields.wtformsext.`**FormField** (**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.FormField`

Deposition form field.

flags
Get flags in form of a proxy.

This proxy accumulats flags stored in this object and all children fields.

get_flags (*filter_func=None*)
Get flags.

json_data
JSON data property.

messages
Message property.

perform_autocomplete (*form*, *name*, *term*, *limit=50*)
Run auto-complete method for field.

This method should not be called directly, instead use Form.autocomplete().

post_process (*form=None*, *formfields=[]*, *extra_processors=[]*, *submit=False*)
Run post process on each subfield.

Run post process on each subfield as well as extra processors defined on form.

process (*formdata*, *data=<unset value>*)
Preprocess formdata in case we are passed a JSON data structure.

reset_field_data(*exclude=[]*)
Reset the `fields.data` value to that of `field.object_data`.

Usually not called directly, but rather through `Form.reset_field_data()`.

Parameters **exclude** – List of formfield names to exclude.

set_flags(*flags*)
Set flags.

class `inspirehep.modules.forms.fields.wtformsext.`**HiddenField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.HiddenField`

class `inspirehep.modules.forms.fields.wtformsext.`**IntegerField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.IntegerField`

class `inspirehep.modules.forms.fields.wtformsext.`**MultipleFileField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.MultipleFileField`

class `inspirehep.modules.forms.fields.wtformsext.`**PasswordField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.PasswordField`

class `inspirehep.modules.forms.fields.wtformsext.`**RadioField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.RadioField`

class `inspirehep.modules.forms.fields.wtformsext.`**SelectField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.SelectField`

class `inspirehep.modules.forms.fields.wtformsext.`**SelectFieldBase**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.SelectFieldBase`

class `inspirehep.modules.forms.fields.wtformsext.`**SelectMultipleField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.SelectMultipleField`

class `inspirehep.modules.forms.fields.wtformsext.`**StringField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.core.StringField`

class `inspirehep.modules.forms.fields.wtformsext.`**SubmitField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.SubmitField`

class `inspirehep.modules.forms.fields.wtformsext.`**TextAreaField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.TextAreaField`

class `inspirehep.modules.forms.fields.wtformsext.`**TextField**(**args*, ***kwargs*)
Bases: *inspirehep.modules.forms.field_base.INSPIREField*, `wtforms.fields.simple.TextField`

class `inspirehep.modules.forms.fields.wtformsext.`**`TimeField`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.field_base.INSPIREField`*, `wtforms.fields.core.TimeField`

Module contents

Init.

inspirehep.modules.forms.validators package

Submodules

inspirehep.modules.forms.validators.dynamic_fields module

class `inspirehep.modules.forms.validators.dynamic_fields.`**`AuthorsValidation`**(*form*, *field*)

Bases: `object`

Validate authors field.

empty_aff: validates if there are empty names with filled affiliations.

author_names: validates if there is at least one author.

`field_flags` **= ('required',)**

class `inspirehep.modules.forms.validators.dynamic_fields.`**`LessThan`**(*fieldname*, *message=None*)

Bases: `object`

Compares the values of two fields. param fieldname: the name of the other field to compare to. param message: error message to raise in case of a validation error. Can be interpolated with *%(other_label)s* and *%(other_name)s* to provide a more helpful error.

inspirehep.modules.forms.validators.simple_fields module

`inspirehep.modules.forms.validators.simple_fields.`**`already_pending_in_holdingpen_validator`**(

Check if there's a submission in the holdingpen with the same arXiv ID.

`inspirehep.modules.forms.validators.simple_fields.`**`arxiv_id_already_pending_in_holdingpen_va`**

Check if there's a submission in the holdingpen with the same arXiv ID.

`inspirehep.modules.forms.validators.simple_fields.`**`arxiv_syntax_validation`**(*form*, *field*)

Validate ArXiv ID syntax.

`inspirehep.modules.forms.validators.simple_fields.`**`date_validator`**(*form*, *field*)

`inspirehep.modules.forms.validators.simple_fields.`**`does_exist_in_inspirehep`**(*query*, *collections=None*)

Check if there exist an item in the db which satisfies query.

Parameters

- **`query`** – http query to check
- **`collections`** – collections to search in; by default searches in the default collection

`inspirehep.modules.forms.validators.simple_fields.`**`doi_already_pending_in_holdingpen_validat`**

Check if there's a submission in the holdingpen with the same DOI.

`inspirehep.modules.forms.validators.simple_fields.`**`duplicated_arxiv_id_validator`**(*form*, *field*)

Check if a record with the same arXiv ID already exists.

`inspirehep.modules.forms.validators.simple_fields.`**`duplicated_doi_validator`**(*form*, *field*)

Check if a record with the same doi already exists.

`inspirehep.modules.forms.validators.simple_fields.`**`duplicated_orcid_validator`**(*form*, *field*)

Check if a record with the same ORCID already exists.

`inspirehep.modules.forms.validators.simple_fields.`**`duplicated_validator`**(*property_name*, *property_value*)

`inspirehep.modules.forms.validators.simple_fields.`**`inspirehep_duplicated_validator`**(*inspire_quer* *property_name*, *collections=None*)

Check if a record with the same doi already exists.

Needs to be wrapped in a function with proper validator signature.

`inspirehep.modules.forms.validators.simple_fields.`**`no_pdf_validator`**(*form*, *field*)

Validate that the field does not contain a link to a PDF.

`inspirehep.modules.forms.validators.simple_fields.`**`pdf_validator`**(*form*, *field*)

Validate that the field contains a link to a PDF.

`inspirehep.modules.forms.validators.simple_fields.`**`year_validator`**(*form*, *field*)

Validate that the field contains an year in an acceptable range.

Module contents

Submodules

inspirehep.modules.forms.bundles module

Bundles for forms used across INSPIRE.

inspirehep.modules.forms.ext module

Forms extension.

class `inspirehep.modules.forms.ext.`**`InspireForms`**(*app=None*)

Bases: `object`

`init_app`(*app*)

inspirehep.modules.forms.field_base module

Implementation of validators, post-processors and auto-complete functions.

Validators

Following is a short overview over how validators may be defined for fields.

Inline validators (always executed):

```
class MyForm(...):
    myfield = MyField()

    def validate_myfield(form, field):
        raise ValidationError("Message")
```

External validators (always executed):

```
def my_validator(form, field):
    raise ValidationError("Message")

class MyForm(...):
        myfield = MyField(validators=[my_validator])
```

Field defined validators (always executed):

```
class MyField(...):
    # ...
    def pre_validate(self, form):
        raise ValidationError("Message")
```

Default field validators (executed only if external validators are not defined):

```
class MyField(...):
    def __init__(self, **kwargs):
        defaults = dict(validators=[my_validator])
        defaults.update(kwargs)
        super(MyField, self).__init__(**defaults)
```

See http://wtforms.simplecodes.com/docs/1.0.4/validators.html for how to write validators.

Post-processors

Post processors follows the same pattern as validators. You may thus specify:

- Inline processors::

  ```
  Form.post_process_<field>(form, field)
  ```

- External processors::

```
def my_processor(form, field):
    ...
    myfield = MyField(processors=[my_processor])
```

- Field defined processors (please method documentation)::

```
Field.post_process(self, form, extra_processors=[])
```

Auto-complete

- External auto-completion function::

```
def my_autocomplete(form, field, limit=50):
    ...
    myfield = MyField(autocomplete=my_autocomplete)
```

- Field defined auto-completion function (please method documentation)::

```
Field.autocomplete(self, form, limit=50)
```

class `inspirehep.modules.forms.field_base.`**`INSPIREField`**(**args*, ***kwargs*)

Bases: `wtforms.fields.core.Field`

Base field that all webdeposit fields must inherit from.

`add_message`(*msg*, *state=None*)

Add a message.

Parameters

- **`msg`** – The message to set
- **`state`** – State of message; info, warning, error, success.

`messages`

Retrieve field messages.

`perform_autocomplete`(*form*, *name*, *term*, *limit=50*)

Run auto-complete method for field.

This method should not be called directly, instead use Form.autocomplete().

`post_process`(*form=None*, *formfields=[]*, *extra_processors=[]*, *submit=False*)

Post process form before saving.

Usually you can do some of the following tasks in the post processing:

- Set field flags (e.g. self.flags.hidden = True or form.<field>.flags.hidden = True).
- Set messages (e.g. self.messages.append('text') and self.message_state = 'info').
- Set values of other fields (e.g. form.<field>.data = '').

Processors may stop the processing chain by raising StopIteration.

IMPORTANT: By default the method will execute custom post processors defined in the webdeposit_config. If you override the method, be sure to call this method to ensure extra processors are called:

```
super(MyField, self).post_process(
    form, extra_processors=extra_processors
)
```

reset_field_data(*exclude=[]*)
Reset the `fields.data` value to that of `field.object_data`.

Usually not called directly, but rather through Form.reset_field_data()

Parameters **exclude** – List of formfield names to exclude.

set_flags(*flags*)
Set field flags.

inspirehep.modules.forms.field_widgets module

Implement custom field widgets.

class `inspirehep.modules.forms.field_widgets.`**BigIconRadioInput**(*icons={}*, ***kwargs*)
Bases: `wtforms.widgets.core.RadioInput`

Render a single radio button with icon.

This widget is most commonly used in conjunction with InlineListWidget or some other listing, as a single radio button is not very useful.

input_type = 'radio'

class `inspirehep.modules.forms.field_widgets.`**ButtonWidget**(*label=''*, *tooltip=None*, *icon=None*, ***kwargs*)
Bases: `object`

Implement Bootstrap HTML5 button.

class `inspirehep.modules.forms.field_widgets.`**ColumnInput**(*widget=None*, *wrapper=None*, ***kwargs*)
Bases: *`inspirehep.modules.forms.field_widgets.WrappedInput`*

Specialized column wrapped input.

wrapper
Wrapper template with description support.

class `inspirehep.modules.forms.field_widgets.`**DynamicItemWidget**(***kwargs*)
Bases: *`inspirehep.modules.forms.field_widgets.ListItemWidget`*

Render each subfield in a ExtendedListWidget enclosed in a div.

It adds also tag with buttons for sorting and removing the item. I.e. something like:

```
<div><span>"buttons</span>:field</div>
```

render_subfield(*subfield*, ***kwargs*)
Render subfield.

class `inspirehep.modules.forms.field_widgets.`**DynamicListWidget**(***kwargs*)
Bases: *`inspirehep.modules.forms.field_widgets.ExtendedListWidget`*

Render a list of fields as a list of divs.

Additionally adds: * A hidden input to keep track of the last index. * An 'add another' item button.

Each subfield is rendered with DynamicItemWidget, which will add buttons for each item to sort and remove the item.

close_tag(*field*, ***kwargs*)
Render close tag.

icon_add = **'fa fa-plus'**

item_kwargs (*field*, *subfield*)
Return keyword arguments for a field.

item_widget = **<inspirehep.modules.forms.field_widgets.DynamicItemWidget object>**

open_tag (*field*, ***kwargs*)
Render open tag.

class `inspirehep.modules.forms.field_widgets.`**ExtendedListWidget** (*html_tag='ul'*, *item_widget=None*, *class_=None*)

Bases: `object`

Render a list of fields as a *ul*, *ol* or *div* list.

This is used for fields which encapsulate a list of other fields as subfields. The widget will try to iterate the field to get access to the subfields and call them to render them.

The *item_widget* decide how subfields are rendered, and usually just provide a thin wrapper around the subfields render method. E.g. ExtendedListWidget renders the ul-tag, while the ListItemWidget renders each li-tag. The content of the li-tag is rendered by the subfield's widget.

close_tag (*field*, ***kwargs*)
Render close tag.

item_kwargs (*field*, *subfield*)
Return keyword arguments for a field.

item_widget = **<inspirehep.modules.forms.field_widgets.ListItemWidget object>**

open_tag (*field*, ***kwargs*)
Render open tag.

class `inspirehep.modules.forms.field_widgets.`**ItemWidget**

Bases: `object`

Render each subfield without additional markup around the subfield.

class `inspirehep.modules.forms.field_widgets.`**ListItemWidget** (*html_tag='li'*, *with_label=True*, *prefix_label=True*, *class_=None*)

Bases: `inspirehep.modules.forms.field_widgets.ItemWidget`

Render each subfield in a ExtendedListWidget as a list element.

If *with_label* is set, the fields label will be rendered. If *prefix_label* is set, the label will be prefixed, otherwise it will be suffixed.

close_tag (*subfield*, ***kwargs*)
Return close tag.

open_tag (*subfield*, ***kwargs*)
Return open tag.

render_subfield (*subfield*, ***kwargs*)
Render subfield.

class `inspirehep.modules.forms.field_widgets.`**TagInput** (*input_type=None*)

Bases: `wtforms.widgets.core.Input`

Implement tag input widget.

input_type = ‘hidden’

class `inspirehep.modules.forms.field_widgets.`**WrappedInput**(*widget=None*, *wrapper=None*, ***kwargs*)

Bases: `wtforms.widgets.core.Input`

Widget to wrap text input in further markup.

wrapped_widget = <wtforms.widgets.core.TextInput object>

wrapper = ‘<div>%(field)s</div>’

inspirehep.modules.forms.filter_utils module

WTForm filters implementation.

Filters can be applied to incoming form data, after process_formdata() has run.

See more information on: http://wtforms.simplecodes.com/docs/1.0.4/fields.html#wtforms.fields.Field

`inspirehep.modules.forms.filter_utils.`**clean_empty_list**(*value*)
Created to clean a list produced by Bootstrap multi-select.

`inspirehep.modules.forms.filter_utils.`**strip_prefixes**(**prefixes*)
Return a filter function that removes leading prefixes from a string.

`inspirehep.modules.forms.filter_utils.`**strip_string**(*value*)
Remove leading and trailing spaces from string.

inspirehep.modules.forms.form module

`inspirehep.modules.forms.form.`**CFG_FIELD_FLAGS = [’hidden’, ‘disabled’, ‘touched’]**
List of WTForm field flags to be saved in draft.

`inspirehep.modules.forms.form.`**CFG_GROUPS_META = {‘classes’: None, ‘indication’: None, ‘description’: None, ‘icon**
Default group metadata.

class `inspirehep.modules.forms.form.`**DataExporter**(*filter_func=None*)
Bases: *inspirehep.modules.forms.form.FormVisitor*

Visitor to export form data into dictionary supporting filtering and key renaming.

Usage:: form = ... visitor = DataExporter(filter_func=lambda f: not f.flags.disabled) visitor.visit(form)

Given e.g. the following form:

```
class MyForm(INSPIREForm):
    title = TextField(export_key='my_title')
    notes = TextAreaField()
    authors = FieldList(FormField(AuthorForm))
```

the visitor will export a dictionary similar to:

```
{'my_title': ..., 'notes': ..., authors: [{...}, ...], }
```

visit_field(*field*)

visit_fieldlist(*fieldlist*)

visit_formfield(*formfield*)

class `inspirehep.modules.forms.form.`**`FormVisitor`**

Bases: `object`

Generic form visitor to iterate over all fields in a form. See DataExporter for example how to export all data.

`visit`(*form_or_field*)

`visit_field`(*field*)

`visit_fieldlist`(*fieldlist*)

`visit_form`(*form*)

`visit_formfield`(*formfield*)

class `inspirehep.modules.forms.form.`**`INSPIREForm`**(**args*, ***kwargs*)

Bases: `wtforms.form.Form`

Generic WebDeposit Form class.

`get_groups`()

Get a list of the (group metadata, list of fields)-tuples. The last element of the list has no group metadata (i.e. None), and contains the list of fields not assigned to any group.

`get_template`()

Get template to render this form. Define a data member *template* to customize which template to use. By default, it will render the template *deposit/run.html*

`json_data`

Return form data in a format suitable for the standard JSON encoder. Return form data in a format suitable for the standard JSON encoder, by calling Field.json_data() on each field if it exists, otherwise is uses the value of Field.data.

`messages`

Return a dictionary of form messages.

`post_process`(*form=None*, *formfields=[]*, *submit=False*)

Run form post-processing.

Run form post-processing by calling *post_process* on each field, passing any extra *Form.post_process_<fieldname>* processors to the field.

If *formfields* are specified, only the given fields' processors will be run (which may touch all fields of the form). The post processing allows the form to alter other fields in the form, via e.g. contacting external services (e.g a DOI field could retrieve title, authors from CrossRef/DataCite).

inspirehep.modules.forms.utils module

Forms utilities.

`inspirehep.modules.forms.utils.`**`filter_empty_elements`**(*recjson*, *list_fields*)

Filter empty fields.

`inspirehep.modules.forms.utils.`**`filter_empty_helper`**(*keys=None*)

Remove empty elements from a list.

inspirehep.modules.forms.validation_utils module

Validation functions.

class `inspirehep.modules.forms.validation_utils.`**`DOISyntaxValidator`**(*message=None*)

Bases: `object`

DOI syntax validator.

`inspirehep.modules.forms.validation_utils.`**`ORCIDValidator`**(*form*, *field*)

Validate that the given ORCID exists.

class `inspirehep.modules.forms.validation_utils.`**`RegexpStopValidator`**(*regex*, *flags=0*, *message=None*)

Bases: `object`

Validates the field against a user provided regexp.

Parameters

- **`regex`** – The regular expression string to use. Can also be a compiled regular expression pattern.
- **`flags`** – The regexp flags to use, for example re.IGNORECASE. Ignored if *regex* is not a string.
- **`message`** – Error message to raise in case of a validation error.

inspirehep.modules.forms.views module

Module contents

Forms module.

inspirehep.modules.hal package

Subpackages

inspirehep.modules.hal.core package

Submodules

inspirehep.modules.hal.core.sword module

HAL SWORD core.

class `inspirehep.modules.hal.core.sword.`**`HttpLib2LayerIgnoreCert`**(*cache_dir*)

Bases: `sword2.http_layer.HttpLib2Layer`

`inspirehep.modules.hal.core.sword.`**`create`**(*tei*, *doc_file=None*)

Create a record on HAL using the SWORD2 protocol.

`inspirehep.modules.hal.core.sword.`**`update`**(*tei*, *hal_id*, *doc_file=None*)

Update a record on HAL using the SWORD2 protocol.

inspirehep.modules.hal.core.tei module

HAL TEI core.

`inspirehep.modules.hal.core.tei.`**`convert_to_tei`**`(`*record*`)`

Return the record formatted in XML+TEI per HAL's specification.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the record formatted in XML+TEI.

Return type string

Examples

```
>>> record = get_db_record('lit', 1407506)
>>> convert_to_tei(record)
<?xml version="1.0" encoding="UTF-8"?>
...
```

Module contents

HAL Core.

Submodules

inspirehep.modules.hal.bulk_push module

IMPORTANT This script is a copy/paste of: https://github.com/inspirehep/inspire-next/issues/2629

It is unreliable and absolutely unmaintainable. It will be refactored with this user story: https://its.cern.ch/jira/browse/INSPIR-249

To be run with: $ /usr/bin/time -v inspirehep hal push

`inspirehep.modules.hal.bulk_push.`**`run`**`(`*username*, *password*, *limit*, *yield_amt*`)`

inspirehep.modules.hal.cli module

inspirehep.modules.hal.config module

HAL configuration.

`inspirehep.modules.hal.config.`**`HAL_COL_IRI`** **= 'https://api-preprod.archives-ouvertes.fr/sword/hal'**

IRI used by the SWORD protocol when creating a new record on HAL.

Note: Use this to send records to their staging instance. To send records to their production instance use the same IRI without `-preprod`.

`inspirehep.modules.hal.config.`**`HAL_DOMAIN_MAPPING`** **= {'Instrumentation': 'phys.phys.phys-ins-det', 'Data Anal**

Mapping used when converting from INSPIRE categories to HAL domains.

`inspirehep.modules.hal.config.`**`HAL_EDIT_IRI`** **= 'https://api-preprod.archives-ouvertes.fr/sword/'**

IRI used by the SWORD protocol when updating an existing record on HAL.

Note: Use this to update records on their staging instance. To update records on their production instance use the same IRI without `-preprod`.

`inspirehep.modules.hal.config.`**`HAL_IGNORE_CERTIFICATES`** **= False**

Whether to check certificates when connecting to HAL.

`inspirehep.modules.hal.config.`**`HAL_USER_NAME`** **= 'hal_user_name'**

Name of the INSPIRE user on HAL.

Note: Its real value is stored in `tbag`. In particular `QA_HAL_USER_NAME` contains the value to use for their staging instance, while `PROD_HAL_USER_NAME` contains the value to use for their production instance.

`inspirehep.modules.hal.config.`**`HAL_USER_PASS`** **= 'hal_user_pass'**

Password of the INSPIRE user on HAL.

Note: Its real value is stored in `tbag`. In particular `QA_HAL_USER_PASS` contains the value to use for their staging instance, while `PROD_HAL_USER_PASS` contains the value to use for their production instance.

inspirehep.modules.hal.ext module

HAL extension.

class `inspirehep.modules.hal.ext.`**`InspireHAL`**(*app=None*)

Bases: `object`

`init_app`(*app*)

`init_config`(*app*)

inspirehep.modules.hal.tasks module

HAL tasks.

(task)`inspirehep.modules.hal.tasks.`**`hal_push`**

Run a hal push.

`inspirehep.modules.hal.tasks.`**`send_hal_push_start_email`**(*mailing_list*)

`inspirehep.modules.hal.tasks.`**`send_hal_push_summary_email`**(*mailing_list, total, ok, now, ko, attached_files=None*)

Sends a nice email with the summary of the hal push.

inspirehep.modules.hal.utils module

HAL utils.

`inspirehep.modules.hal.utils.`**`get_authors`**`(`*record*`)`

Return the authors of a record.

Queries the Institution records linked from the authors affiliations to add, whenever it exists, the HAL identifier of the institution to the affiliation.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the authors of the record.

Return type list(dict)

Examples

```
>>> record = {
...     'authors': [
...         'affiliations': [
...             {
...                 'record': {
...                     '$ref': 'http://localhost:5000/api/institutions/902725',
...                 }
...             },
...         ],
...     ],
... }
>>> authors = get_authors(record)
>>> authors[0]['hal_id']
'300037'
```

`inspirehep.modules.hal.utils.`**`get_conference_city`**`(`*record*`)`

Return the first city of a Conference record.

Parameters **`record`** (`InspireRecord`) – a Conference record.

Returns the first city of the Conference record.

Return type string

Examples

```
>>> record = {'address': [{'cities': ['Tokyo']}]}
>>> get_conference_city(record)
'Tokyo'
```

`inspirehep.modules.hal.utils.`**`get_conference_country`**`(`*record*`)`

Return the first country of a Conference record.

Parameters **`record`** (`InspireRecord`) – a Conference record.

Returns the first country of the Conference record.

Return type string

Examples

```
>>> record = {'address': [{'country_code': 'JP'}]}
>>> get_conference_country(record)
'jp'
```

`inspirehep.modules.hal.utils.`**`get_conference_end_date`**`(`*record*`)`

Return the closing date of a conference record.

Parameters **`record`** (`InspireRecord`) – a Conference record.

Returns the closing date of the Conference record.

Return type string

Examples

```
>>> record = {'closing_date': '1999-11-19'}
>>> get_conference_end_date(record)
'1999-11-19'
```

`inspirehep.modules.hal.utils.`**`get_conference_record`**`(`*record*, *default=None*`)`

Return the first Conference record associated with a record.

Queries the database to fetch the first Conference record referenced in the `publication_info` of the record.

Parameters

- **`record`** (`InspireRecord`) – a record.
- **`default`** – value to be returned if no conference record present/found

Returns the first Conference record associated with the record.

Return type *InspireRecord*

Examples

```
>>> record = {
...     'publication_info': [
...         {
...             'conference_record': {
...                 '$ref': '/api/conferences/972464',
...             },
...         },
...     ],
... }
>>> conference_record = get_conference_record(record)
>>> conference_record['control_number']
972464
```

`inspirehep.modules.hal.utils.`**`get_conference_start_date`**`(`*record*`)`

Return the opening date of a conference record.

Parameters **`record`** (`InspireRecord`) – a Conference record.

Returns the opening date of the Conference record.

Return type string

Examples

```
>>> record = {'opening_date': '1999-11-16'}
>>> get_conference__start_date(record)
'1999-11-16'
```

`inspirehep.modules.hal.utils.`**`get_conference_title`**(*record*, *default=’‘*)

Return the first title of a Conference record.

Parameters **`record`** (`InspireRecord`) – a Conference record.

Returns the first title of the Conference record.

Return type string

Examples

```
>>> record = {'titles': [{'title': 'Workshop on Neutrino Physics'}]}
>>> get_conference_title(record)
'Workshop on Neutrino Physics'
```

`inspirehep.modules.hal.utils.`**`get_divulgation`**(*record*)

Return 1 if a record is intended for the general public, 0 otherwise.

Parameters **`record`** (`InspireRecord`) – a record.

Returns 1 if the record is intended for the general public, 0 otherwise.

Return type int

Examples

```
>>> get_divulgation({'publication_type': ['introductory']})
1
```

`inspirehep.modules.hal.utils.`**`get_document_types`**(*record*)

Return all document types of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns all document types of the record.

Return type list(str)

Examples

```
>>> get_document_types({'document_type': ['article']})
['article']
```

`inspirehep.modules.hal.utils.`**`get_doi`**(*record*)

Return the first DOI of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first DOI of the record.

Return type string

Examples

```
>>> get_doi({'dois': [{'value': '10.1016/0029-5582(61)90469-2'}]})
'10.1016/0029-5582(61)90469-2'
```

`inspirehep.modules.hal.utils.`**`get_domains`**(*record*)

Return the HAL domains of a record.

Uses the mapping in the configuration to convert all INSPIRE categories to the corresponding HAL domains.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the HAL domains of the record.

Return type list(str)

Examples

```
>>> record = {'inspire_categories': [{'term': 'Experiment-HEP'}]}
>>> get_domains(record)
['phys.hexp']
```

`inspirehep.modules.hal.utils.`**`get_inspire_id`**(*record*)

Return the INSPIRE id of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the INSPIRE id of the record.

Return type int

Examples

```
>>> get_inspire_id({'control_number': 1507156})
1507156
```

`inspirehep.modules.hal.utils.`**`get_journal_issue`**(*record*)

Return the issue of the journal a record was published into.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the issue of the journal the record was published into.

Return type string

Examples

```
>>> record = {
...     'publication_info': [
...         {'journal_issue': '5'},
...     ],
... }
```

```
>>> get_journal_issue(record)
'5'
```

`inspirehep.modules.hal.utils.`**`get_journal_title`**`(`*record*`)`

Return the title of the journal a record was published into.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the title of the journal the record was published into.

Return type string

Examples

```
>>> record = {
...     'publication_info': [
...         {'journal_title': 'Phys.Part.Nucl.Lett.'},
...     ],
... }
>>> get_journal_title(record)
'Phys.Part.Nucl.Lett.'
```

`inspirehep.modules.hal.utils.`**`get_journal_volume`**`(`*record*`)`

Return the volume of the journal a record was published into.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the volume of the journal the record was published into.

Return type string

Examples

```
>>> record = {
...     'publication_info': [
...         {'journal_volume': 'D94'},
...     ],
... }
>>> get_journal_volume(record)
'D94'
```

`inspirehep.modules.hal.utils.`**`get_language`**`(`*record*`)`

Return the first language of a record.

If it is not specified in the record we assume that the language is English, so we return `'en'`.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first language of the record.

Return type string

Examples

```
>>> get_language({'languages': ['it']})
'it'
```

`inspirehep.modules.hal.utils.`**`get_page_artid`**(*record, separator='-'*)

Return the page range or the article id of a record.

Parameters

- **`record`** (`InspireRecord`) – a record
- **`separator`** (*`basestring`*) – optional page range symbol, defaults to a single dash

Returns the page range or the article id of the record.

Return type string

Examples

```
>>> record = {
...     'publication_info': [
...         {'artid': '054021'},
...     ],
... }
>>> get_page_artid(record)
'054021'
```

`inspirehep.modules.hal.utils.`**`get_page_artid_for_publication_info`**(*publication_info, separator*)

Return the page range or the article id of a publication_info entry.

Parameters

- **`publication_info`** (*`dict`*) – a publication_info field entry of a record
- **`separator`** (*`basestring`*) – optional page range symbol, defaults to a single dash

Returns the page range or the article id of the record.

Return type string

Examples

```
>>> publication_info = {'artid': '054021'}
>>> get_page_artid(publication_info)
'054021'
```

`inspirehep.modules.hal.utils.`**`get_peer_reviewed`**(*record*)

Return 1 if a record is peer reviewed, 0 otherwise.

Parameters **`record`** (`InspireRecord`) – a record.

Returns 1 if the record is peer reviewed, 0 otherwise.

Return type int

Examples

```
>>> get_peer_reviewed({'refereed': True})
1
```

`inspirehep.modules.hal.utils.`**`get_publication_date`**`(`*record*`)`

Return the date in which a record was published.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the date in which the record was published.

Return type string

Examples

```
>>> get_publication_date({'publication_info': [{'year': 2017}]})
'2017'
```

`inspirehep.modules.hal.utils.`**`is_published`**`(`*record*`)`

Return if a record is published.

We say that a record is published if it is citeable, which means that it has enough information in a `publication_info`, or if we know its DOI and a `journal_title`, which means it is in press.

Parameters **`record`** (`InspireRecord`) – a record.

Returns whether the record is published.

Return type bool

Examples

```
>>> record = {
...     'dois': [
...         {'value': '10.1016/0029-5582(61)90469-2'},
...     ],
...     'publication_info': [
...         {'journal_title': 'Nucl.Phys.'},
...     ],
... }
>>> is_published(record)
True
```

inspirehep.modules.hal.views module

HAL views.

Module contents

HAL module.

This module converts INSPIRE literature records to the XML+TEI format supported by Hyper Articles en Ligne (HAL), a French open archive of scholarly documents.

The Jinja2 Python library is used to convert records into a HAL-supported format, after which the Python SWORD client posts these records to the HAL SWORD API.

inspirehep.modules.literaturesuggest package

Submodules

inspirehep.modules.literaturesuggest.bundles module

Bundles for author forms.

inspirehep.modules.literaturesuggest.ext module

LiteratureSuggest extension.

class `inspirehep.modules.literaturesuggest.ext.`**`InspireLiteratureSuggest`**(*app=None*)

Bases: `object`

`init_app`(*app*)

inspirehep.modules.literaturesuggest.forms module

Contains forms related to INSPIRE Literature suggestion.

class `inspirehep.modules.literaturesuggest.forms.`**`AuthorInlineForm`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Author inline form.

`affiliation` **= <UnboundField(TextField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.ColumnInput object>**

`name` **= <UnboundField(TextField, (), {‘widget’: <inspirehep.modules.forms.field_widgets.ColumnInput object>, ‘export_**

class `inspirehep.modules.literaturesuggest.forms.`**`CheckboxButton`**(*msg=’‘*)

Bases: `object`

Checkbox button.

class `inspirehep.modules.literaturesuggest.forms.`**`LiteratureForm`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Literature form fields.

`abstract` **= <UnboundField(TextAreaField, (), {‘default’: ‘’, ‘label’: ‘Abstract’, ‘export_key’: ‘abstract’, ‘widget_classe**

`additional_url` **= <UnboundField(TextField, (), {‘label’: ‘Link to additional information (e.g. abstract)’, ‘validators’:**

`arxiv_id` **= <UnboundField(ArXivField, (), {‘label’: ‘arXiv ID’, ‘export_key’: ‘arxiv_id’, ‘validators’: [<function arxiv_**

`authors` **= <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class ‘inspirehep.modules.literaturesugges**

`book_title` **= <UnboundField(TextField, (), {‘label’: ‘Book Title’, ‘widget_classes’: ‘form-control chapter-related’})>**

`categories_arXiv` **= <UnboundField(TextField, (), {‘widget’: <wtforms.widgets.core.HiddenInput object>, ‘export_ke**

`collaboration` **= <UnboundField(TextField, (), {‘label’: ‘Collaboration’, ‘export_key’: ‘collaboration’, ‘widget_classes**

`conf_name` **= <UnboundField(TextField, (), {‘autocomplete’: ‘conference’, ‘label’: ‘Conference Information’, ‘placehold**

`conference_id` **= <UnboundField(TextField, (), {‘widget’: <wtforms.widgets.core.HiddenInput object>, ‘export_key’:**

`defense_date` **= <UnboundField(TextField, (), {‘label’: ‘Date of Defense’, ‘validators’: [<function date_validator>], ‘de**

`degree_type` **= <UnboundField(SelectField, (), {‘default’: ‘phd’, ‘label’: ‘Degree Type’, ‘widget_classes’: ‘form-control**

`doi` = <UnboundField(DOIField, (), {'processors': [], 'export_key': 'doi', 'label': 'DOI', 'validators': [<inspirehep.modu

`end_page` = <UnboundField(TextField, (), {'placeholder': 'End page of the chapter', 'widget_classes': 'form-control cha

`experiment` = <UnboundField(TextField, (), {'autocomplete': 'experiment', 'placeholder': 'Start typing for suggestions

`extra_comments` = <UnboundField(TextAreaField, (), {'label': 'Comments', 'description': 'Any extra comments relate

`field_sizes` = {'thesis_date': 'col-xs-12 col-md-4', 'start_page': 'col-xs-12 col-md-3', 'degree_type': 'col-xs-12 col-md

`find_book` = <UnboundField(TextField, (), {'label': 'Find Book', 'placeholder': 'Start typing for suggestions', 'descripti

`groups` = [('Import information', ['arxiv_id', 'doi', 'import_buttons']), ('Document Type', ['type_of_doc']), ('Links', ['u

`import_buttons` = <UnboundField(SubmitField, (), {'widget': <function import_buttons_widget>, 'label': ' '})>

`institution` = <UnboundField(TextField, (), {'autocomplete': 'affiliation', 'label': 'Institution', 'placeholder': 'Start t

`issue` = <UnboundField(TextField, (), {'label': 'Issue', 'widget_classes': 'form-control article-related'})>

`journal_title` = <UnboundField(TextField, (), {'autocomplete': 'journal', 'label': 'Journal Title', 'placeholder': 'Star

`language` = <UnboundField(LanguageField, (), {'default': 'en', 'label': 'Language', 'export_key': 'language', 'choices':

`language_choices` = [('zh', u'Chinese'), ('en', u'English'), ('fr', u'French'), ('de', u'German'), ('it', u'Italian'), ('ja',

`license_url` = <UnboundField(TextField, (), {'widget': <wtforms.widgets.core.HiddenInput object>, 'export_key': 'lic

`nonpublic_note` = <UnboundField(TextAreaField, (), {'widget': <function wrap_nonpublic_note>, 'label': 'Proceedin

`note` = <UnboundField(TextAreaField, (), {'widget': <wtforms.widgets.core.HiddenInput object>, 'export_key': 'note'})

`other_language` = <UnboundField(LanguageField, (), {'label': 'Other Language', 'widget_classes': 'form-control', 'e

`other_language_choices` = [(u'ab', u'Abkhazian'), (u'aa', u'Afar'), (u'af', u'Afrikaans'), (u'ak', u'Akan'), (u'sq', u

`page_range_article_id` = <UnboundField(TextField, (), {'label': 'Page Range/Article ID', 'description': 'e.g. 1-100'

`parent_book` = <UnboundField(TextField, (), {'widget': <wtforms.widgets.core.HiddenInput object>})>

`preprint_created` = <UnboundField(TextField, (), {'widget': <wtforms.widgets.core.HiddenInput object>, 'export_k

`publication_date` = <UnboundField(TextField, (), {'label': 'Publication Date', 'widget_classes': 'form-control book-

`publication_place` = <UnboundField(TextField, (), {'label': 'Publication Place', 'widget_classes': 'form-control boo

`publisher_name` = <UnboundField(TextField, (), {'label': 'Publisher', 'widget_classes': 'form-control book-related'})>

`references` = <UnboundField(TextAreaField, (), {'label': 'References', 'description': 'Please paste the references in pl

`report_numbers` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class 'inspirehep.modules.litera

`series_title` = <UnboundField(TextField, (), {'autocomplete': 'journal', 'label': 'Series Title', 'widget_classes': 'for

`series_volume` = <UnboundField(TextField, (), {'label': 'Volume', 'widget_classes': 'form-control book-related'})>

`start_page` = <UnboundField(TextField, (), {'placeholder': 'Start page of the chapter', 'widget_classes': 'form-control

`subject` = <UnboundField(SelectMultipleField, (), {'widget_classes': 'form-control', 'label': 'Subject', 'export_key': 's

`supervisors` = <UnboundField(DynamicFieldList, (<UnboundField(FormField, (<class 'inspirehep.modules.literature

`thesis_date` = <UnboundField(TextField, (), {'label': 'Date of Submission', 'validators': [<function date_validator>],

`title` = <UnboundField(TitleField, (), {'widget_classes': 'form-control', 'label': 'Title', 'export_key': 'title', 'validators

`title_arXiv` = <UnboundField(TitleField, (), {'widget': <wtforms.widgets.core.HiddenInput object>, 'export_key': 'ti

`title_crossref` = <UnboundField(TitleField, (), {'widget': <wtforms.widgets.core.HiddenInput object>, 'export_key

`title_translation` = <UnboundField(TitleField, (), {'label': 'Translated Title', 'export_key': 'title_translation', 'des

type_of_doc = <UnboundField(SelectField, (), {'default': 'article', 'choices': [('article', 'Article/Conference paper'), ('

types_of_doc = [('article', 'Article/Conference paper'), ('thesis', 'Thesis'), ('book', 'Book'), ('chapter', 'Book chapter'

url = <UnboundField(TextField, (), {'label': 'Link to PDF', 'validators': [<function pdf_validator>], 'placeholder': 'http

volume = <UnboundField(TextField, (), {'label': 'Volume', 'widget_classes': 'form-control article-related'})>

year = <UnboundField(TextField, (), {'widget_classes': 'form-control article-related', 'label': 'Year', 'validators': [<fun

class `inspirehep.modules.literaturesuggest.forms.`**`ReportNumberInlineForm`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Repor number inline form.

report_number = <UnboundField(TextField, (), {'widget': <inspirehep.modules.forms.field_widgets.ColumnInput obje

class `inspirehep.modules.literaturesuggest.forms.`**`UnorderedDynamicItemWidget`**(***kwargs*)

Bases: *`inspirehep.modules.forms.field_widgets.DynamicItemWidget`*

class `inspirehep.modules.literaturesuggest.forms.`**`UnsortedDynamicListWidget`**(***kwargs*)

Bases: *`inspirehep.modules.forms.field_widgets.DynamicListWidget`*

class `inspirehep.modules.literaturesuggest.forms.`**`UrlInlineForm`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.forms.form.INSPIREForm`*

Url inline form.

url = <UnboundField(TextField, (), {'export_key': 'full_url', 'widget': <inspirehep.modules.forms.field_widgets.Column

`inspirehep.modules.literaturesuggest.forms.`**`import_buttons_widget`**(*field*, ***dummy_kwargs*)

Button for import data and skip.

`inspirehep.modules.literaturesuggest.forms.`**`importdata_button`**(*field*, ***dummy_kwargs*)

Import data button.

`inspirehep.modules.literaturesuggest.forms.`**`journal_title_kb_mapper`**(*val*)

Return object ready to autocomplete journal titles.

`inspirehep.modules.literaturesuggest.forms.`**`radiochoice_buttons`**(*field*, ***dummy_kwargs*)

Radio choice buttons.

`inspirehep.modules.literaturesuggest.forms.`**`skip_importdata`**(*field*, ***dummy_kwargs*)

Skip Import data button.

`inspirehep.modules.literaturesuggest.forms.`**`wrap_nonpublic_note`**(*field*, ***dummy_kwargs*)

Proceedings box with tooltip.

inspirehep.modules.literaturesuggest.normalizers module

`inspirehep.modules.literaturesuggest.normalizers.`**`check_book_existence`**(*title*)

`inspirehep.modules.literaturesuggest.normalizers.`**`check_journal_existence`**(*title*)

`inspirehep.modules.literaturesuggest.normalizers.`**`find_book_id`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`get_user_email`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`get_user_orcid`**(*obj*, *formdata*) `inspirehep.modules.literaturesuggest.normalizers.`**`normalize_formdata`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`normalize_journal_title`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`normalize_provided_doi`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`remove_english_language`**(*obj*, *formdata*)

`inspirehep.modules.literaturesuggest.normalizers.`**`split_page_range_article_id`**(*obj*, *formdata*)

inspirehep.modules.literaturesuggest.tasks module

`inspirehep.modules.literaturesuggest.tasks.`**`curation_ticket_context`**(*user*, *obj*)

`inspirehep.modules.literaturesuggest.tasks.`**`curation_ticket_needed`**(**args*, ***kwargs*)
Check if the a curation ticket is needed.

`inspirehep.modules.literaturesuggest.tasks.`**`formdata_to_model`**(*obj*, *formdata*)
Manipulate form data to match literature data model.

`inspirehep.modules.literaturesuggest.tasks.`**`new_ticket_context`**(*user*, *obj*)
Context for literature new tickets.

`inspirehep.modules.literaturesuggest.tasks.`**`reply_ticket_context`**(*user*, *obj*)
Context for literature replies.

inspirehep.modules.literaturesuggest.views module

INSPIRE Literature suggestion blueprint.

`inspirehep.modules.literaturesuggest.views.`**`create`**(**args*, ***kwargs*)
View for INSPIRE suggestion create form.

`inspirehep.modules.literaturesuggest.views.`**`submit`**()
Get form data and start workflow.

`inspirehep.modules.literaturesuggest.views.`**`success`**()
Render success template for the user.

`inspirehep.modules.literaturesuggest.views.`**`success_book_parent`**()
Render success template for the user.

`inspirehep.modules.literaturesuggest.views.`**`validate`**()
Validate form and return validation errors.

FIXME: move to forms module as a generic /validate where we can pass the for class to validate.

Module contents

LiteratureSuggest module.

inspirehep.modules.migrator package

Subpackages

inspirehep.modules.migrator.serializers package

Subpackages

inspirehep.modules.migrator.serializers.schemas package

Submodules

inspirehep.modules.migrator.serializers.schemas.json module

Marshmallow JSON error schema.

class `inspirehep.modules.migrator.serializers.schemas.json.`**`Error`**(*extra=None, only=None, exclude=(), prefix=u'‘, strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

Schema for mirror records with errors.

class **`Meta`**

`strict` **= True**

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.migrator.serializers.schemas.json.`**`ErrorList`**(*extra=None, only=None, exclude=(), prefix=u'‘, strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

Schema for list of mirror records with errors.

class **`Meta`**

`strict` = **True**

`opts` = **<marshmallow.schema.SchemaOpts object>**

Module contents

Migrator schemas.

Module contents

Migrator serializers.

Submodules

inspirehep.modules.migrator.cli module

Manage migrator from INSPIRE legacy instance.

`inspirehep.modules.migrator.cli.`**`halt_if_debug_mode`**(*force*)

inspirehep.modules.migrator.dumper module

Migrator dumper.

`inspirehep.modules.migrator.dumper.`**`marshmallow_dumper`**(*schema_class*)

Marshmallow dumper.

`inspirehep.modules.migrator.dumper.`**`migrator_error_list_dumper`**(*results, many=False*)

inspirehep.modules.migrator.ext module

Migrator extension.

class `inspirehep.modules.migrator.ext.`**`InspireMigrator`**(*app=None*)

Bases: `object`

`init_app`(*app*)

inspirehep.modules.migrator.models module

Models for Migrator.

class `inspirehep.modules.migrator.models.`**`LegacyRecordsMirror`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`

`collection`

`error`

classmethod `from_marcxml`(*raw_record*)

Create an instance from a MARCXML record.

The record must have a `001` tag containing the recid, otherwise it raises a ValueError.

`last_updated`

`marcxml`

marcxml column wrapper to compress/decompress on the fly.

`re_recid` = <_sre.SRE_Pattern object at 0x6073dd0>

`recid`

`valid`

`inspirehep.modules.migrator.models.`**`timestamp_before_update`**(*mapper*, *connection*, *target*)

Update *last_updated* property with current time on *before_update* event.

inspirehep.modules.migrator.permissions module

inspirehep.modules.migrator.tasks module

Manage migration from INSPIRE legacy instance.

`inspirehep.modules.migrator.tasks.`**`chunker`**(*iterable*, *chunksize=100*)

(task)`inspirehep.modules.migrator.tasks.`**`continuous_migration`**

Task to continuously migrate what is pushed up by Legacy.

`inspirehep.modules.migrator.tasks.`**`disable_orcid_push`**(*task_function*)

Temporarily disable ORCID push

Decorator to temporarily disable ORCID push while a given task is running, and only for that task. Takes care of restoring the previous state in case of errors or when the task is finished. This does not interfere with other tasks, firstly because of ditto, secondly because configuration is only changed within the worker's process (thus doesn't affect parallel tasks).

`inspirehep.modules.migrator.tasks.`**`insert_into_mirror`**(*raw_records*)

`inspirehep.modules.migrator.tasks.`**`migrate_and_insert_record`**(*raw_record*, *skip_files=False*)

Migrate a record and insert it if valid, or log otherwise.

`inspirehep.modules.migrator.tasks.`**`migrate_from_file`**(*source*, *wait_for_results=False*)

`inspirehep.modules.migrator.tasks.`**`migrate_from_mirror`**(*also_migrate=None*, *wait_for_results=False*, *skip_files=None*)

Migrate legacy records from the local mirror.

By default, only the records that have not been migrated yet are migrated.

Parameters

- **`also_migrate`** (*`Optional[string]`*) – if set to `'broken'`, also broken records will be migrated. If set to `'all'`, all records will be migrated.

- **`skip_files`** (*`Optional[bool]`*) – flag indicating whether the files in the record metadata should be copied over from legacy and attach to the record. If None, the corresponding setting is read from the configuration.
- **`wait_for_results`** (*`bool`*) – flag indicating whether the task should wait for the migration to finish (if True) or fire and forget the migration tasks (if False).

(task)`inspirehep.modules.migrator.tasks.`**`migrate_recids_from_mirror`**

`inspirehep.modules.migrator.tasks.`**`migrate_record_from_legacy`**(*recid*)

`inspirehep.modules.migrator.tasks.`**`migrate_record_from_mirror`**(*prod_record*, *skip_files=False*)

Migrate a mirrored legacy record into an Inspire record.

Parameters

- **`prod_record`** (`LegacyRecordsMirror`) – the mirrored record to migrate.
- **`skip_files`** (*`bool`*) – flag indicating whether the files in the record metadata should be copied over from legacy and attach to the record.

Returns the migrated record metadata, which is also inserted into the database.

Return type dict

`inspirehep.modules.migrator.tasks.`**`populate_mirror_from_file`**(*source*)

`inspirehep.modules.migrator.tasks.`**`read_file`**(*source*)

`inspirehep.modules.migrator.tasks.`**`split_blob`**(*blob*)

Split the blob using <record.*?>.*?</record> as pattern.

`inspirehep.modules.migrator.tasks.`**`split_stream`**(*stream*)

Split the stream using <record.*?>.*?</record> as pattern.

This operates line by line in order not to load the entire file in memory.

inspirehep.modules.migrator.utils module

Migrator utils.

`inspirehep.modules.migrator.utils.`**`get_collection`**(*marc_record*)

`inspirehep.modules.migrator.utils.`**`get_collection_from_marcxml`**(*marcxml*)

inspirehep.modules.migrator.views module

class `inspirehep.modules.migrator.views.`**`MigratorErrorListResource`**

Bases: `flask.views.MethodView`

Return a list of errors belonging to invalid mirror records.

`decorators` **= [<flask_principal.IdentityContext object>]**

`get`()

`methods` **= ['GET']**

`inspirehep.modules.migrator.views.`**`migrator_error_list_resource`**(**args*, ***kw*)

Return a list of errors belonging to invalid mirror records.

Module contents

INSPIRE migrator module.

inspirehep.modules.orcid package

Submodules

inspirehep.modules.orcid.builder module

Builds an ORCID work record.

class `inspirehep.modules.orcid.builder.`**`OrcidBuilder`**

Bases: `object`

Class used to build ORCID-compatible work records in JSON.

`add_arxiv`(*value*, *relationship=None*)

Add arXiv identifier to the record.

Parameters

- **`value`** (*`string`*) – the identifier itself
- **`relationship`** (*`string`*) – either "part-of" or "self", optional, see *OrcidBuilder._make_external_id_field*

`add_citation`(*_type*, *value*)

Add a citation string.

Parameters

- **`_type`** (*`string`*) – citation type, one of: https://git.io/vdKXv#L313-L321
- **`value`** (*`string`*) – citation string for the provided citation type

`add_contributor`(*credit_name*, *role='author'*, *orcid=None*, *email=None*)

Adds a contributor entry to the record.

Parameters

- **`credit_name`** (*`string`*) – contributor's name
- **`orcid`** (*`string`*) – ORCID identifier string
- **`role`** (*`string`*) – role, see *OrcidBuilder._make_contributor_field*
- **`email`** (*`string`*) – contributor's email address

`add_country`(*country_code*)

Set country if the ORCID record.

Parameters **`country_code`** (*`string`*) – ISO ALPHA-2 country code

`add_doi`(*value*, *relationship=None*)

Add DOI to the record.

Parameters

- **`value`** (*`string`*) – the identifier itself
- **`relationship`** (*`string`*) – either "part-of" or "self", optional, see *OrcidBuilder._make_external_id_field*

add_external_id(*type*, *value*, *url=None*, *relationship=None*)

Add external identifier to the record.

Parameters

- **type** (*string*) – type of external ID (doi, etc.)
- **value** (*string*) – the identifier itself
- **url** (*string*) – URL for the resource
- **relationship** (*string*) – either "part-of" or "self", optional, see *OrcidBuilder._make_external_id_field*

add_journal_title(*journal_title*)

Set title of the publication containing the record.

Parameters **journal_title** (*string*) – Title of publication containing the record.

After ORCID v2.0 schema (https://git.io/vdKXv#L268-L280): "The title of the publication or group under which the work was published. - If a journal, include the journal title of the work. - If a book chapter, use the book title. - If a translation or a manual, use the series title. - If a dictionary entry, use the dictionary title. - If a conference poster, abstract or paper, use the conference name."

add_publication_date(*partial_date*)

Set publication date field.

Parameters **partial_date** (*inspire_utils.date.PartialDate*) – publication date

add_recid(*value*, *url*, *relationship=None*)

Add Inspire recid to the record.

Parameters

- **value** (*string*) – the recid.
- **url** (*string*) – url to the Inspire record.
- **relationship** (*string*) – either "part-of" or "self", optional, see *OrcidBuilder._make_external_id_field*

add_title(*title*, *subtitle=None*, *translated_title=None*)

Set a title of the work, and optionaly a subtitle.

Parameters

- **title** (*string*) – title of the work
- **subtitle** (*string*) – subtitle of the work
- **translated_title** (*Tuple[string, string]*) – tuple consiting of the translated title and its language code

add_type(*work_type*)

Add a work type.

Parameters **work_type** (*string*) – type of work, see: https://git.io/vdKXv#L118-L155

add_url(*url*)

Add a url.

Parameters **url** (*string*) – alternative url of the record

get_xml()
Get an XML record.

Returns ORCID work record compatible with API v2.0

Return type lxml.etree._Element

set_put_code(*put_code*)
Set the put-code of an ORCID record, to update existing one.

Parameters **put_code** (`string | integer`) – a number, being a put code

set_visibility(*visibility*)
Set visibility setting on ORCID.

Can only be set during record creation.

Parameters **visibility** (`string`) – one of (private, limited, registered-only, public), see https://git.io/vdKXt#L904-L937

inspirehep.modules.orcid.cache module

class `inspirehep.modules.orcid.cache.`**OrcidCache**(*orcid*, *recid*)
Bases: `object`

delete_work_putcode
Delete the putcode for the given (orcid, recid).

has_work_content_changed
True if the work content has changed compared to the cached version.

Parameters **inspire_record** (`InspireRecord`) – InspireRecord instance. If provided, the hash for the record content is re-computed.

read_work_putcode
Read the putcode for the given (orcid, recid).

redis

write_work_putcode
Write the putcode and the hash for the given (orcid, recid).

Parameters

- **putcode** (`string`) – the putcode used to push the record to ORCID.
- **inspire_record** (`InspireRecord`) – InspireRecord instance. If provided, the hash for the record content is re-computed.

Raises `ValueError` – when the putcode is empty.

inspirehep.modules.orcid.converter module

Handle conversion from INSPIRE records to ORCID.

class `inspirehep.modules.orcid.converter.`**ExternalIdentifier**(*type*, *value*)
Bases: `tuple`

type
Alias for field number 0

`value`
Alias for field number 1

class `inspirehep.modules.orcid.converter.`**`OrcidConverter`**(*record*, *url_pattern*, *put_code=None*, *visibility=None*)

Bases: `object`

Coverter for the Orcid format.

`INSPIRE_DOCTYPE_TO_ORCID_TYPE` **= {‘note’: ‘other’, ‘proceedings’: ‘edited-book’, ‘book’: ‘book’, ‘book chapter’:**

`INSPIRE_TO_ORCID_ROLES_MAP` **= {‘supervisor’: None, ‘editor’: ‘editor’, ‘author’: ‘author’}**

`added_external_identifiers`

`arxiv_eprint`
Get arXiv ID of a record.

`bibtex_citation`

`book_series_title`
Get record’s book series title.

`conference_country`
Get conference record country.

`conference_title`
Get record’s conference title.

`doi`
Get DOI of a record.

`get_xml`
Create an ORCID XML representation of the record.

Parameters **`do_add_bibtex_citation`** (*`bool`*) – True to add BibTeX-serialized record

Returns ORCID XML work record

Return type lxml.etree._Element

`journal_title`
Get record’s journal title.

`orcid_for_inspire_author`(*author*)
ORCID identifier for an INSPIRE author field.

Parameters **`author`** (*`dict`*) – an author field from INSPIRE literature record

Returns ORCID identifier of an author, if available

Return type string

`orcid_role_for_inspire_author`(*author*)
ORCID role for an INSPIRE author field.

Parameters **`author`** (*`dict`*) – an author field from INSPIRE literature record

Returns ORCID role of a person

Return type string

`orcid_work_type`
Get record’s ORCID work type.

`publication_date`
(Partial) date of publication.

Returns publication date

Return type partial_date (inspire_utils.date.PartialDate)

`recid`
Get INSPIRE record ID.

`subtitle`
Get record subtitle.

`title`
Get record title.

`title_translation`
Translated title.

Returns translated title and the language code of the translation, if available

Return type Tuple[string, string]

inspirehep.modules.orcid.domain_models module

class `inspirehep.modules.orcid.domain_models.`**`OrcidPusher`**(*orcid*, *recid*, *oauth_token*, *do_fail_if_duplicated_identifier=False*)

Bases: `object`

`push`

inspirehep.modules.orcid.exceptions module

exception `inspirehep.modules.orcid.exceptions.`**`BaseOrcidPusherException`**(**args*, ***kwargs*)

Bases: `exceptions.Exception`

exception `inspirehep.modules.orcid.exceptions.`**`DuplicatedExternalIdentifierPusherException`**(**arg*
***kw*

Bases: *`inspirehep.modules.orcid.exceptions.BaseOrcidPusherException`*

The underneath Orcid service client response raised DuplicatedExternalIdentifierPusherException. We checked for the clashing work, pushed it and repeated the original operation which failed again.

exception `inspirehep.modules.orcid.exceptions.`**`InputDataInvalidException`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.orcid.exceptions.BaseOrcidPusherException`*

The underneath Orcid service client response included an error related to input data like TokenInvalidException, OrcidNotFoundException, PutcodeNotFoundPutException. Note: that re-trying would not help in this case.

exception `inspirehep.modules.orcid.exceptions.`**`PutcodeNotFoundInCacheAfterCachingAllPutcodes`**(***
***

Bases: *`inspirehep.modules.orcid.exceptions.BaseOrcidPusherException`*

No putcode was found in cache after having cached all author putcodes.

exception `inspirehep.modules.orcid.exceptions.`**`PutcodeNotFoundInOrcidException`**(**args*, ***kwargs*)

Bases: *`inspirehep.modules.orcid.exceptions.BaseOrcidPusherException`*

No putcode was found in ORCID API.

exception `inspirehep.modules.orcid.exceptions.`**`RecordNotFoundException`**(**args*, ***kwargs*)

Bases: `inspirehep.modules.orcid.exceptions.BaseOrcidPusherException`

inspirehep.modules.orcid.ext module

Search extension.

class `inspirehep.modules.orcid.ext.`**`InspireOrcid`**(*app=None*)

Bases: `object`

`init_app`(*app*)

`init_config`(*app*)

inspirehep.modules.orcid.putcode_getter module

class `inspirehep.modules.orcid.putcode_getter.`**`OrcidPutcodeGetter`**(*orcid*, *oauth_token*)

Bases: `object`

`get_all_inspire_putcodes_and_recids_iter`()

Query ORCID api and get all the Inspire putcodes for the given ORCID.

`get_putcodes_and_recids_by_identifiers_iter`(*identifiers*)

Yield putcode and recid for each work matched by the external identifiers. Note: external identifiers of type ‘other-id’ are skipped.

Parameters **`identifiers`** (`List[inspirehep.modules.orcid.converter.ExternalIdentifier]`) – list af all external identifiers added after the xml conversion.

inspirehep.modules.orcid.tasks module

Manage ORCID OAUTH token migration from INSPIRE legacy instance.

exception `inspirehep.modules.orcid.tasks.`**`RemoteTokenOrcidMismatch`**(*user*, *orcids*)

Bases: `exceptions.Exception`

(task)`inspirehep.modules.orcid.tasks.`**`import_legacy_orcid_tokens`**

Celery task to import OAUTH ORCID tokens from legacy. Note: bind=True for compatibility with @time_execution.

`inspirehep.modules.orcid.tasks.`**`legacy_orcid_arrays`**()

Generator to fetch token data from redis.

Note: this function consumes the queue populated by the legacy tasklet: inspire/bibtasklets/bst_orcidsync.py

Yields *list* – user data in the form of [orcid, token, email, name]

(task)`inspirehep.modules.orcid.tasks.`**`orcid_push`**

Celery task to push a record to ORCID.

Parameters

- **`self`** (*`celery.Task`*) – the task
- **`orcid`** (*`String`*) – an orcid identifier.
- **`rec_id`** (*`Int`*) – inspire record’s id to push to ORCID.

- **`oauth_token`** (*`String`*) – orcid token.
- **`kwargs_to_pusher`** (*`Dict`*) – extra kwargs to pass to the pusher object.

inspirehep.modules.orcid.utils module

ORCID utils.

class `inspirehep.modules.orcid.utils.`**`RetryMixin`**(**args*, ***kwargs*)

Bases: `celery.app.task.Task`

`request`

`retry`(**args*, ***kwargs*)

`inspirehep.modules.orcid.utils.`**`account_setup`**(*remote*, *token*, *resp*)

Perform additional setup after user have been logged in.

This is a modified version of invenio_oauthclient.contrib.orcid.account_setup that stores additional metadata.

Parameters

- **`remote`** – The remote application.
- **`token`** – The token value.
- **`resp`** – The response.

`inspirehep.modules.orcid.utils.`**`apply_celery_task_with_retry`**(*task_func*, *args=None*, *kwargs=None*, *max_retries=5*, *countdown=10*, *time_limit=None*)

When executing a (bind=True) task synchronously (with *mytask.apply()* or just calling it as a function *mytask()*) the *self.retry()* does not work, but the original exception is raised (without any retry) so you "lose" the exception management logic written in the task code.

This function overcome such limitation. Example:

Celery task: @shared_task(bind=True) def normalize_name_task(self, first_name, last_name, nick_name=' '):

try: result = ... network call ...

except RequestException as exc:

exception = None

raise self.retry(max_retries=3, countdown=5, exc=exception)

return result

Call the task sync with retry: result = apply_celery_task_with_retry(

normalize_name_task, args=('John', 'Doe'), kwargs=dict(nick_name='Dee'), max_retries=2, countdown=5*60, time_limit=2*60*60

)

Note: it assumes that @shared_task is the first (the one on top) decorator for the Celery task.

Parameters

- **`task_func`** – Celery task function to be run.
- **`args`** – the positional arguments to pass on to the task.

- **`kwargs`** – the keyword arguments to pass on to the task.
- **`max_retries`** – maximum number of retries before raising MaxRetriesExceededError.
- **`countdown`** – hard time limit for each attempt. If the last attempt It can be a callable, eg:

  backoff = lambda retry_count: 2 ** (retry_count + 1) apply_celery_task_with_retry(..., countdown=backoff)

- **`time_limit`** – hard time limit for each single attempt in seconds. If the last attempt fails because of the time limit, raises TimeLimitExceeded.

Returns: what the task_func returns.

`inspirehep.modules.orcid.utils.`**`get_literature_recids_for_orcid`**(*orcid*)

Return the Literature recids that were claimed by an ORCiD.

We record the fact that the Author record X has claimed the Literature record Y by storing in Y an author object with a `$ref` pointing to X and the key `curated_relation` set to `True`. Therefore this method first searches the DB for the Author records for the one containing the given ORCiD, and then uses its recid to search in ES for the Literature records that satisfy the above property.

Parameters **`orcid`** (*`str`*) – the ORCiD.

Returns the recids of the Literature records that were claimed by that ORCiD.

Return type list(int)

`inspirehep.modules.orcid.utils.`**`get_orcids_for_push`**(*record*)

Obtain the ORCIDs associated to the list of authors in the Literature record.

The ORCIDs are looked up both in the `ids` of the `authors` and in the Author records that have claimed the paper.

Parameters **`record`** (*`dict`*) – metadata from a Literature record

Returns all ORCIDs associated to these authors

Return type Iterator[str]

`inspirehep.modules.orcid.utils.`**`get_push_access_tokens`**(*orcids*)

Get the remote tokens for the given ORCIDs.

Parameters **`orcids`** (*`List[str]`*) – ORCIDs to get the tokens for.

Returns pairs of (ORCID, access_token), for ORCIDs having a token. These are similar to named tuples, in that the values can be retrieved by index or by attribute, respectively `id` and `access_token`.

Return type sqlalchemy.util._collections.result

`inspirehep.modules.orcid.utils.`**`log_service_response`**(*logger*, *response*, *extra_message=None*)

Module contents

ORCID integration module.

inspirehep.modules.pidstore package

Subpackages

inspirehep.modules.pidstore.providers package

Submodules

inspirehep.modules.pidstore.providers.recid module

INSPIRE Record Id provider.

class `inspirehep.modules.pidstore.providers.recid.`**`InspireRecordIdProvider`**(*pid*)

Bases: `invenio_pidstore.providers.base.BaseProvider`

Record identifier provider.

classmethod **`create`**(*object_type=None*, *object_uuid=None*, ***kwargs*)

Create a new record identifier.

`default_status` **= 'K'**

Record IDs are by default registered immediately.

`pid_provider` **= None**

Provider name. The provider name is not recorded in the PID since the provider does not provide any additional features besides creation of record ids.

`pid_type` **= None**

Type of persistent identifier.

Module contents

Submodules

inspirehep.modules.pidstore.fetchers module

Persistent identifier minters.

class `inspirehep.modules.pidstore.fetchers.`**`FetchedPID`**(*provider*, *pid_type*, *pid_value*)

Bases: `tuple`

`pid_type`

Alias for field number 1

`pid_value`

Alias for field number 2

`provider`

Alias for field number 0

`inspirehep.modules.pidstore.fetchers.`**`inspire_recid_fetcher`**(*record_uuid*, *data*)

Fetch a record's identifiers.

inspirehep.modules.pidstore.minters module

Persistent identifier minters.

`inspirehep.modules.pidstore.minters.`**`inspire_recid_minter`**(*record_uuid*, *data*)
Mint record identifiers.

inspirehep.modules.pidstore.utils module

PIDStore utils.

`inspirehep.modules.pidstore.utils.`**`get_endpoint_from_pid_type`**(*pid_type*)
Return the endpoint corresponding to a `pid_type`.

`inspirehep.modules.pidstore.utils.`**`get_pid_type_from_endpoint`**(*endpoint*)
Return the `pid_type` corresponding to an endpoint.

`inspirehep.modules.pidstore.utils.`**`get_pid_type_from_schema`**(*schema*)
Return the `pid_type` corresponding to a schema URL.

The schema name corresponds to the `endpoint` in all cases except for Literature records. This implementation exploits this by falling back to `get_pid_type_from_endpoint`.

`inspirehep.modules.pidstore.utils.`**`get_pid_types_from_endpoints`**()

Module contents

inspirehep.modules.records package

Subpackages

inspirehep.modules.records.mappings package

Subpackages

inspirehep.modules.records.mappings.v5 package

Module contents

Module contents

inspirehep.modules.records.serializers package

Subpackages

inspirehep.modules.records.serializers.fields package

Submodules

inspirehep.modules.records.serializers.fields.list_with_limit module

class `inspirehep.modules.records.serializers.fields.list_with_limit.`**`ListWithLimit`**(*cls_or_instance*
***kwargs*)

Bases: `marshmallow.fields.List`

inspirehep.modules.records.serializers.fields.nested_without_empty_objects module

class `inspirehep.modules.records.serializers.fields.nested_without_empty_objects.`**`NestedWithou`**

Bases: `marshmallow.fields.Nested`

Module contents

inspirehep.modules.records.serializers.schemas package

Subpackages

inspirehep.modules.records.serializers.schemas.json package

Subpackages

inspirehep.modules.records.serializers.schemas.json.authors package

Subpackages

inspirehep.modules.records.serializers.schemas.json.authors.common package

Submodules

inspirehep.modules.records.serializers.schemas.json.authors.common.position module

class `inspirehep.modules.records.serializers.schemas.json.authors.common.position.`**`PositionSch`**

Bases: `marshmallow.schema.Schema`

`get_display_date`(*data*)

`opts` = **<marshmallow.schema.SchemaOpts object>**

Module contents

Module contents

class `inspirehep.modules.records.serializers.schemas.json.authors.`**`AuthorsMetadataSchemaV1`**(*extr only exclud prefix= stric man context= load dum partial=*

Bases: `marshmallow.schema.Schema`

static **`get_facet_author_name`**(*data*)

static **`get_should_display_positions`**(*data*)

`opts` **= <marshmallow.schema.SchemaOpts object>**

`strip_empty`(*data*)

class `inspirehep.modules.records.serializers.schemas.json.authors.`**`AuthorsRecordSchemaJSONUIV1`**

Bases: *`inspirehep.modules.records.serializers.schemas.base.JSONSchemaUIV1`*

`opts` **= <marshmallow.schema.SchemaOpts object>**

inspirehep.modules.records.serializers.schemas.json.literature package

Subpackages

inspirehep.modules.records.serializers.schemas.json.literature.common package

Submodules

inspirehep.modules.records.serializers.schemas.json.literature.common.accelerator_experiment module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.accelerator_exper`

Bases: `marshmallow.schema.Schema`

`get_control_numbers_to_resolved_experiments_map`(*data*)

`get_name`(*item*)

`get_resolved_record_or_experiment`(*experiment_records_map*, *experiment*)

`opts` = **<marshmallow.schema.SchemaOpts object>**

`resolve_experiment_records`(*data*, *many*)

inspirehep.modules.records.serializers.schemas.json.literature.common.author module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.author.`**`AuthorSche`**

Bases: `marshmallow.schema.Schema`

filter(*data*)

get_first_name(*data*)

get_last_name(*data*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.citation_item module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.citation_item.`**Cit**

Bases: `marshmallow.schema.Schema`

opts = <marshmallow.schema.SchemaOpts object>

strip_empty(*data*)

inspirehep.modules.records.serializers.schemas.json.literature.common.collaboration module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.collaboration.`**Co**

Bases: `marshmallow.schema.Schema`

REGEX_COLLABORATIONS_WITH_SUFFIX = <_sre.SRE_Pattern object>

filter(*data*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.collaboration_with_suffix module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.collaboration_wit`

Bases: *inspirehep.modules.records.serializers.schemas.json.literature.common.collaboration.CollaborationSchemaV1*

filter(*data*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.conference_info_item module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.conference_info_i`

Bases: `marshmallow.schema.Schema`

opts = <marshmallow.schema.SchemaOpts object>

resolve_conference_record_as_root(*pub_info_item*)

inspirehep.modules.records.serializers.schemas.json.literature.common.doi module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.doi.`**`DOISchemaV1`**(*e*
o
e
c
p
f
s
n
c
t
l
a
p
t

Bases: `marshmallow.schema.Schema`

`filter`(*data*, *many*)

`opts` = **<marshmallow.schema.SchemaOpts object>**

static **`remove_duplicate_doi_values`**(*dois*)

inspirehep.modules.records.serializers.schemas.json.literature.common.external_system_identifier module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.external_system_i`

Bases: `marshmallow.schema.Schema`

`filter`(*data*, *many*)

`get_link_for_kekscan_schema`(*external_system_id_value*)

`get_url_link`(*data*)

`get_url_name`(*data*)

`is_missing_url_name_or_link`(*external_system_id*)

`opts` = **<marshmallow.schema.SchemaOpts object>**

`schema_to_url_link_prefix_map` = **{‘hal’: ‘https://hal.archives-ouvertes.fr/’, ‘ads’: ‘http://adsabs.harvard.edu/abs**

`schema_to_url_name_map` = **{‘hal’: ‘HAL Archives Ouvertes’, ‘ads’: ‘ADS Abstract Service’, ‘cds’: ‘CERN Documen**

take_first_id_foreach_url_name(*external_system_ids*)

take_ids_that_have_all_fields(*external_system_ids*)

inspirehep.modules.records.serializers.schemas.json.literature.common.isbn module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.isbn.`**IsbnSchemaV1**

Bases: `marshmallow.schema.Schema`

get_formatted_medium(*isbn*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.publication_info_item module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.publication_info_`

Bases: `marshmallow.schema.Schema`

empty_if_display_display_fields_missing(*data*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.reference_item module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.reference_item.`**Re**

Bases: `marshmallow.schema.Schema`

`filter_references`(*data*, *many*)

`force_each_collaboration_to_be_object`(*data*)

`get_arxiv_eprints`(*data*)

`get_dois`(*data*)

`get_misc`(*data*)

`get_reference_or_linked_reference_with_label`(*data*, *reference_record*)

`get_reference_record_id`(*data*)

`get_resolved_reference`(*data*, *reference_records*)

`get_resolved_references_by_control_number`(*data*)

`get_titles`(*data*)

`opts` = <marshmallow.schema.SchemaOpts object>

`strip_empty`(*data*)

inspirehep.modules.records.serializers.schemas.json.literature.common.supervisor module

class `inspirehep.modules.records.serializers.schemas.json.literature.common.supervisor.`**Superv**

Bases: *`inspirehep.modules.records.serializers.schemas.json.literature.`*

common.author.AuthorSchemaV1

filter(*data*)

opts = <marshmallow.schema.SchemaOpts object>

inspirehep.modules.records.serializers.schemas.json.literature.common.thesis_info module

class inspirehep.modules.records.serializers.schemas.json.literature.common.thesis_info.**Thesi**

Bases: marshmallow.schema.Schema

get_formatted_date(*info*)

get_formatted_defense_date(*info*)

get_formatted_degree_type(*info*)

opts = <marshmallow.schema.SchemaOpts object>

Module contents

Module contents

class inspirehep.modules.records.serializers.schemas.json.literature.**LiteratureAuthorsSchemaJ**

Bases: *inspirehep.modules.records.serializers.schemas.base.JSONSchemaUIV1*

Schema for literature authors.

opts = <marshmallow.schema.SchemaOpts object>

class `inspirehep.modules.records.serializers.schemas.json.literature.`**`LiteratureRecordSchemaJS`**

Bases: *`inspirehep.modules.records.serializers.schemas.base.JSONSchemaUIV1`*

Schema for record UI.

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.records.serializers.schemas.json.literature.`**`LiteratureReferencesSche`**

Bases: *`inspirehep.modules.records.serializers.schemas.base.JSONSchemaUIV1`*

Schema for references.

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.records.serializers.schemas.json.literature.`**`MetadataAuthorsSchemaV1`**

Bases: `marshmallow.schema.Schema`

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.records.serializers.schemas.json.literature.`**`MetadataReferencesSchema`**

Bases: `marshmallow.schema.Schema`

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.records.serializers.schemas.json.literature.`**`RecordMetadataSchemaV1`**(*e*
o
e
c
p
f
s
m
c
t
l
a
p
t

Bases: `marshmallow.schema.Schema`

`get_formatted_date`(*data*)

static **`get_len_or_missing`**(*maybe_none_list*)

`get_number_of_authors`(*data*)

`get_number_of_references`(*data*)

`opts` **= <marshmallow.schema.SchemaOpts object>**

`strip_empty`(*data*)

Module contents

inspirehep.modules.records.serializers.schemas.latex package

Module contents

class `inspirehep.modules.records.serializers.schemas.latex.`**`LatexSchema`**(*extra=None, only=None, exclude=(), prefix=u'', strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

`get_author_names`(*data*)

`get_collaborations`(*data*)

`get_current_date`(*data*)

`get_publication_info`(*data*)

`get_texkey`(*data*)

`opts` = **<marshmallow.schema.SchemaOpts object>**

Submodules

inspirehep.modules.records.serializers.schemas.base module

Schema for parsing literature records.

class `inspirehep.modules.records.serializers.schemas.base.`**`JSONSchemaUIV1`**(*extra=None, only=None, exclude=(), prefix=u'', strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

JSON schema.

`opts` **= <marshmallow.schema.SchemaOpts object>**

class `inspirehep.modules.records.serializers.schemas.base.`**`PybtexSchema`**

Bases: `object`

`load`(*record*)

Deserialize an INSPIRE record into a Pybtex Entity.

Takes an INSPIRE record and converts it to a `pybtex.database.Entity`. Special treatment is applied to authors, which are expressed using `pybtex.database.Person` if they are real persons, and passed like other fields if they are corporate authors. Human-authors supersede corporate authors.

Parameters **`record`** (*`dict`*) – literature record from API

Returns Pybtex entity

Return type pybtex.database.Entity

Module contents

inspirehep.modules.records.serializers.writers package

Submodules

inspirehep.modules.records.serializers.writers.bibtex_writer module

class `inspirehep.modules.records.serializers.writers.bibtex_writer.`**`BibtexWriter`**(*encoding=None*)

Bases: `pybtex.database.output.bibtex.Writer`

Formats bibtex, but limits total number of authors displayed.

Module contents

Plugins for pybtex to generate other bibliography styles.

Submodules

inspirehep.modules.records.serializers.config module

`inspirehep.modules.records.serializers.config.`**`COMMON_FIELDS_FOR_ENTRIES`** **= ['key', 'SLACcitation', '**

BibTeX fields shared among all bibtex entries.

`inspirehep.modules.records.serializers.config.`**`FIELDS_FOR_ENTRY_TYPE`** **= {'inbook': ['chapter', 'publish**

Specific fields for a given bibtex entry.

Note: Since we're trying to match as many as possible it doesn't matter whether they're mandatory or optional

`inspirehep.modules.records.serializers.config.`**`MAX_AUTHORS_BEFORE_ET_AL`** **= 10**

Maximum number of authors to be displayed without truncation.

Note: For more than `MAX_AUTHORS_BEFORE_ET_AL` only the first author should be displayed and a suitable truncation method is applied.

inspirehep.modules.records.serializers.fields_export module

`inspirehep.modules.records.serializers.fields_export.`**`bibtex_document_type`**(*doc_type*, *obj*)

Return the BibTeX entry type.

Maps the INSPIRE `document_type` to a BibTeX entry type. Also checks `thesis_info.degree_type` in case it's a thesis, as it stores the information on which kind of thesis we're dealing with.

Parameters

- **`doc_type`** (*`text_type`*) – INSPIRE document type.
- **`obj`** (*`dict`*) – literature record.

Returns bibtex document type for the given INSPIRE entry.

Return type text_type

`inspirehep.modules.records.serializers.fields_export.`**`bibtex_type_and_fields`**(*data*)

Return a BibTeX doc type and fields needed to be included in a BibTeX record.

Parameters **`data`** (*`dict`*) – inspire record

Returns bibtex document type and fields

Return type tuple

`inspirehep.modules.records.serializers.fields_export.`**`extractor`**(*field*)

`inspirehep.modules.records.serializers.fields_export.`**`get_address`**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**`get_arxiv_prefix`**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**`get_author`**(*data*, *doc_type*)

Get corporate author of a record.

Note: Only used to generate author field if corporate_author is the author.

`inspirehep.modules.records.serializers.fields_export.`**`get_authors_with_role`**(*authors*, *role*)

Extract names of people from an authors field given their roles.

Parameters

- **`authors`** – authors field of the record.
- **`role`** – string specifying the role 'author', 'editor', etc.

Returns of names of people

Return type list of text_type

`inspirehep.modules.records.serializers.fields_export.`**`get_best_publication_info`**(*data*)

Return the most comprehensive publication_info entry.

Parameters **data** (*dict*) – inspire record

Returns a publication_info entry or default if not found any

Return type dict

`inspirehep.modules.records.serializers.fields_export.`**get_booktitle**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_collaboration**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_country_name_by_code**(*code*, *default=None*)

Return a country name string from a country code.

Parameters

- **code** (*str*) – country code in INSPIRE 2 letter format based on ISO 3166-1 alpha-2
- **default** – value to be returned if no country of a given code exists

Returns name of a country, or `default` if no such country.

Return type text_type

`inspirehep.modules.records.serializers.fields_export.`**get_date**(*data*, *doc_type*)

Return a publication/thesis/imprint date.

Parameters

- **data** (*dict*) – INSPIRE literature record to be serialized
- **doc_type** (*text_type*) – BibTeX document type, as reported by *bibtex_document_type*

Returns publication date for a record.

Return type PartialDate

`inspirehep.modules.records.serializers.fields_export.`**get_doi**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_edition**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_eprint**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_isbn**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_journal**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_month**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_note**(*data*, *doc_type*)

Write and addendum/errata information to the BibTeX note field.

Traverse publication_info looking for erratum and addendum in *publication_info.material* field and build a string of references to those publication entries.

Returns formatted list of the errata and addenda available for a given record

Return type string

`inspirehep.modules.records.serializers.fields_export.`**get_number**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_pages**(*data*, *doc_type*) `inspirehep.modules.records.serializers.fields_export.`**get_primary_class**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_publisher**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_report_number**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_school**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_series**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_title**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_type**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_url**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_volume**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**get_year**(*data*, *doc_type*)

`inspirehep.modules.records.serializers.fields_export.`**make_extractor**()

Create a function store decorator.

Creates a decorator function that is used to collect extractor functions. They are put in a dictionary with the field they extract as keys. An extractor function is a function which returns a BibTeX field value given an inspire record and a document type.

Returns a decorator with a store for pre-processing/extracting functions.

Return type function

inspirehep.modules.records.serializers.json_literature module

Marshmallow based JSON serializer for records.

class `inspirehep.modules.records.serializers.json_literature.`**FacetsJSONUISerializer**(*schema_clas* *‘invenio_records_* **kwargs)

Bases: `invenio_records_rest.serializers.json.JSONSerializer`

JSON brief format serializer.

serialize_facets(*query_results*, **kwargs)

class `inspirehep.modules.records.serializers.json_literature.`**LiteratureCitationsJSONSerialize**

Bases: `invenio_records_rest.serializers.json.JSONSerializer`

preprocess_record(*pid*, *record*, *links_factory=None*, **kwargs)

Prepare a record and persistent identifier for serialization.

serialize(*pid*, *data*, *links_factory=None*, **kwargs)

class `inspirehep.modules.records.serializers.json_literature.`**`LiteratureJSONUISerializer`**(*schem* *'invenio_re* ***kwa*

Bases: `invenio_records_rest.serializers.json.JSONSerializer`

JSON brief format serializer.

`preprocess_record`(*pid*, *record*, *links_factory=None*, ***kwargs*)

`preprocess_search_hit`(*pid*, *record_hit*, *links_factory=None*, ***kwargs*)

`inspirehep.modules.records.serializers.json_literature.`**`get_citations_count`**(*original_record*)
Try to get citations

inspirehep.modules.records.serializers.latex module

Latex serializer for records.

class `inspirehep.modules.records.serializers.latex.`**`LatexSerializer`**(*format*, ***kwargs*)

Bases: `invenio_records_rest.serializers.marshmallow.MarshmallowMixin`, `invenio_records_rest.serializers.base.PreprocessorMixin`

Latex serializer for records.

`latex_template`()

`preprocess_record`(*pid*, *record*, *links_factory=None*, ***kwargs*)
Prepare a record and persistent identifier for serialization.

`serialize`(*pid*, *record*, *links_factory=None*, ***kwargs*)
Serialize a single record and persistent identifier.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for record links.

`serialize_search`(*pid_fetcher*, *search_result*, *links=None*, *item_links_factory=None*)
Serialize search result(s).

Parameters

- **`pid_fetcher`** – Persistent identifier fetcher.
- **`search_result`** – Elasticsearch search result.
- **`links`** – Dictionary of links to add to response.

Returns serialized search result(s)

Return type str

inspirehep.modules.records.serializers.marcxml module

MARCXML serializer.

class `inspirehep.modules.records.serializers.marcxml.`**`MARCXMLSerializer`**

Bases: `object`

MARCXML serializer.

`serialize`(*pid*, *record*, *links_factory=None*)

Serialize a single record as MARCXML.

`serialize_search`(*pid_fetcher*, *search_result*, *links=None*, *item_links_factory=None*)

Serialize a search result as MARCXML.

inspirehep.modules.records.serializers.pybtex_serializer_base module

BibTex serializer for records.

class `inspirehep.modules.records.serializers.pybtex_serializer_base.`**`PybtexSerializerBase`**(*schem*
write

Bases: `object`

Pybtex serializer for records.

`create_bibliography`(*record_list*)

Create a pybtex bibliography from individual entries.

Parameters **`record_list`** – A list of records of the bibliography.

Returns a serialized bibliography.

Return type str

`create_bibliography_entry`(*record*)

Get a texkey and bibliography entry for an inspire record.

Use the schema in `self.schema` to create a Pybtex bibliography entry and retrieve respective texkey from a `record`.

Parameters **`record`** – A literature record.

Returns bibliography entry as a (texkey, pybtex_entry) tuple.

Return type tuple

`serialize`(*pid*, *record*, *links_factory=None*)

Serialize a single Bibtex record.

Parameters

- **`pid`** – Persistent identifier instance.
- **`record`** – Record instance.
- **`links_factory`** – Factory function for the link generation, which are added to the response.

Returns single serialized Bibtex record

Return type str

`serialize_search`(*pid_fetcher*, *search_result*, *links=None*, *item_links_factory=None*)

Serialize search result(s).

Parameters

- **`pid_fetcher`** – Persistent identifier fetcher.
- **`search_result`** – Elasticsearch search result.

- **`links`** – Dictionary of links to add to response.

Returns serialized search result(s)

Return type str

inspirehep.modules.records.serializers.response module

Serialization response factories.

Responsible for creating a HTTP response given the output of a serializer.

`inspirehep.modules.records.serializers.response.`**`facets_responsify`**(*serializer*, *mimetype*)

Create a Facets serializer

As aggregations were removed from search query, now second call to the server is required to acquire data for Facets

Parameters

- **`serializer`** – Serializer instance.
- **`mimetype`** – MIME type of response.

`inspirehep.modules.records.serializers.response.`**`record_responsify_nocache`**(*serializer*, *mimetype*)

Create a Records-REST response serializer with no cache.

This is useful for formats such as bibtex where the code that generates the format might change so we don't want to use caching

Parameters

- **`serializer`** – Serializer instance.
- **`mimetype`** – MIME type of response.

Module contents

Record serialization.

Submodules

inspirehep.modules.records.api module

Inspire Records

class `inspirehep.modules.records.api.`**`ESRecord`**(*data*, *model=None*)

Bases: *`inspirehep.modules.records.api.InspireRecord`*

Record class that fetches records from ElasticSearch.

classmethod **`get_record`**(*object_uuid*, *with_deleted=False*)

Get record instance from ElasticSearch.

`updated`

Get last updated timestamp.

class `inspirehep.modules.records.api.`**`InspireRecord`**(*data*, *model=None*)
Bases: `invenio_records_files.api.Record`

Record class that fetches records from DataBase.

`add_document_or_figure`(*metadata*, *stream=None*, *is_document=True*, *file_name=None*, *key=None*)
Add a document or figure to the record.

Parameters

- **`metadata`** (*`dict`*) – metadata of the document or figure, see the schemas for more details, will be validated.
- **`stream`** (*`file like object`*) – if passed, will extract the file contents from it.
- **`is_document`** (*`bool`*) – if the given information is for a document, set to `` `False` `` for a figure.
- **`file_name`** (*`str`*) – Name of the file, used as a basis of the key for the files store.
- **`key`** (*`str`*) – if passed, will use this as the key for the files store and ignore `file_name`, use it to overwrite existing keys.

Returns metadata of the added document or figure.

Return type dict

Raises `TypeError` – if not `file_name` nor `key` are passed (one of them is required).

classmethod **`create`**(*data*, *id_=None*, ***kwargs*)
Override the default `create`.

To handle also the docmuments and figures retrieval.

Note: Might create an extra revision in the record if it had to download any documents or figures.

Keyword Arguments

- **`id`** (*`uuid`*) – an optional uuid to assign to the created record object.
- **`files_src_records`** (*`List[InspireRecord]`*) – if passed, it will try to get the files for the documents and figures from the first record in the list that has it in it's files iterator before downloading them, for example to merge existing records.
- **`skip_files`** (*`bool`*) – if `True` it will skip the files retrieval described above. Note also that, if not passed, it will fall back to the value of the `RECORDS_SKIP_FILES` configuration variable.

Examples

```
>>> record = {
...     '$schema': 'hep.json',
... }
>>> record = InspireRecord.create(record)
>>> record.commit()
```

classmethod **`create_or_update`**(*data*, ***kwargs*)
Create or update a record.

It will check if there is any record registered with the same `control_number` and `pid_type`. If it's `True`, it will update the current record, otherwise it will create a new one.

Keyword Arguments

- **`files_src_records`** (*List[InspireRecord]*) – if passed, it will try to get the files for the documents and figures from the first record in the list that has it in it's files iterator before downloading them, for example to merge existing records.
- **`skip_files`** (*bool*) – if `True` it will skip the files retrieval described above. Note also that, if not passed, it will fall back to the value of the `RECORDS_SKIP_FILES` configuration variable.

Examples

```
>>> record = {
...     '$schema': 'hep.json',
... }
>>> record = InspireRecord.create_or_update(record)
>>> record.commit()
```

`delete`()

Mark as deleted all pidstores for a specific record.

`download_documents_and_figures`(*only_new=False*, *src_records=()*)

Gets all the documents and figures of the record, and downloads them to the files property.

If the record does not have a control number yet, this function will do nothing and it will be left to the caller the task of calling it again once the control number is set.

When iterating through the documents and figures, the following happens:

- **if *url* field points to the files api:**
  - and there's no *src_records*: * and *only_new* is *False*: it will throw an error, as that

    would be the case that the record was created from scratch with a document that was already downloaded from another record, but that record was not passed, so we can't get the file.

    - **and *only_new* is *True*:**
      - if *key* exists in the current record files: it will do nothing, as the file is already there.
      - if *key* does not exist in the current record files: An exception will be thrown, as the file can't be retrieved.
  - and there's a *src_records*: * and *only_new* is *False*:
    - if *key* exists in the src_records files: it will download the file from the local path derived from the src_records files.
    - if *key* does not exist in the src_records files: An exception will be thrown, as the file can't be retrieved.
    - **and *only_new* is *True*:**
      - if *key* exists in the current record files: it will do nothing, as the file is already there.
      - if *key* does not exist in the current record files: * if *key* exists in the src_records files: it will download

the file from the local path derived from the src_records files.

- if *key* does not exist in the src_records files: An exception will be thrown, as the file can't be retrieved.

- if *url* field does not point to the files api: it will try to download the new file.

Parameters

- **`only_new`** (*`bool`*) – If True, will not re-download any files if the document['key'] matches an existing downloaded file.
- **`src_records`** (*`List[InspireRecord]`*) – if passed, it will try to get the files from this record files iterator before downloading them, for example to merge existing records.

`dumps`()

Returns a dict 'representation' of the record.

Note: this is not suitable to create a new record from it, as the representation will include some extra fields that should not be present in the record's json, see the 'to_dict' method instead.

`get_citations_count`(*show_duplicates=False*)

Returns citations count for this record.

`get_citing_records_query`

`get_modified_references`()

Return the ids of the references diff between the latest and the previous version.

The diff includes references added or deleted. Changes in a reference's content won't be detected.

Also, it detects if record was deleted/un-deleted compared to the previous version and, in such cases, returns the full list of references.

References not linked to any record will be ignored.

Note: record should be committed to DB in order to correctly get the previous version.

Returns pids of references changed from the previous version.

Return type Set[Tuple[str, int]]

`merge`(*other*)

Redirect pidstore of current record to the other InspireRecord.

Parameters **`other`** (`InspireRecord`) – The record that self(record) is going to be redirected.

static `mint`(*id_*, *data*)

Mint the record.

`to_dict`()

Gets a deep copy of the record's json.

`update`(*data*, ***kwargs*)

Override the default `update`.

To handle also the docmuments and figures retrieval.

Keyword Arguments

- **`files_src_records`** (`InspireRecord`) – if passed, it will try to get the files for the documents and figures from this record's files iterator before downloading them, for example to merge existing records.

- **skip_files** (*bool*) – if `True` it will skip the files retrieval described above. Note also that, if not passed, it will fall back to the value of the `RECORDS_SKIP_FILES` configuration variable.

validate()

Validate the record, also ensuring format compliance.

class `inspirehep.modules.records.api.`**referenced_records**(**args*, ***kwargs*)

Bases: `sqlalchemy.sql.functions.GenericFunction`

identifier = 'referenced_records'

name = 'referenced_records'

type = ARRAY(Text())

inspirehep.modules.records.checkers module

Records checkers.

`inspirehep.modules.records.checkers.`**add_linked_ids**(*dois*, *arxiv_ids*, *linked_ids*)

Increase the amount of times a paper with a specific doi has been cited by using its corresponding arxiv eprint and viceversa

`double_count` is used to count the times that a doi and an arxiv eprint appear in the same paper so that we don't count them twice in the final result

`inspirehep.modules.records.checkers.`**calculate_score_of_reference**(*counted_reference*)

Given a tuple of the number of times cited by a core record and a non core record, calculate a score associated with a reference.

The score is calculated giving five times more importance to core records

`inspirehep.modules.records.checkers.`**check_unlinked_references**()

Return two lists with the unlinked references that have a doi or an arxiv id.

If the reference read has a doi or an arxiv id, it is stored in the data structure. Once all the data is read, it is ordered by most relevant to less relevant.

`inspirehep.modules.records.checkers.`**get_all_unlinked_references**()

Return a list of dict, in which each dictionary corresponds to one reference object and the status of core or non core

`inspirehep.modules.records.checkers.`**increase_cited_count**(*result*, *identifier*, *core*)

Increases the number of times a reference with the same identifier has appeared

`inspirehep.modules.records.checkers.`**order_dictionary_into_list**(*result_dict*)

Return `result_dict` as an ordered list of tuples

inspirehep.modules.records.cli module

class `inspirehep.modules.records.cli.`**MyThreadPool**(*processes=None*, *initializer=None*, *initargs=()*)

Bases: `multiprocessing.pool.ThreadPool`

imap_unordered(*func*, *iterable*, *second_argument*, *chunksize=1*)

Like *imap()* method but ordering of results is arbitrary

`inspirehep.modules.records.cli.`**get_query_records_to_index**(*pid_types*)

Return a query for retrieving all non deleted records by pid_type

Parameters **pid_types** (*List[str]*) – a list of pid types

Returns SQLAlchemy query for non deleted record with pid type in *pid_types*

inspirehep.modules.records.cli.**next_batch**(*iterator*, *batch_size*)

Get first batch_size elements from the iterable, or remaining if less.

Parameters

- **iterator** – the iterator for the iterable
- **batch_size** – size of the requested batch

Returns batch (list)

inspirehep.modules.records.errors module

exception inspirehep.modules.records.errors.**MissingCitedRecordError**

Bases: invenio_records.errors.RecordsError

exception inspirehep.modules.records.errors.**MissingInspireRecordError**

Bases: invenio_records.errors.RecordsError

inspirehep.modules.records.ext module

Records extension.

class inspirehep.modules.records.ext.**InspireRecords**(*app=None*)

Bases: object

init_app(*app*)

inspirehep.modules.records.facets module

inspirehep.modules.records.facets.**must_match_all_filter**(*field*)

Bool filter containing a list of must matches.

inspirehep.modules.records.facets.**range_author_count_filter**(*field*)

Range filter for returning record only with 1 <= authors <= 10.

inspirehep.modules.records.json_ref_loader module

Resource-aware json reference loaders to be used with jsonref.

class inspirehep.modules.records.json_ref_loader.**AbstractRecordLoader**(*store=()*, *cache_results=True*)

Bases: jsonref.JsonLoader

Base for resource-aware record loaders.

Resolves the refered resource by the given uri by first checking against local resources.

get_record(*pid_type*, *recid*)

get_remote_json(*uri*, ***kwargs*)

class inspirehep.modules.records.json_ref_loader.**DatabaseJsonLoader**(*store=()*, *cache_results=True*)

Bases: *inspirehep.modules.records.json_ref_loader.AbstractRecordLoader*

get_record(*pid_type*, *recid*)

class `inspirehep.modules.records.json_ref_loader.`**`ESJsonLoader`**(*store=()*, *cache_results=True*)

Bases: *`inspirehep.modules.records.json_ref_loader.AbstractRecordLoader`*

Resolve resources by retrieving them from Elasticsearch.

`get_record`(*pid_type*, *recid*)

`inspirehep.modules.records.json_ref_loader.`**`SCHEMA_LOADER_CLS`**

Used in invenio-jsonschemas to resolve relative $ref.

alias of `JsonLoader`

`inspirehep.modules.records.json_ref_loader.`**`load_resolved_schema`**(*name*)

Load a JSON schema with all references resolved.

Parameters **`name`** (*`str`*) – name of the schema to load.

Returns the JSON schema with resolved references.

Return type dict

Examples

```
>>> resolved_schema = load_resolved_schema('authors')
```

`inspirehep.modules.records.json_ref_loader.`**`replace_refs`**(*obj*, *source='db'*)

Replaces record refs in obj by bypassing HTTP requests.

Any reference URI that comes from the same server and references a resource will be resolved directly either from the database or from Elasticsearch.

Parameters

- **`obj`** – Dict-like object for which '$ref' fields are recursively replaced.
- **`source`** –

  List of sources from which to resolve the references. It can be any of:

  - 'db' - resolve from Database
  - 'es' - resolve from Elasticsearch
  - 'http' - force using HTTP

Returns The same obj structure with the '$ref' fields replaced with the object available at the given URI.

inspirehep.modules.records.permissions module

class `inspirehep.modules.records.permissions.`**`RecordPermission`**(*record*, *func*, *user*)

Bases: `invenio_access.permissions.Permission`

Record permission.

- Read access given if collection not restricted.
- Update access given to admins and cataloguers.
- All other actions are denied for the moment.

`can`()

Determine access.

classmethod **`create`**(*record*, *action*, *user=None*)

Create a record permission.

`read_actions` **= ['read']**

`update_actions` = ['update']

`inspirehep.modules.records.permissions.`**`deny`**(*user*, *record*)
Deny access.

`inspirehep.modules.records.permissions.`**`get_user_collections`**()
Get user restricted collections.

`inspirehep.modules.records.permissions.`**`has_admin_permission`**(*user*, *record*)
Check if user has admin access to record.

`inspirehep.modules.records.permissions.`**`has_read_permission`**(*user*, *record*)
Check if user has read access to the record.

`inspirehep.modules.records.permissions.`**`has_update_permission`**(*user*, *record*)
Check if user has update access to the record.

`inspirehep.modules.records.permissions.`**`load_restricted_collections`**()

`inspirehep.modules.records.permissions.`**`load_user_collections`**(*app*, *user*)
Load user restricted collections upon login.

Receiver for flask_login.user_logged_in

`inspirehep.modules.records.permissions.`**`record_read_permission_factory`**(*record=None*)
Record permission factory.

`inspirehep.modules.records.permissions.`**`record_update_permission_factory`**(*record=None*)
Record permission factory.

inspirehep.modules.records.receivers module

Records receivers.

`inspirehep.modules.records.receivers.`**`assign_phonetic_block`**(*sender*, *record*, **args*, ***kwargs*)
Assign a phonetic block to each signature of a Literature record.

Uses the NYSIIS algorithm to compute a phonetic block from each signature's full name, skipping those that are not recognized as real names, but logging an error when that happens.

`inspirehep.modules.records.receivers.`**`assign_uuid`**(*sender*, *record*, **args*, ***kwargs*)
Assign a UUID to each signature of a Literature record.

`inspirehep.modules.records.receivers.`**`enhance_before_index`**(*record*)
Run all the receivers that enhance the record for ES in the right order.

Note: `populate_recid_from_ref` **MUST** come before `populate_bookautocomplete` because the latter puts a JSON reference in a completion _source, which would be expanded to an incorrect `_source_recid` by the former.

`inspirehep.modules.records.receivers.`**`enhance_record`**(*sender*, *record*, **args*, ***kwargs*)
Enhance the record for ES

`inspirehep.modules.records.receivers.`**`index_after_commit`**(*sender*, *changes*)
Index a record in ES after it was committed to the DB.

This cannot happen in an `after_record_commit` receiver from Invenio-Records because, despite the name, at that point we are not yet sure whether the record has been really committed to the DB.

`inspirehep.modules.records.receivers.`**`push_to_orcid`**

If needed, queue the push of the new changes to ORCID.

inspirehep.modules.records.tasks module

Records tasks.

(task)`inspirehep.modules.records.tasks.`**`batch_reindex`**

Task for bulk reindexing records.

`inspirehep.modules.records.tasks.`**`get_merged_records`**`()`

`inspirehep.modules.records.tasks.`**`get_records_to_update`**`(`*old_ref*`)`

(task)`inspirehep.modules.records.tasks.`**`index_modified_citations_from_record`**

Index records from the record's citations.

This tasks retries itself in 2 scenarios: - A new record is saved but it is not yet visible by this task bacause the transaction is not finished yet (RecordGetterError).

- When a record is updated, but new changes are not yet in DB, for the

same reason as above (StaleDataError).

Parameters

- **`pid_type`** (*`String`*) – pid type of the record
- **`pid_value`** (*`String`*) – pid value of the record
- **`db_version`** (*`Int`*) – the correct version of the record that we expect to index. This prevents loading stale data from the DB.

Raise: MissingCitedRecordError in case cited records are not found

(task)`inspirehep.modules.records.tasks.`**`merge_merged_records`**

Merge all records that were marked as merged.

`inspirehep.modules.records.tasks.`**`update_links`**`(`*record*, *old_ref*, *new_ref*`)`

(task)`inspirehep.modules.records.tasks.`**`update_refs`**

Update references in the entire database.

Replaces all occurrences of `old_ref` with `new_ref`, provided that they happen at one of the paths listed in `INSPIRE_REF_UPDATER_WHITELISTS`.

inspirehep.modules.records.utils module

Record related utils.

`inspirehep.modules.records.utils.`**`get_author_display_name`**`(`*name*`)`

Returns the display name in format Firstnames Lastnames

`inspirehep.modules.records.utils.`**`get_author_with_record_facet_author_name`**`(`*author*`)`

`inspirehep.modules.records.utils.`**`get_endpoint_from_record`**`(`*record*`)`

Return the endpoint corresponding to a record.

`inspirehep.modules.records.utils.`**`get_linked_records_in_field`**`(`*record*, *field_path*`)`

Get all linked records in a given field.

Parameters

- **`record`** (*`dict`*) – the record containing the links
- **`field_path`** (*`string`*) – a dotted field path specification understandable by `get_value`, containing a json reference to another record.

Returns an iterator on the linked record.

Return type Iterator[dict]

Warning: Currently, the order in which the linked records are yielded is different from the order in which they appear in the record.

Example

```
>>> record = {'references': [
...     {'record': {'$ref': 'https://labs.inspirehep.net/api/literature/1234'}},
...     {'record': {'$ref': 'https://labs.inspirehep.net/api/data/421'}},
... ]}
>>> get_linked_record_in_field(record, 'references.record')
[...]
```

`inspirehep.modules.records.utils.`**`get_pid_from_record_uri`**(*record_uri*)
Transform a URI to a record into a (pid_type, pid_value) pair.

`inspirehep.modules.records.utils.`**`is_author`**(*record*)

`inspirehep.modules.records.utils.`**`is_book`**(*record*)

`inspirehep.modules.records.utils.`**`is_data`**(*record*)

`inspirehep.modules.records.utils.`**`is_experiment`**(*record*)

`inspirehep.modules.records.utils.`**`is_hep`**(*record*)

`inspirehep.modules.records.utils.`**`is_institution`**(*record*)

`inspirehep.modules.records.utils.`**`is_journal`**(*record*)

`inspirehep.modules.records.utils.`**`populate_abstract_source_suggest`**(*record*)
Populate the `abstract_source_suggest` field in Literature records.

`inspirehep.modules.records.utils.`**`populate_affiliation_suggest`**(*record*)
Populate the `affiliation_suggest` field of Institution records.

`inspirehep.modules.records.utils.`**`populate_author_count`**(*record*)
Populate the `author_count` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_author_suggest`**(*record*, **args*, ***kwargs*)
Populate the `author_suggest` field of Authors records.

`inspirehep.modules.records.utils.`**`populate_authors_full_name_unicode_normalized`**(*record*)
Populate the `authors.full_name_normalized` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_authors_name_variations`**(*record*)
Generate name variations for an Author record.

`inspirehep.modules.records.utils.`**`populate_bookautocomplete`**(*record*)
Populate the ``bookautocomplete` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_citations_count`**(*record*)
Populate citations_count in ES from

`inspirehep.modules.records.utils.`**`populate_earliest_date`**(*record*)
Populate the `earliest_date` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_experiment_suggest`**(*record*)
Populates experiment_suggest field of experiment records.

`inspirehep.modules.records.utils.`**`populate_facet_author_name`**`(`*record*`)`

Populate the `facet_author_name` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_inspire_document_type`**`(`*record*`)`

Populate the `facet_inspire_doc_type` field of Literature records.

`inspirehep.modules.records.utils.`**`populate_name_variations`**`(`*record*`)`

Generate name variations for each signature of a Literature record.

`inspirehep.modules.records.utils.`**`populate_number_of_references`**`(`*record*`)`

Generate name variations for each signature of a Literature record.

`inspirehep.modules.records.utils.`**`populate_recid_from_ref`**`(`*record*`)`

Extract recids from all JSON reference fields and add them to ES.

For every field that has as a value a JSON reference, adds a sibling after extracting the record identifier. Siblings are named by removing `record` occurrences and appending `_recid` without doubling or prepending underscores to the original name.

Example:

```
{'record': {'$ref': 'http://x/y/2}}
```

is transformed to:

```
{
    'recid': 2,
    'record': {'$ref': 'http://x/y/2},
}
```

For every list of object references adds a new list with the corresponding recids, whose name is similarly computed.

Example:

```
{
    'records': [
        {'$ref': 'http://x/y/1'},
        {'$ref': 'http://x/y/2'},
    ],
}
```

is transformed to:

```
{
    'recids': [1, 2],
    'records': [
        {'$ref': 'http://x/y/1'},
        {'$ref': 'http://x/y/2'},
    ],
}
```

`inspirehep.modules.records.utils.`**`populate_title_suggest`**`(`*record*`)`

Populate the `title_suggest` field of Journals records.

inspirehep.modules.records.views module

Data model package.

class `inspirehep.modules.records.views.`**`Facets`**(***kwargs*)

Bases: `invenio_rest.views.ContentNegotiatedMethodView`

`get`(**args*, ***kwargs*)

`methods` **= ['GET']**

`view_name` **= '{0}_facets'**

class `inspirehep.modules.records.views.`**`LiteratureCitationsResource`**(***kwargs*)

Bases: `invenio_rest.views.ContentNegotiatedMethodView`

`get`(*pid_value*, **args*, ***kwargs*)

`methods` **= ['GET']**

`view_name` **= 'literature_citations'**

`inspirehep.modules.records.views.`**`facets_view`**(**args*, ***kwargs*)

`inspirehep.modules.records.views.`**`literature_citations_view`**(**args*, ***kwargs*)

inspirehep.modules.records.wrappers module

class `inspirehep.modules.records.wrappers.`**`AdminToolsMixin`**

Bases: `object`

`admin_tools`

class `inspirehep.modules.records.wrappers.`**`AuthorsRecord`**(*data*, *model=None*)

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for author records.

`title`

Get preferred title.

class `inspirehep.modules.records.wrappers.`**`ConferencesRecord`**(*data*, *model=None*)

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for conference records.

`title`

Get preferred title.

class `inspirehep.modules.records.wrappers.`**`ExperimentsRecord`**(*data*, *model=None*)

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for experiment records.

`title`

Get preferred title.

class `inspirehep.modules.records.wrappers.`**`InstitutionsRecord`**(*data*, *model=None*)

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for institution records.

`title`

Get preferred title.

class `inspirehep.modules.records.wrappers.`**`JobsRecord`**`(`*data*, *model=None*`)`

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for job records.

`similar`

`title`

Get preferred title.

class `inspirehep.modules.records.wrappers.`**`JournalsRecord`**`(`*data*, *model=None*`)`

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for journal records.

`name_variants`

Get name variations.

`publisher`

Get preferred title.

`title`

Get preferred title.

`urls`

Get urls.

class `inspirehep.modules.records.wrappers.`**`LiteratureRecord`**`(`*data*, *model=None*`)`

Bases: *`inspirehep.modules.records.api.ESRecord`*, *`inspirehep.modules.records.wrappers.AdminToolsMixin`*

Record class specialized for literature records.

`conference_information`

Conference information.

Returns a list with information about conferences related to the record.

`external_system_identifiers`

External system identification information.

Returns a list that contains information on first of each kind of external_system_idenitfiers

`get_link_info_for_external_sys_identifiers``(`*extid*, *ext_sys_id_info*`)`

Urls and names for external system identifiers

Returns a dictionary with 2 key value pairs, the first of which is the name of the external_system_identifier and the second is a link to the record in that external_system_identifer

`publication_information`

Publication information.

Returns a list with information about each publication note in the record.

`title`

Get preferred title.

Module contents

Data model package.

inspirehep.modules.refextract package

Submodules

inspirehep.modules.refextract.config module

Refextract config.

`inspirehep.modules.refextract.config.`**`REFERENCE_MATCHER_DATA_CONFIG`** **= {‘doc_type’: ‘data’, ‘source’: [’c**

Configuration for matching data records. Please note that the index and doc_type are different for data records.

`inspirehep.modules.refextract.config.`**`REFERENCE_MATCHER_DEFAULT_PUBLICATION_INFO_CONFIG`** **= {‘do**

Configuration for matching all HEP records using publication_info. These are separate from the unique queries since these can result in multiple matches (particularly in the case of errata).

`inspirehep.modules.refextract.config.`**`REFERENCE_MATCHER_JHEP_AND_JCAP_PUBLICATION_INFO_CONFI`**

Configuration for matching records JCAP and JHEP records using the publication_info, since we have to look at the year as well for accurate matching. These are separate from the unique queries since these can result in multiple matches (particularly in the case of errata).

`inspirehep.modules.refextract.config.`**`REFERENCE_MATCHER_UNIQUE_IDENTIFIERS_CONFIG`** **= {‘doc_type’:**

Configuration for matching all HEP records (including JHEP and JCAP records) using unique identifiers.

inspirehep.modules.refextract.matcher module

`inspirehep.modules.refextract.matcher.`**`match_reference`**(*reference*, *previous_matched_recid=None*)

Match a reference using inspire-matcher.

Parameters

- **`reference`** (`dict`) – the metadata of a reference.
- **`previous_matched_recid`** (`int`) – the record id of the last matched reference from the list of references.

Returns the matched reference.

Return type dict

`inspirehep.modules.refextract.matcher.`**`match_reference_with_config`**(*reference*, *config*, *previous_matched_recid=None*)

Match a reference using inspire-matcher given the config.

Parameters

- **`reference`** (`dict`) – the metadata of the reference.
- **`config`** (`dict`) – the list of inspire-matcher configurations for queries.
- **`previous_matched_recid`** (`int`) – the record id of the last matched reference from the list of references.

Returns the matched reference.

Return type dict

`inspirehep.modules.refextract.matcher.`**`match_references`**(*references*)

Match references to their respective records in INSPIRE.

Parameters **`references`** (`list`) – the list of references.

Returns the matched references.

Return type list

inspirehep.modules.refextract.tasks module

Refextract tasks.

(task)`inspirehep.modules.refextract.tasks.`**`create_journal_kb_file`**

Populate refextracts's journal KB from the database.

Uses two raw DB queries that use syntax specific to PostgreSQL to generate a file in the format that refextract expects, that is a list of lines like:

```
SOURCE---DESTINATION
```

which represents that `SOURCE` is translated to `DESTINATION` when found.

Note that refextract expects `SOURCE` to be normalized, which means removing all non alphanumeric characters, collapsing all contiguous whitespace to one space and uppercasing the resulting string.

inspirehep.modules.refextract.utils module

Refextract utils.

class `inspirehep.modules.refextract.utils.`**`KbWriter`**`(`*kb_path*`)`

Bases: `object`

`add_entry``(`*value*, *kb_key*`)`

Module contents

RefExtract integration.

inspirehep.modules.search package

Submodules

inspirehep.modules.search.api module

class `inspirehep.modules.search.api.`**`AuthorsSearch`**`(`***kwargs*`)`

Bases: `invenio_search.api.RecordsSearch`, *`inspirehep.modules.search.api.SearchMixin`*

Elasticsearch-dsl specialized class to search in Authors database.

class Meta

`doc_types` **= 'authors'**

`index` **= 'records-authors'**

`default_fields``()`

What fields to use when no keyword is specified.

class `inspirehep.modules.search.api.`**`ConferencesSearch`**`(`***kwargs*`)`

Bases: `invenio_search.api.RecordsSearch`, *`inspirehep.modules.search.api.SearchMixin`*

Elasticsearch-dsl specialized class to search in Conferences database.

class **Meta**

doc_types = 'conferences'

index = 'records-conferences'

default_fields()
What fields to use when no keyword is specified.

query_from_iq(*query_string*)
Initialize ES DSL object using INSPIRE query parser.

Parameters **query_string** (*string*) – Query string as a user would input in INSPIRE's search box.

Returns Elasticsearch DSL search class

class `inspirehep.modules.search.api.`**DataSearch**(***kwargs*)
Bases: `invenio_search.api.RecordsSearch`, *inspirehep.modules.search.api.SearchMixin*

Elasticsearch-dsl specialized class to search in Data database.

class **Meta**

doc_types = 'data'

index = 'records-data'

default_fields()
What fields to use when no keyword is specified.

class `inspirehep.modules.search.api.`**ExperimentsSearch**(***kwargs*)
Bases: `invenio_search.api.RecordsSearch`, *inspirehep.modules.search.api.SearchMixin*

Elasticsearch-dsl specialized class to search in Experiments database.

class **Meta**

doc_types = 'experiments'

index = 'records-experiments'

default_fields()
What fields to use when no keyword is specified.

class `inspirehep.modules.search.api.`**InstitutionsSearch**(***kwargs*)
Bases: `invenio_search.api.RecordsSearch`, *inspirehep.modules.search.api.SearchMixin*

Elasticsearch-dsl specialized class to search in Institutions database.

class **Meta**

doc_types = 'institutions'

index = 'records-institutions'

default_fields()
What fields to use when no keyword is specified.

`query_from_iq`(*query_string*)

Initialize ES DSL object using INSPIRE query parser.

Parameters **`query_string`** (*`string`*) – Query string as a user would input in INSPIRE's search box.

Returns Elasticsearch DSL search class

class `inspirehep.modules.search.api.`**`JobsSearch`**(***kwargs*)

Bases: `invenio_search.api.RecordsSearch`, *`inspirehep.modules.search.api.SearchMixin`*

Elasticsearch-dsl specialized class to search in Jobs database.

class **`Meta`**

`doc_types` = **'jobs'**

`index` = **'records-jobs'**

`default_fields`()

What fields to use when no keyword is specified.

class `inspirehep.modules.search.api.`**`JournalsSearch`**(***kwargs*)

Bases: `invenio_search.api.RecordsSearch`, *`inspirehep.modules.search.api.SearchMixin`*

Elasticsearch-dsl specialized class to search in Journals database.

class **`Meta`**

`doc_types` = **'journals'**

`index` = **'records-journals'**

`default_fields`()

What fields to use when no keyword is specified.

class `inspirehep.modules.search.api.`**`LiteratureSearch`**(***kwargs*)

Bases: `invenio_search.api.RecordsSearch`, *`inspirehep.modules.search.api.SearchMixin`*

Elasticsearch-dsl specialized class to search in Literature database.

class **`Meta`**

`default_filter` = **Match(_collections='Literature')**

`doc_types` = **'hep'**

`index` = **'records-hep'**

static **`citations`**(*record*, *page=1*, *size=10*)

`default_fields`()

What fields to use when no keyword is specified.

`query_from_iq`(*query_string*)

Initialize ES DSL object using INSPIRE query parser.

Parameters **`query_string`** (*`string`*) – Query string as a user would input in INSPIRE's search box.

Returns Elasticsearch DSL search class

class `inspirehep.modules.search.api.`**`SearchMixin`**
Bases: `object`

Mixin that adds helper functions to ElasticSearch DSL classes.

`get_source`(*uuid*, ***kwargs*)
Get source from a given uuid.

This function mimics the behaviour from the low level ES library get_source function.

Parameters **`uuid`** (*`UUID`*) – uuid of document to be retrieved.
Returns dict

`mget`(*uuids*, ***kwargs*)
Get source from a list of uuids.

Parameters **`uuids`** (*`list of strings representing uuids`*) – uuids of documents to be retrieved.
Returns list of JSON documents

`query_from_iq`(*query_string*)
Initialize ES DSL object using INSPIRE query parser.

Parameters **`query_string`** (*`string`*) – Query string as a user would input in INSPIRE's search box.
Returns Elasticsearch DSL search class

inspirehep.modules.search.bundles module

UI for Invenio-Search.

inspirehep.modules.search.ext module

Search extension.

class `inspirehep.modules.search.ext.`**`InspireSearch`**(*app=None*)
Bases: `object`

`init_app`(*app*)

inspirehep.modules.search.facets module

`inspirehep.modules.search.facets.`**`hep_author_publications`**()

inspirehep.modules.search.query_factory module

INSPIRE Query class to wrap the Q object from elasticsearch-dsl.

`inspirehep.modules.search.query_factory.`**`inspire_query_factory`**()
Create an Elastic Search DSL query instance using the generated Elastic Search query by the parser.

inspirehep.modules.search.search_factory module

INSPIRE search factory used in invenio-records-rest.

`inspirehep.modules.search.search_factory.`**`default_inspire_facets_factory`**(*search*, *index*)

`inspirehep.modules.search.search_factory.`**`inspire_facets_factory`**(*self*, *search*)

Parse query using Inspire-Query-Parser and prepare facets for it :param self: REST view. :param search: Elastic search DSL search instance.

Returns: Tuple with search instance and URL arguments.

`inspirehep.modules.search.search_factory.`**`inspire_filter_factory`**(*search*, *urlkwargs*, *search_index*)

Copies behaviour of default facets factory but without the aggregations, As facets factory is also responsible for filtering the year and author (invenio mess) :param search: Elastic search DSL search instance. :param urlkwargs: :param search_index: index name

Returns: tuple with search and urlarguments

`inspirehep.modules.search.search_factory.`**`inspire_search_factory`**(*self*, *search*)

Parse query using Inspire-Query-Parser.

Parameters

- **`self`** – REST view.
- **`search`** – Elastic search DSL search instance.

Returns Tuple with search instance and URL arguments.

`inspirehep.modules.search.search_factory.`**`select_source`**(*search*)

If search_idex is records-hep it filters the output to get only the useful data.

Parameters

- **`search`** – Elastic search DSL search instance.
- **`search_index`** – Index name

Returns: Elastic search DSL search instance.

inspirehep.modules.search.utils module

`inspirehep.modules.search.utils.`**`get_facet_configuration`**(*search_index*)

inspirehep.modules.search.views module

Search blueprint in order for template and static files to be loaded.

`inspirehep.modules.search.views.`**`default_sortoption`**(*sort_options*)

Get defualt sort option for Invenio-Search-JS.

`inspirehep.modules.search.views.`**`format_sortoptions`**(*sort_options*)

Create sort options JSON dump for Invenio-Search-JS.

`inspirehep.modules.search.views.`**`search`**()

Search page ui.

`inspirehep.modules.search.views.`**`sorted_options`**(*sort_options*)

Sort sort options for display.

`inspirehep.modules.search.views.`**`suggest`**()

Power typeahead.js search bar suggestions.

Module contents

Search module.

inspirehep.modules.submissions package

Subpackages

inspirehep.modules.submissions.serializers package

Subpackages

inspirehep.modules.submissions.serializers.schemas package

Submodules

inspirehep.modules.submissions.serializers.schemas.author module

class `inspirehep.modules.submissions.serializers.schemas.author.`**`Author`**(*extra=None, only=None, exclude=(), prefix=u'', strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

`before_dump`(*data*)

`build_author`(*data*)

`get_first_or_missing`(*value*)

`get_full_name`(*family_name, given_name*)

`get_name_splitted`(*data*)

`get_value_by_description_key`(*data, value*)

`get_value_or_missing`(*value*)

`opts` **= <marshmallow.schema.SchemaOpts object>**

Module contents

Schemas module.

Submodules

inspirehep.modules.submissions.serializers.json module

Submission loaders.

Module contents

Submission module.

Submodules

inspirehep.modules.submissions.loaders module

Submission loaders.

inspirehep.modules.submissions.loaders.**author_loader**(*schema_class*)

inspirehep.modules.submissions.loaders.**loader**(*schema_class*)

inspirehep.modules.submissions.tasks module

inspirehep.modules.submissions.tasks.**curation_ticket_context**(*user*, *obj*)
Context for authornew replies.

inspirehep.modules.submissions.tasks.**curation_ticket_needed**(**args*, ***kwargs*)
Check if the a curation ticket is needed.

inspirehep.modules.submissions.tasks.**new_ticket_context**(*user*, *obj*)
Context for authornew new tickets.

inspirehep.modules.submissions.tasks.**reply_ticket_context**(*user*, *obj*)
Context for authornew replies.

inspirehep.modules.submissions.tasks.**update_ticket_context**(*user*, *obj*)
Context for authornew new tickets.

inspirehep.modules.submissions.utils module

inspirehep.modules.submissions.utils.**get_record_from_legacy**(*record_id=None*)

inspirehep.modules.submissions.views module

Submissions views.

class inspirehep.modules.submissions.views.**SubmissionsResource**
Bases: flask.views.MethodView

decorators = [<function login_required>]

endpoint_to_data_type = {‘literature’: ‘hep’, ‘authors’: ‘authors’}

endpoint_to_form_serializer = {‘authors’: <class ‘inspirehep.modules.submissions.serializers.schemas.author.Au

endpoint_to_workflow_name = {'literature': 'article', 'authors': 'author'}

get (*endpoint*, *pid_value=None*)

methods = ['GET', 'POST', 'PUT']

post (*endpoint*)

put (*endpoint*, *pid_value*)

start_workflow_for_submission (*endpoint*, *submission_data*, *control_number=None*)

inspirehep.modules.submissions.views.**login_required** (*func*)

inspirehep.modules.submissions.views.**submissions_view** (**args*, ***kwargs*)

Module contents

Submission module.

inspirehep.modules.theme package

Submodules

inspirehep.modules.theme.bundles module

Inspire bundles.

inspirehep.modules.theme.ext module

Invenio standard theme.

class inspirehep.modules.theme.ext.**INSPIRETheme** (*app=None*, ***kwargs*)

Bases: `object`

Invenio theme extension.

init_app (*app*, *assets=None*, ***kwargs*)
Initialize application object.

init_config (*config*)
Initialize configuration.

setup_app (*app*)
Initialize Gravatar extension.

inspirehep.modules.theme.jinja2filters module

Jinja utilities for INSPIRE.

inspirehep.modules.theme.jinja2filters.**ads_links** (*record*)

inspirehep.modules.theme.jinja2filters.**apply_template_on_array** (*array*, *template_path*, ***common_context*)

Render a template specified by 'template_path'.

For every item in array, renders the template passing the item as 'content' parameter. Additionally attaches 'common_context' as other rendering arguments.

Returns list of rendered html strings.

Parameters

- **`array`** – iterable with specific context
- **`template_path`** – path to the template

Return type list of strings

`inspirehep.modules.theme.jinja2filters.`**`author_profile`**(*record*)
Return array of rendered links to authors.

`inspirehep.modules.theme.jinja2filters.`**`author_urls`**(*l*, *separator*)

`inspirehep.modules.theme.jinja2filters.`**`back_to_search_link`**(*referer*, *collection*)
Creates link to go back to search results in detailed pages.

`inspirehep.modules.theme.jinja2filters.`**`citation_phrase`**(*count*)

`inspirehep.modules.theme.jinja2filters.`**`clean_roles`**(*roles*)
Extract names from user roles.

`inspirehep.modules.theme.jinja2filters.`**`collection_select_current`**(*collection_name*, *current_collection*)
Returns the active collection based on the current collection page.

`inspirehep.modules.theme.jinja2filters.`**`conference_date`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`construct_date_format`**(*date*)

`inspirehep.modules.theme.jinja2filters.`**`count_plots`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`email_link`**(*value*)
Return single email rendered (mailto).

`inspirehep.modules.theme.jinja2filters.`**`email_links`**(*value*)
Return array of rendered links to emails.

`inspirehep.modules.theme.jinja2filters.`**`epoch_to_year_format`**(*date*)

`inspirehep.modules.theme.jinja2filters.`**`experiment_date`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`experiment_link`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`find_collection_from_url`**(*url*)
Returns the collection based on the URL.

`inspirehep.modules.theme.jinja2filters.`**`format_author_name`**(*name*)

`inspirehep.modules.theme.jinja2filters.`**`format_cnum_with_hyphons`**(*value*)

`inspirehep.modules.theme.jinja2filters.`**`format_cnum_with_slash`**(*value*)

`inspirehep.modules.theme.jinja2filters.`**`format_date`**(*date*)
Displays a date in a human-friendly format.

`inspirehep.modules.theme.jinja2filters.`**`institutes_links`**(*record*)
Return array of rendered links to institutes.

`inspirehep.modules.theme.jinja2filters.`**`is_cataloger`**(*user*)
Check if user has a cataloger role.

`inspirehep.modules.theme.jinja2filters.`**`is_external_link`**(*url*)
Checks if given url is an external link.

`inspirehep.modules.theme.jinja2filters.`**`is_list`**(*value*)
Checks if an object is a list.

`inspirehep.modules.theme.jinja2filters.`**`is_upper`**(*s*)

`inspirehep.modules.theme.jinja2filters.`**`join_array`**(*eval_ctx*, *value*, *separator*)

`inspirehep.modules.theme.jinja2filters.`**`join_nested_lists`**(*l*, *sep*)

`inspirehep.modules.theme.jinja2filters.`**`json_dumps`**(*data*)

`inspirehep.modules.theme.jinja2filters.`**`limit_facet_elements`**(*l*)

`inspirehep.modules.theme.jinja2filters.`**`link_to_hep_affiliation`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`new_line_after`**(*text*)

`inspirehep.modules.theme.jinja2filters.`**`proceedings_link`**(*record*)

`inspirehep.modules.theme.jinja2filters.`**`publication_info`**(*record*)
Display inline publication and conference information.

The record is a LiteratureRecord instance

`inspirehep.modules.theme.jinja2filters.`**`remove_duplicates_from_list`**(*l*)

`inspirehep.modules.theme.jinja2filters.`**`sanitize_arxiv_pdf`**(*arxiv_value*)
Sanitizes the arXiv PDF link so it is always correct

`inspirehep.modules.theme.jinja2filters.`**`sanitize_collection_name`**(*collection_name*)
Changes ‘hep’ to ‘literature’ and ‘hepnames’ to ‘authors’.

`inspirehep.modules.theme.jinja2filters.`**`search_for_experiments`**(*value*)

`inspirehep.modules.theme.jinja2filters.`**`show_citations_number`**(*citation_count*)
Shows citations number

`inspirehep.modules.theme.jinja2filters.`**`sort_list_by_dict_val`**(*l*)

`inspirehep.modules.theme.jinja2filters.`**`strip_leading_number_plot_caption`**(*text*)

`inspirehep.modules.theme.jinja2filters.`**`url_links`**(*record*)
Return array of rendered links.

`inspirehep.modules.theme.jinja2filters.`**`words`**(*value*, *limit*, *separator=' '*)
Return first *limit* number of words ending by *separator*‘ ‘

`inspirehep.modules.theme.jinja2filters.`**`words_to_end`**(*value*, *limit*, *separator=' '*)
Return last *limit* number of words ending by *separator*‘ ‘

inspirehep.modules.theme.views module

Theme views.

exception `inspirehep.modules.theme.views.`**`UnhealthCeleryTestException`**
Bases: `exceptions.Exception`

exception `inspirehep.modules.theme.views.`**`UnhealthTestException`**
Bases: `exceptions.Exception`

`inspirehep.modules.theme.views.`**`ajax_citations`**()
Handler for datatables citations view

Deprecated since version 2018-08-23.

`inspirehep.modules.theme.views.`**`ajax_conference_contributions`**`()`

Handler for other conference contributions

`inspirehep.modules.theme.views.`**`ajax_experiment_contributions`**`()`

Handler for experiment contributions

`inspirehep.modules.theme.views.`**`ajax_experiments_people`**`()`

Datatable handler to get people working in an experiment.

`inspirehep.modules.theme.views.`**`ajax_institutions_experiments`**`()`

Datatable handler to get experiments in an institution.

`inspirehep.modules.theme.views.`**`ajax_institutions_papers`**`()`

Datatable handler to get papers from an institution.

`inspirehep.modules.theme.views.`**`ajax_institutions_people`**`()`

Datatable handler to get people working in an institution.

`inspirehep.modules.theme.views.`**`ajax_other_conferences`**`()`

Handler for other conferences in the series

`inspirehep.modules.theme.views.`**`ajax_references`**`()`

Handler for datatables references view.

Deprecated since version 2018-06-07.

`inspirehep.modules.theme.views.`**`author_new`**`()`

`inspirehep.modules.theme.views.`**`author_update`**`()`

`inspirehep.modules.theme.views.`**`conferences`**`()`

View for conferences collection landing page.

`inspirehep.modules.theme.views.`**`data`**`()`

View for data collection landing page.

`inspirehep.modules.theme.views.`**`experiments`**`()`

View for experiments collection landing page.

`inspirehep.modules.theme.views.`**`get_experiment_publications`**`(`*experiment_name*`)`

Get paper count for a given experiment.

Parameters **`experiment_name`** (*`string`*) – canonical name of the experiment.

`inspirehep.modules.theme.views.`**`get_institution_experiments_datatables_rows`**`(`*hits*`)`

Row used by datatables to render institution experiments.

`inspirehep.modules.theme.views.`**`get_institution_experiments_from_es`**`(`*icn*`)`

Get experiments from a given institution.

To avoid killing ElasticSearch the number of experiments is limited.

Parameters **`icn`** (*`string`*) – Institution canonical name.

`inspirehep.modules.theme.views.`**`get_institution_papers_datatables_rows`**`(`*hits*`)`

Row used by datatables to render institution papers.

`inspirehep.modules.theme.views.`**`get_institution_papers_from_es`**`(`*recid*`)`

Get papers where some author is affiliated with institution.

Parameters **`recid`** (*`string`*) – id of the institution.

`inspirehep.modules.theme.views.`**`get_institution_people_datatables_rows`**`(`*recid*`)`

Datatable rows to render people working in an institution.

Parameters **`recid`** (*`string`*) – id of the institution.

`inspirehep.modules.theme.views.`**`health`**

(task)`inspirehep.modules.theme.views.`**`health_celery_task`**

`inspirehep.modules.theme.views.`**`healthcelery`**

`inspirehep.modules.theme.views.`**`hepnames`**`()`
View for authors collection landing page.

`inspirehep.modules.theme.views.`**`index`**`()`
View for literature collection landing page.

`inspirehep.modules.theme.views.`**`institutions`**`()`
View for institutions collection landing page.

`inspirehep.modules.theme.views.`**`insufficient_permissions`**`(`*error*`)`

`inspirehep.modules.theme.views.`**`internal_error`**`(`*error*`)`

`inspirehep.modules.theme.views.`**`jobs`**`()`
View for jobs collection landing page.

`inspirehep.modules.theme.views.`**`journals`**`()`
View for journals collection landing page.

`inspirehep.modules.theme.views.`**`linkedaccounts`**`()`
Redirect to the homepage when logging in with ORCID.

`inspirehep.modules.theme.views.`**`literature_new`**`()`

`inspirehep.modules.theme.views.`**`login_success`**`()`
Injects current user to the template and passes it to the parent tab.

`inspirehep.modules.theme.views.`**`page_not_found`**`(`*error*`)`

`inspirehep.modules.theme.views.`**`ping`**`()`

`inspirehep.modules.theme.views.`**`postfeedback`**`()`
Handler to create a ticket from user feedback.

`inspirehep.modules.theme.views.`**`record`**`(`*control_number*`)`

`inspirehep.modules.theme.views.`**`register_menu_items`**`()`
Hack to remove children of Settings menu

`inspirehep.modules.theme.views.`**`unauthorized`**`(`*error*`)`

`inspirehep.modules.theme.views.`**`unhealth`**

(task)`inspirehep.modules.theme.views.`**`unhealth_celery_task`**

`inspirehep.modules.theme.views.`**`unhealthcelery`**

Module contents

INSPIRE theme and filters.

inspirehep.modules.tools package

Submodules

inspirehep.modules.tools.authorlist module

Functions to parse an authorlist.

`inspirehep.modules.tools.authorlist.`**`create_authors`**(*text*)

Split text in (useful) blocks, sepatated by empty lines. 1 block: no affiliations 2 blocks: authors and affiliations more blocks: authors grouped by affiliation (not implemented yet)

Returns with two keys: `authors` of the form `(author_fullname, [author_affiliations])` and `warnings` which is a list of strings.

Return type dict

`inspirehep.modules.tools.authorlist.`**`determine_aff_type`**(*text*)

Guess format for affiliations. Return corresponding search pattern.

`inspirehep.modules.tools.authorlist.`**`determine_aff_type_character`**(*char_list*)

Guess whether affiliation are by number, letter or symbols (e.g. dagger). Numbers and letters should not be mixed.

`inspirehep.modules.tools.authorlist.`**`parse_affiliations`**(*text*)

Determine how affiliations are formatted. Return hash of id:affiliation

Allowed formats: don’t mix letters and numbers, lower-case letters only

1 CERN, Switzerland 2 DESY, Germany

1 CERN, Switzerland 2DESY, Germany

a CERN, Switzerland bb DESY, Germany

-

CERN, Switzerland # DESY, Germany

`inspirehep.modules.tools.authorlist.`**`parse_authors`**(*text*, *affiliations*)

Parse author names and convert to Lastname, Firstnames. Can be separated by ‘,’, newline or affiliation tag. Returns: List of tuples: (author_fullname, [author_affiliations]) List of strings: warnings

`inspirehep.modules.tools.authorlist.`**`split_id`**(*word*)

Separate potential aff-ids . E.g.: *‘12%$’ -> [’*‘, ‘12’ ‘%’, ‘$’]

inspirehep.modules.tools.bundles module

Tools bundles.

inspirehep.modules.tools.ext module

Tools extension.

class `inspirehep.modules.tools.ext.`**`InspireTools`**(*app=None*)

Bases: `object`

`init_app`(*app*)

inspirehep.modules.tools.utils module

Utility functions for various tools.

`inspirehep.modules.tools.utils.`**`authorlist`**`(`*text*`)`

Return an author-structure parsed from text and optional additional information.

inspirehep.modules.tools.views module

Tools views.

class `inspirehep.modules.tools.views.`**`InputTextForm`**`(`**args*, ***kwargs*`)`

Bases: *inspirehep.modules.forms.form.INSPIREForm*

Input form class.

`author_string` **= <UnboundField(TextAreaField, ('Author string',), {'render_kw': {'rows': 10, 'cols': 50}})>**

`inspirehep.modules.tools.views.`**`authorlist_form`**`()`

Render the authorlist page for formatting author strings.

`inspirehep.modules.tools.views.`**`tools_page`**`()`

Render the splash page for list of tools.

Module contents

Tools module.

inspirehep.modules.workflows package

Subpackages

inspirehep.modules.workflows.actions package

Submodules

inspirehep.modules.workflows.actions.author_approval module

Approval action for INSPIRE arXiv harvesting.

class `inspirehep.modules.workflows.actions.author_approval.`**`AuthorApproval`**

Bases: `object`

Class representing the author approval action.

`name` **= 'Approve author'**

static **`resolve`**`(`*obj*, **args*, ***kwargs*`)`

Resolve the action taken in the approval action.

inspirehep.modules.workflows.actions.hep_approval module

Approval action for INSPIRE arXiv harvesting.

class `inspirehep.modules.workflows.actions.hep_approval.`**`HEPApproval`**

Bases: `object`

Class representing the approval action.

`name` **= 'Approve record'**

static **`resolve`**(*obj*, **args*, ***kwargs*)

Resolve the action taken in the approval action.

inspirehep.modules.workflows.actions.match_approval module

Match action for INSPIRE.

class `inspirehep.modules.workflows.actions.match_approval.`**`MatchApproval`**

Bases: `object`

Class representing the match action.

`name` **= 'Match action'**

static **`resolve`**(*obj*, **args*, ***kwargs*)

Resolve the action taken in the approval action.

inspirehep.modules.workflows.actions.merge_approval module

Merge action for INSPIRE.

class `inspirehep.modules.workflows.actions.merge_approval.`**`MergeApproval`**

Bases: `object`

Class representing the merge action.

`name` **= 'Merge records'**

static **`resolve`**(*obj*, **args*, ***kwargs*)

Resolve the action taken in the approval action.

Module contents

Inspire workflows.

inspirehep.modules.workflows.mappings package

Subpackages

inspirehep.modules.workflows.mappings.v5 package

Module contents

Module contents

inspirehep.modules.workflows.serializers package

Subpackages

inspirehep.modules.workflows.serializers.schemas package

Submodules

inspirehep.modules.workflows.serializers.schemas.json module

Marshmallow JSON worfklow schema.

class `inspirehep.modules.workflows.serializers.schemas.json.`**`WorkflowSchemaJSONV1`**(*extra=None, only=None, exclude=(), prefix=u'', strict=None, many=False, context=None, load_only=(), dump_only=(), partial=False*)

Bases: `marshmallow.schema.Schema`

Schema for workflows.

class **`Meta`**

`strict` **= True**

`opts` **= <marshmallow.schema.SchemaOpts object>**

Module contents

Workflows schemas.

Module contents

Workflows serializers.

inspirehep.modules.workflows.tasks package

Submodules

inspirehep.modules.workflows.tasks.actions module

Tasks related to user actions.

`inspirehep.modules.workflows.tasks.actions.`**`add_core`**(**args*, ***kwargs*)

Mark a record as CORE if it was approved as CORE.

`inspirehep.modules.workflows.tasks.actions.`**`count_reference_coreness`**(**args*, ***kwargs*)

Count number of core/non-core matched references.

`inspirehep.modules.workflows.tasks.actions.`**`download_documents`**(**args*, ***kwargs*)

`inspirehep.modules.workflows.tasks.actions.`**`error_workflow`**(*message*)

Force an error in the workflow with the given message.

`inspirehep.modules.workflows.tasks.actions.`**`fix_submission_number`**(**args*, ***kwargs*)

Ensure that the submission number contains the workflow object id.

Unlike form submissions, records coming from HEPCrawl can't know yet which workflow object they will create, so they use the crawler job id as their submission number. We would like to have there instead the id of the workflow object from which they came from, so that, given a record, we can link to their original Holding Pen entry.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`halt_record`**(*action=None*, *message=None*)

Halt the workflow for approval with optional action.

`inspirehep.modules.workflows.tasks.actions.`**`in_production_mode`**(**args*, ***kwargs*)

Check if we are in production mode

`inspirehep.modules.workflows.tasks.actions.`**`is_arxiv_paper`**(**args*, ***kwargs*)

Check if a workflow contains a paper from arXiv.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns whether the workflow contains a paper from arXiv.

Return type bool

`inspirehep.modules.workflows.tasks.actions.`**`is_experimental_paper`**(**args*, ***kwargs*)

Check if a workflow contains an experimental paper.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns whether the workflow contains an experimental paper.

Return type bool

`inspirehep.modules.workflows.tasks.actions.`**`is_marked`**`(`*key*`)`

Check if the workflow object has a specific mark.

`inspirehep.modules.workflows.tasks.actions.`**`is_record_accepted`**`(`**args*, ***kwargs*`)`

Check if the record was approved.

`inspirehep.modules.workflows.tasks.actions.`**`is_record_relevant`**`(`**args*, ***kwargs*`)`

Shall we halt this workflow for potential acceptance or just reject?

`inspirehep.modules.workflows.tasks.actions.`**`is_submission`**`(`**args*, ***kwargs*`)`

Check if a workflow contains a submission.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns whether the workflow contains a submission.

Return type bool

`inspirehep.modules.workflows.tasks.actions.`**`jlab_ticket_needed`**`(`**args*, ***kwargs*`)`

Check if the a JLab curation ticket is needed.

`inspirehep.modules.workflows.tasks.actions.`**`load_from_source_data`**`(`**args*, ***kwargs*`)`

Restore the workflow data and extra_data from source_data.

`inspirehep.modules.workflows.tasks.actions.`**`mark`**`(`*key*, *value*`)`

Mark the workflow object by putting a value in a key in extra_data.

Note: Important. Committing a change to the database before saving the current workflow object will wipe away any content in `extra_data` not saved previously.

Parameters

- **`key`** – the key used to mark the workflow
- **`value`** – the value assigned to the key

Returns the decorator to decorate a workflow object

Return type func

`inspirehep.modules.workflows.tasks.actions.`**`normalize_journal_titles`**`(`**args*, ***kwargs*`)`

Normalize the journal titles

Normalize the journal titles stored in the *journal_title* field of each object contained in *publication_info*.

Note: The DB is queried in order to get the *$ref* of each journal and add it in *journal_record*.

Todo

Refactor: it must be checked that *normalize_journal_title* is appropriate.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`populate_journal_coverage`**`(`**args*, ***kwargs*`)`

Populate `journal_coverage` from the Journals DB.

Searches in the Journals DB if the current article was published in a journal that we harvest entirely, then populates the `journal_coverage` key in `extra_data` with `'full'` if it was, “‘partial’ otherwise.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`populate_submission_document`**`(`**args*, ***kwargs*`)`

`inspirehep.modules.workflows.tasks.actions.`**`preserve_root`**`(`**args*, ***kwargs*`)`

Save the current workflow payload to be used as root for the merger.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`refextract`**`(`**args*, ***kwargs*`)`

Extract references from various sources and add them to the workflow.

Runs `refextract` on both the PDF attached to the workflow and the references provided by the submitter, if any, then chooses the one that generated the most and attaches them to the workflow object.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`reject_record`**`(`*message*`)`

Reject record with message.

`inspirehep.modules.workflows.tasks.actions.`**`save_workflow`**`(`**args*, ***kwargs*`)`

Save the current workflow.

Saves the changes applied to the given workflow object in the database.

Note: The `save` function only indexes the current workflow. For this reason, we need to `db.session.commit()`.

Todo

Refactor: move this logic inside `WorkflowObject.save()`.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`set_refereed_and_fix_document_type`**`(`**args*, ***kwargs*`)`

Set the `refereed` field using the Journals DB.

Searches in the Journals DB if the current article was published in journals that we know for sure to be peer-reviewed, or that publish both peer-reviewed and non peer-reviewed content but for which we can infer that it belongs to the former category, and sets the `refereed` key in `data` to `True` if that was the case. If instead we know for sure that all journals in which it published are **not** peer-reviewed we set it to `False`.

Also replaces the `article` document type with `conference paper` if the paper was only published in non refereed proceedings.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.actions.`**`shall_halt_workflow`**`(`**args*, ***kwargs*`)`

Check if the workflow shall be halted.

`inspirehep.modules.workflows.tasks.actions.`**`validate_record`**`(`*schema*`)`

inspirehep.modules.workflows.tasks.arxiv module

Tasks used in OAI harvesting for arXiv record manipulation.

`inspirehep.modules.workflows.tasks.arxiv.`**`arxiv_author_list`**`(`*stylesheet='authorlist2marcxml.xsl'*`)`

Extract authors from any author XML found in the arXiv archive.

Parameters
- **`obj`** – Workflow Object to process
- **`eng`** – Workflow Engine processing the object

`inspirehep.modules.workflows.tasks.arxiv.`**`arxiv_derive_inspire_categories`**`(`**args*, ***kwargs*`)`

Derive `inspire_categories` from the arXiv categories.

Uses side effects to populate the `inspire_categories` key in `obj.data` by converting its arXiv categories.

Parameters
- **`obj`** (*`WorkflowObject`*) – a workflow object.
- **`eng`** (*`WorkflowEngine`*) – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.arxiv.`**`arxiv_package_download`**`(`**args*, ***kwargs*`)`

Perform the package download step for arXiv records.

Parameters
- **`obj`** – Workflow Object to process
- **`eng`** – Workflow Engine processing the object

`inspirehep.modules.workflows.tasks.arxiv.`**`arxiv_plot_extract`**`(`**args*, ***kwargs*`)`

Extract plots from an arXiv archive.

Parameters
- **`obj`** – Workflow Object to process
- **`eng`** – Workflow Engine processing the object

`inspirehep.modules.workflows.tasks.arxiv.`**`populate_arxiv_document`**`(`**args*, ***kwargs*`)`

inspirehep.modules.workflows.tasks.beard module

Set of workflow tasks for beard API.

`inspirehep.modules.workflows.tasks.beard.`**`get_beard_url`**`()`
Return the BEARD URL endpoint, if any.

`inspirehep.modules.workflows.tasks.beard.`**`guess_coreness`**`(`**args*, ***kwargs*`)`
Workflow task to ask Beard API for a coreness assessment.

`inspirehep.modules.workflows.tasks.beard.`**`prepare_payload`**`(`*record*`)`
Prepare payload to send to Beard API.

inspirehep.modules.workflows.tasks.classifier module

Set of tasks for classification.

`inspirehep.modules.workflows.tasks.classifier.`**`classify_paper`**`(`*taxonomy=None*, *rebuild_cache=False*, *no_cache=False*, *output_limit=20*, *spires=False*, *match_mode='full'*, *with_author_keywords=False*, *extract_acronyms=False*, *only_core_tags=False*, *fast_mode=False*`)`
Extract keywords from a pdf file or metadata in a OAI harvest.

`inspirehep.modules.workflows.tasks.classifier.`**`clean_instances_from_data`**`(`*output*`)`
Check if specific keys are of InstanceType and replace them with their id.

`inspirehep.modules.workflows.tasks.classifier.`**`filter_core_keywords`**`(`**args*, ***kwargs*`)`
Filter core keywords.

inspirehep.modules.workflows.tasks.magpie module

Set of workflow tasks for MagPie API.

`inspirehep.modules.workflows.tasks.magpie.`**`filter_magpie_response`**`(`*labels*, *limit*`)`
Filter response from Magpie API, keeping most relevant labels.

`inspirehep.modules.workflows.tasks.magpie.`**`get_magpie_url`**`()`
Return the Magpie URL endpoint, if any.

`inspirehep.modules.workflows.tasks.magpie.`**`guess_categories`**`(`**args*, ***kwargs*`)`
Workflow task to ask Magpie API for a subject area assessment.

`inspirehep.modules.workflows.tasks.magpie.`**`guess_experiments`**`(`**args*, ***kwargs*`)`
Workflow task to ask Magpie API for a subject area assessment.

`inspirehep.modules.workflows.tasks.magpie.`**`guess_keywords`**`(`**args*, ***kwargs*`)`
Workflow task to ask Magpie API for a keywords assessment.

`inspirehep.modules.workflows.tasks.magpie.`**`prepare_magpie_payload`**`(`*record*, *corpus*`)`
Prepare payload to send to Magpie API.

inspirehep.modules.workflows.tasks.manual_merging module

Tasks related to manual merging.

`inspirehep.modules.workflows.tasks.manual_merging.`**`halt_for_merge_approval`**`(`**args*, ***kwargs*`)`

Wait for curator approval.

Pauses the workflow using the `merge_approval` action, which is resolved whenever the curator says that the conflicts have been solved.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.manual_merging.`**`merge_records`**`(`**args*, ***kwargs*`)`

Perform a manual merge.

Merges two records stored in the workflow object as the content of the `head` and `update` keys, and stores the result in `obj.data`. Also stores the eventual conflicts in `obj.extra_data['conflicts']`.

Because this is a manual merge we assume that the two records have no common ancestor, so `root` is the empty dictionary.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.manual_merging.`**`save_roots`**`(`**args*, ***kwargs*`)`

Save and update the head roots and delete the update roots from the db.

If both head and update have a root from a given source, then the older one is removed and the newer one is assigned tot the head. Otherwise, assign the update roots from sources that are missing among the head roots to the head. i.e. it is an union-like operation.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.manual_merging.`**`store_records`**`(`**args*, ***kwargs*`)`

Store the records involved in the manual merge.

Performs the following steps:

1. Updates the `head` so that it contains the result of the merge.
2. Marks the `update` as merged with the `head` and deletes it.
3. Populates the `deleted_records` and `new_record` keys in, respectively, `head` and `update` so that they contain a JSON reference to each other.

Todo

The last step should be performed by the `merge` method itself.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

inspirehep.modules.workflows.tasks.matching module

Tasks to check if the incoming record already exist.

`inspirehep.modules.workflows.tasks.matching.`**`auto_approve`**(*obj*, *eng*)

Check if auto approve the current ingested article.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns True when the record belongs to an arXiv category that is fully harvested or if the primary category is *physics.data-an*, otherwise False.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`delete_self_and_stop_processing`**(**args*, ***kwargs*)

Delete both versions of itself and stops the workflow.

`inspirehep.modules.workflows.tasks.matching.`**`exact_match`**(**args*, ***kwargs*)

Return `True` if the record is already present in the system.

Uses the default configuration of the `inspire-matcher` to find duplicates of the current workflow object in the system.

Also sets the `matches.exact` property in `extra_data` to the list of control numbers that matched.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns `True` if the workflow object has a duplicate in the system `False` otherwise.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`fuzzy_match`**(**args*, ***kwargs*)

Return `True` if a similar record is found in the system.

Uses a custom configuration for `inspire-matcher` to find records similar to the current workflow object's payload in the system.

Also sets the `matches.fuzzy` property in `extra_data` to the list of the brief of first 5 record that matched.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns `True` if the workflow object has a duplicate in the system `False` otherwise.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`has_fully_harvested_category`**(*record*)

Check if the record in *obj.data* has fully harvested categories.

Parameters **`record`** (`dict`) – the ingested article.

Returns True when the record belongs to an arXiv category that is fully harvested, otherwise False.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`has_more_than_one_exact_match`**(**args*, ***kwargs*)

Does the record have more than one exact match.

`inspirehep.modules.workflows.tasks.matching.`**`has_same_source`**(*extra_data_key*)

Match a workflow in obj.extra_data[*extra_data_key*] by the source.

Takes a list of workflows from extra_data using as key *extra_data_key* and goes through them checking if at least one workflow has the same source of the current workflow object.

Parameters

- **`extra_data_key`** – the key to retrieve a workflow list from the current

- **object.** (workflow) –

Returns True if a workflow, whose id is in obj.extra_data[*extra_data_key*], matches the current workflow by the source.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`is_fuzzy_match_approved`**(**args*, ***kwargs*)

Check if a fuzzy match has been approved by a human.

`inspirehep.modules.workflows.tasks.matching.`**`match_non_completed_wf_in_holdingpen`**(*obj*, *eng*)

Return `True` if a matching wf is processing in the HoldingPen.

Uses a custom configuration of the `inspire-matcher` to find duplicates of the current workflow object in the Holding Pen not in the COMPLETED state.

Also sets `holdingpen_matches` in `extra_data` to the list of ids that matched.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns `True` if the workflow object has a duplicate in the Holding Pen that is not COMPLETED, `False` otherwise.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`match_previously_rejected_wf_in_holdingpen`**(*obj*, *eng*)

Return `True` if matches a COMPLETED and rejected wf in the HoldingPen.

Uses a custom configuration of the `inspire-matcher` to find duplicates of the current workflow object in the Holding Pen in the COMPLETED state, marked as `approved = False`.

Also sets `holdingpen_matches` in `extra_data` to the list of ids that matched.

Parameters

- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns `True` if the workflow object has a duplicate in the Holding Pen that is not COMPLETED, `False` otherwise.

Return type bool

`inspirehep.modules.workflows.tasks.matching.`**`pending_in_holding_pen`**(**args*, ***kwargs*)

Return the list of matching workflows in the holdingpen.

Matches the holdingpen records by their `arxiv_eprint`, their `doi`, and by a custom validator function.

Parameters

- **`obj`** – a workflow object.
- **`validation_func`** – a function used to filter the matched records.

Returns the ids matching the current `obj` that satisfy `validation_func`.

Return type (list)

`inspirehep.modules.workflows.tasks.matching.`**`physics_data_an_is_primary_category`**(*record*)

`inspirehep.modules.workflows.tasks.matching.`**`raise_if_match_wf_in_error_or_initial`**(*obj*, *eng*)

Raise if a matching wf is in ERROR or INITIAL state in the HoldingPen.

Uses a custom configuration of the `inspire-matcher` to find duplicates of the current workflow object in the Holding Pen not in the that are in ERROR or INITIAL state.

If any match is found, it sets `error_workflows_matched` in `extra_data` to the list of ids that matched and raise an error.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.matching.`**`set_core_in_extra_data`**`(*args, **kwargs)`

Set *core=True* in *obj.extra_data* if the record belongs to a core arXiv category

`inspirehep.modules.workflows.tasks.matching.`**`set_exact_match_as_approved_in_extradata`**`(*args, **kwarg`

Set the best match in *matches.approved* in extra_data.

`inspirehep.modules.workflows.tasks.matching.`**`set_fuzzy_match_approved_in_extradata`**`(*args, **kwargs)`

Set the human approved match in *matches.approved* in extra_data.

`inspirehep.modules.workflows.tasks.matching.`**`stop_matched_holdingpen_wfs`**`(obj, eng)`

Stop the matched workflow objects in the holdingpen.

Stops the matched workflows in the holdingpen by replacing their steps with a new one defined on the fly, containing a `stop` step, and executing it. For traceability reason, these workflows are also marked as `'stopped-by-wf'`, whose value is the current workflow's id.

In the use case of harvesting twice an article, this function is involved to stop the first workflow and let the current one being processed, since it the latest metadata.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.matching.`**`stop_processing`**`(*args, **kwargs)`

Stop processing the given workflow.

Stops the given workflow engine. This causes the stop of all the workflows related to it.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

inspirehep.modules.workflows.tasks.merging module

Tasks related to record merging.

`inspirehep.modules.workflows.tasks.merging.`**`has_conflicts`**`(*args, **kwargs)`

Return if the workflow has any confilicts.

`inspirehep.modules.workflows.tasks.merging.`**`merge_articles`**`(*args, **kwargs)`

Merge two articles.

The workflow payload is overwritten by the merged record, the conflicts are stored in `extra_data.conflicts`. Also, it adds a `callback_url` which contains the endpoint which resolves the merge conflicts.

Note: When the feature flag `FEATURE_FLAG_ENABLE_MERGER` is `False` it will skip the merge.

---

inspirehep.modules.workflows.tasks.refextract module

Workflow tasks using refextract API.

`inspirehep.modules.workflows.tasks.refextract.`**`extract_journal_info`**`(`**args, **kwargs*`)`

Extract the journal information from `pubinfo_freetext`.

Runs `extract_journal_reference` on the `pubinfo_freetext` key of each `publication_info`, if it exists, and uses the extracted information to populate the other keys.

Parameters
- **`obj`** – a workflow object.
- **`eng`** – a workflow engine.

Returns None

`inspirehep.modules.workflows.tasks.refextract.`**`extract_references_from_pdf`**`(`**args, **kwargs*`)`

Extract references from PDF and return in INSPIRE format.

`inspirehep.modules.workflows.tasks.refextract.`**`extract_references_from_raw_ref`**`(`*reference, custom_kbs_file=Non*

Extract references from raw references in reference element.

Parameters
- **`reference`** (*`dict`*) – a schema-compliant element of the `references` field. If it already contains a structured reference (that is, a `reference` key), no further processing is done. Otherwise, the contents of the `raw_refs` is extracted by `refextract`.
- **`custom_kbs_file`** (*`dict`*) – configuration for refextract knowledge bases.

Returns a list of schema-compliant elements of the `references` field, with all previously unextracted references extracted.

Return type List[dict]

Note: This function returns a list of references because one raw reference might correspond to several references.

`inspirehep.modules.workflows.tasks.refextract.`**`extract_references_from_raw_refs`**`(`**args, **kwargs*`)`

Extract references from raw references in reference list.

Parameters
- **`references`** (*`List[dict]`*) – a schema-compliant `references` field. If an element already contains a structured reference (that is, a `reference` key), it is not modified. Otherwise, the contents of the `raw_refs` is extracted by `refextract`.
- **`custom_kbs_file`** (*`dict`*) – configuration for refextract knowledge bases.

Returns a schema-compliant `references` field, with all previously unextracted references extracted.

Return type List[dict]

`inspirehep.modules.workflows.tasks.refextract.`**`extract_references_from_text`**`(`**args, **kwargs*`)`

Extract references from text and return in INSPIRE format.

inspirehep.modules.workflows.tasks.submission module

Contains INSPIRE specific submission tasks.

`inspirehep.modules.workflows.tasks.submission.`**`cleanup_pending_workflow`**(*args, **kwargs)

Cleans up the pending workflow entry for this workflow if any.

`inspirehep.modules.workflows.tasks.submission.`**`close_ticket`**(ticket_id_key='ticket_id')

Close the ticket associated with this record found in given key.

`inspirehep.modules.workflows.tasks.submission.`**`create_ticket`**(template, context_factory=None, queue='Test', ticket_id_key='ticket_id')

Create a ticket for the submission.

Creates the ticket in the given queue and stores the ticket ID in the extra_data key specified in ticket_id_key.

`inspirehep.modules.workflows.tasks.submission.`**`filter_keywords`**(*args, **kwargs)

Removes non-accepted keywords from the metadata

`inspirehep.modules.workflows.tasks.submission.`**`prepare_keywords`**(*args, **kwargs)

Prepares the keywords in the correct format to be sent

`inspirehep.modules.workflows.tasks.submission.`**`reply_ticket`**(template=None, context_factory=None, keep_new=False)

Reply to a ticket for the submission.

`inspirehep.modules.workflows.tasks.submission.`**`send_robotupload`**(url=None, callback_url='callback/workflows/robotuplo
mode='insert', extra_data_key=None)

Get the MARCXML from the model and ship it.

If callback_url is set the workflow will halt and the callback is responsible for resuming it.

`inspirehep.modules.workflows.tasks.submission.`**`send_to_legacy`**(obj, eng)

`inspirehep.modules.workflows.tasks.submission.`**`submit_rt_ticket`**(*args, **kwargs)

Submit ticket to RT with the given parameters.

`inspirehep.modules.workflows.tasks.submission.`**`wait_webcoll`**(*args, **kwargs)

inspirehep.modules.workflows.tasks.upload module

Tasks related to record uploading.

`inspirehep.modules.workflows.tasks.upload.`**`set_schema`**(*args, **kwargs)

Make sure schema is set properly and resolve it.

`inspirehep.modules.workflows.tasks.upload.`**`store_record`**(*args, **kwargs)

Insert or replace a record.

`inspirehep.modules.workflows.tasks.upload.`**`store_root`**(*args, **kwargs)

Insert or update the current record head's root into the `WorkflowsRecordSources` table.

Module contents

Workflows tasks.

inspirehep.modules.workflows.utils package

Module contents

Workflows utils.

`inspirehep.modules.workflows.utils.`**`convert`**`(`*xml*, *xslt_filename*`)`
Convert XML using given XSLT stylesheet.

`inspirehep.modules.workflows.utils.`**`copy_file_to_workflow`**`(`**args*, ***kwargs*`)`

`inspirehep.modules.workflows.utils.`**`do_not_repeat`**`(`*step_id*`)`
Decorator used to skip workflow steps when a workflow is re-run.

Will store the result of running the workflow step in source_data.persistent_data after running the first time, and skip the step on the following runs, also applying previously recorded 'changes' to extra_data.

The decorated function has to conform to the following signature:
def decorated_step(obj: WorkflowObject, eng: WorkflowEngine) -> Dict[str, Any]: ...
Where obj and eng are usual arguments following the protocol of all workflow steps. The returned value of the decorated_step will be used as a patch to be applied on the workflow object's source data (which 'replays' changes made by the workflow step).

Parameters **`step_id`** (*`str`*) – name of the workflow step, to be used as key in persistent_data
Returns the decorator
Return type callable

`inspirehep.modules.workflows.utils.`**`download_file_to_workflow`**`(`**args*, ***kwargs*`)`
Download a file to a specified workflow.

The `workflow.files` property is actually a method, which returns a `WorkflowFilesIterator`. This class inherits a custom `__setitem__` method from its parent, `FilesIterator`, which ends up calling `save` on an `invenio_files_rest.storage.pyfs.PyFSFileStorage` instance through `ObjectVersion` and `FileObject`. This method consumes the stream passed to it and saves in its place a `FileObject` with the details of the downloaded file.

Consuming the stream might raise a `ProtocolError` because the server might terminate the connection before sending any data. In this case we retry 5 times with exponential backoff before giving up.

`inspirehep.modules.workflows.utils.`**`get_document_in_workflow`**`(`**args*, ***kwds*`)`
Context manager giving the path to the document attached to a workflow object.
Arg: obj: workflow object

Returns The path to a local copy of the document. If no documents are present, it retuns None. If several documents are present, it prioritizes the fulltext. If several documents with the same priority are present, it takes the first one and logs an error.
Return type Optional[str]

`inspirehep.modules.workflows.utils.`**`get_resolve_edit_article_callback_url`**`()`
Resolve edit_article workflow letting it continue.

Note: It's using `inspire_workflows.callback_resolve_edit_article` route.

`inspirehep.modules.workflows.utils.`**`get_resolve_merge_conflicts_callback_url`**`()`
Resolve validation callback.

Returns the callback url for resolving the merge conflicts.

Note: It's using `inspire_workflows.callback_resolve_merge_conflicts` route.

`inspirehep.modules.workflows.utils.`**`get_resolve_validation_callback_url`**`()`

Resolve validation callback.

Returns the callback url for resolving the validation errors.

Note: It's using `inspire_workflows.callback_resolve_validation` route.

`inspirehep.modules.workflows.utils.`**`get_source_for_root`**`(`*source*`)`

Source for the root workflow object.

Parameters **`source`** (*`str`*) – the record source.

Returns the source for the root workflow object.

Return type (str)

Note: For the time being any workflow with `acquisition_source.source` different than `arxiv` and `submitter` will be stored as `publisher`.

`inspirehep.modules.workflows.utils.`**`get_validation_errors`**`(`*data*, *schema*`)`

Creates a `validation_errors` dictionary.

Parameters

- **`data`** (*`dict`*) – the object to validate.
- **`schema`** (*`str`*) – the name of the schema.

Returns `validation_errors` formatted dict.

Return type dict

`inspirehep.modules.workflows.utils.`**`ignore_timeout_error`**`(`*return_value=None*`)`

Ignore the TimeoutError, returning return_value when it happens.

Quick fix for `refextract` and `plotextract` tasks only. It shouldn't be used for others!

`inspirehep.modules.workflows.utils.`**`insert_wf_record_source`**`(`*json*, *record_uuid*, *source*`)`

Stores a record in the WorkflowRecordSource table in the db.

Parameters

- **`json`** (*`dict`*) – the record's content to store
- **`record_uuid`** (*`uuid`*) – the record's uuid
- **`source`** (*`string`*) – the source of the record

`inspirehep.modules.workflows.utils.`**`json_api_request`**`(`**args*, ***kwargs*`)`

Make JSON API request and return JSON response.

`inspirehep.modules.workflows.utils.`**`log_workflows_action`**`(`*action*, *relevance_prediction*, *object_id*, *user_id*, *source*, *user_action=''*`)`

Log the action taken by user compared to a prediction.

`inspirehep.modules.workflows.utils.`**`read_all_wf_record_sources`**`(`*record_uuid*`)`

Retrieve all `WorkflowRecordSource` for a given record id.

Parameters **`record_uuid`** (*`uuid`*) – the uuid of the record

Returns the `WorkflowRecordSource``s related to ``record_uuid`

Return type (list)

`inspirehep.modules.workflows.utils.`**`read_wf_record_source`**(*record_uuid*, *source*)

Retrieve a record from the `WorkflowRecordSource` table.

Parameters

- **`record_uuid`** (*`uuid`*) – the uuid of the record
- **`source`** (*`string`*) – the acquisition source value of the record

Returns the given record, if any or None

Return type (dict)

`inspirehep.modules.workflows.utils.`**`timeout_with_config`**(*config_key*)

Decorator to set a configurable timeout on a function.

Parameters **`config_key`** (*`str`*) – config key with a integer value representing the time in seconds after which the decorated function will abort, raising a `TimeoutError`. If the key is not present in the config, a `KeyError` is raised.

Note: This function is needed because it's impossible to pass a value read from the config as an argument to a decorator, as it gets evaluated before the application context is set up.

`inspirehep.modules.workflows.utils.`**`with_debug_logging`**(*func*)

Generate a debug log with info on what's going to run.

It tries its best to use the logging facilities of the object passed or the application context before falling back to the python logging facility.

inspirehep.modules.workflows.workflows package

Submodules

inspirehep.modules.workflows.workflows.article module

Workflow for processing single arXiv records harvested.

class `inspirehep.modules.workflows.workflows.article.`**`Article`**

Bases: `object`

Article ingestion workflow for Literature collection.

`data_type` **= 'hep'**

`name` **= 'HEP'**

`workflow` **= [<function load_from_source_data>, <function set_schema>, [<function mark>, <function mark>, <functio**

inspirehep.modules.workflows.workflows.author module

Workflow for processing single arXiv records harvested.

class `inspirehep.modules.workflows.workflows.author.`**`Author`**

Bases: `object`

Author ingestion workflow for HEPNames/Authors collection.

`data_type` **= 'authors'**

`name` **= 'Author'**

`workflow` **= [<function load_from_source_data>, <function set_schema>, <function validate_record>, [<function IF_EL**

inspirehep.modules.workflows.workflows.edit_article module

class `inspirehep.modules.workflows.workflows.edit_article.`**`EditArticle`**

Bases: `object`

Editing workflow for Literature collection.

`data_type` = **'hep'**

`name` = **'edit_article'**

`workflow` = **[<function change_status_to_waiting>, <function validate_record>, <function update_record>, <function se**

`inspirehep.modules.workflows.workflows.edit_article.`**`change_status_to_waiting`**(**args*, ***kwargs*)

`inspirehep.modules.workflows.workflows.edit_article.`**`update_record`**(*obj*, *eng*)

inspirehep.modules.workflows.workflows.manual_merge module

class `inspirehep.modules.workflows.workflows.manual_merge.`**`ManualMerge`**

Bases: `object`

`data_type` = **''**

`name` = **'MERGE'**

`workflow` = **[<function merge_records>, <function halt_for_merge_approval>, <function save_roots>, <function store_r**

`inspirehep.modules.workflows.workflows.manual_merge.`**`start_merger`**(*head_id*, *update_id*, *current_user_id=None*)

Start a new ManualMerge workflow to merge two records manually.

Parameters

- **`head_id`** – the id of the first record to merge. This record is the one that will be updated with the new information.
- **`update_id`** – the id of the second record to merge. This record is the one that is going to be deleted and replaced by *head*.
- **`current_user_id`** – Id of the current user provided by the Flask app.

Returns the current workflow object's id.

Return type (int)

Module contents

Our workflows.

Submodules

inspirehep.modules.workflows.bundles module

Bundles for forms used across INSPIRE.

inspirehep.modules.workflows.config module

Workflows configuration.

`inspirehep.modules.workflows.config.`**`WORKFLOWS_PLOTEXTRACT_TIMEOUT`** **= 300**
Time in seconds a plotextract task is allowed to run before it is killed.

`inspirehep.modules.workflows.config.`**`WORKFLOWS_REFEXTRACT_TIMEOUT`** **= 600**
Time in seconds a refextract task is allowed to run before it is killed.

inspirehep.modules.workflows.errors module

exception `inspirehep.modules.workflows.errors.`**`CallbackError`**
Bases: `invenio_workflows.errors.WorkflowsError`

Callback exception.

`code` **= 400**

`error_code` **= 'CALLBACK_ERROR'**

`errors` **= None**

`message` **= 'Workflow callback error.'**

`to_dict`()
Execption to dictionary.

`workflow` **= None**

exception `inspirehep.modules.workflows.errors.`**`CallbackMalformedError`**(*errors=None*, ***kwargs*)
Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Malformed request exception.

`error_code` **= 'MALFORMED'**

`message` **= 'The workflow request is malformed.'**

exception `inspirehep.modules.workflows.errors.`**`CallbackRecordNotFoundError`**(*recid*, ***kwargs*)
Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Record not found exception.

`code` **= 404**

`error_code` **= 'RECORD_NOT_FOUND'**

exception `inspirehep.modules.workflows.errors.`**`CallbackValidationError`**(*workflow_data*, ***kwargs*)
Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Validation error exception.

`error_code` **= 'VALIDATION_ERROR'**

`message` **= 'Validation error.'**

exception `inspirehep.modules.workflows.errors.`**`CallbackWorkflowNotFoundError`**(*workflow_id*, ***kwargs*)
Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Workflow not found exception.

code = 404

error_code = 'WORKFLOW_NOT_FOUND'

exception `inspirehep.modules.workflows.errors.`**`CallbackWorkflowNotInMergeState`**(*workflow_id*, ***kwargs*)

Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Workflow not in validation error exception.

error_code = 'WORKFLOW_NOT_IN_MERGE_STATE'

exception `inspirehep.modules.workflows.errors.`**`CallbackWorkflowNotInValidationError`**(*workflow_id*, ***kwargs*)

Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Validation workflow not in validation error exception.

error_code = 'WORKFLOW_NOT_IN_ERROR_STATE'

exception `inspirehep.modules.workflows.errors.`**`CallbackWorkflowNotInWaitingEditState`**(*workflow_id*, ***kwargs*)

Bases: *`inspirehep.modules.workflows.errors.CallbackError`*

Workflow not in validation error exception.

error_code = 'WORKFLOW_NOT_IN_WAITING_FOR_CURATION_STATE'

exception `inspirehep.modules.workflows.errors.`**`DownloadError`**

Bases: `invenio_workflows.errors.WorkflowsError`

Error representing a failed download in a workflow.

exception `inspirehep.modules.workflows.errors.`**`MergeError`**

Bases: `invenio_workflows.errors.WorkflowsError`

Error representing a failed merge in a workflow.

inspirehep.modules.workflows.ext module

Workflows extension.

class `inspirehep.modules.workflows.ext.`**`InspireWorkflows`**(*app=None*)

Bases: `object`

`init_app`(*app*)

`init_config`(*app*)

inspirehep.modules.workflows.loaders module

Workflows loader.

`inspirehep.modules.workflows.loaders.`**`marshmallow_loader`**(*schema_class*, *partial=False*)

Marshmallow loader.

`inspirehep.modules.workflows.loaders.`**`workflow_loader`**()

inspirehep.modules.workflows.models module

Extra models for workflows.

class `inspirehep.modules.workflows.models.`**`Timestamp`**

Bases: `object`

Timestamp model mix-in with fractional seconds support. SQLAlchemy-Utils timestamp model does not have support for fractional seconds.

`created` **= Column(None, DateTime(), table=None, default=ColumnDefault(<function utcnow>))**

`updated` **= Column(None, DateTime(), table=None, default=ColumnDefault(<function utcnow>))**

class `inspirehep.modules.workflows.models.`**`WorkflowsAudit`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`

`action`

`created`

`decision`

`id`

`object_id`

`save`()

Save object to persistent storage.

`score`

`source`

`user_action`

`user_id`

class `inspirehep.modules.workflows.models.`**`WorkflowsPendingRecord`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`

`record_id`

`workflow_id`

class `inspirehep.modules.workflows.models.`**`WorkflowsRecordSources`**(***kwargs*)

Bases: `sqlalchemy.ext.declarative.api.Model`, *`inspirehep.modules.workflows.models.Timestamp`*

`created`

`json`

`record_uuid`

`source`

`updated`

`inspirehep.modules.workflows.models.`**`timestamp_before_update`**(*mapper*, *connection*, *target*)

Update *updated* property with current time on *before_update* event.

inspirehep.modules.workflows.proxies module

Extra models for workflows.

`inspirehep.modules.workflows.proxies.`**`load_antikeywords`**`(`**args*, ***kwds*`)`

Loads list of antihep keywords with cached gotcha.

inspirehep.modules.workflows.search module

Search factory for INSPIRE workflows UI.

We specify in this custom search factory which fields elasticsearch should return in order to not always return the entire record.

Add a key path to the includes variable to include it in the API output when listing/searching across workflow objects (Holding Pen).

`inspirehep.modules.workflows.search.`**`holdingpen_search_factory`**`(`*self*, *search*, ***kwargs*`)`

Override search factory.

inspirehep.modules.workflows.views module

Callback blueprint for interaction with legacy.

class `inspirehep.modules.workflows.views.`**`ResolveEditArticleResource`**

Bases: `flask.views.MethodView`

Resolve *edit_article* callback.

When the workflow needs to resolve conficts, the workflow stops in `HALTED` state, to continue this endpoint is called. If it's called and the conflicts are not resolved it will just save the workflow.

Parameters **`workflow_data`** (*`dict`*) – the workflow object send in the request's payload.

`methods` **= ['PUT']**

`put``()`

Handle callback for merge conflicts.

class `inspirehep.modules.workflows.views.`**`ResolveMergeResource`**

Bases: `flask.views.MethodView`

Resolve merge callback.

When the workflow needs to resolve conficts, the workflow stops in `HALTED` state, to continue this endpoint is called. If it's called and the conflicts are not resolved it will just save the workflow.

Parameters **`workflow_data`** (*`dict`*) – the workflow object send in the request's payload.

`methods` **= ['PUT']**

`put``()`

Handle callback for merge conflicts.

class `inspirehep.modules.workflows.views.`**`ResolveValidationResource`**

Bases: `flask.views.MethodView`

Resolve validation error callback.

`methods` **= ['PUT']**

put()

Handle callback from validation errors.

When validation errors occur, the workflow stops in `ERROR` state, to continue this endpoint is called.

Parameters **workflow_data** (*dict*) – the workflow object send in the request's payload.

Examples

An example of successful call:

$ curl http://web:5000/callback/workflows/resolve_validation_errors -H "Host: localhost:5000" -H "Content-Type: application/json" -d '{

"_extra_data": { ... extra data content

}, "id": 910648, "metadata": {

"$schema": "https://labs.inspirehep.net/schemas/records/hep.json", ... record content

}

}'

The response:

HTTP 200 OK

{"mesage": "Workflow 910648 validated, continuing it."}

A failed example:

$ curl http://web:5000/callback/workflows/resolve_validation_errors -H "Host: localhost:5000" -H "Content-Type: application/json" -d '{

"_extra_data": { ... extra data content

}, "id": 910648, "metadata": {

"$schema": "https://labs.inspirehep.net/schemas/records/hep.json", ... record content

}

}'

The error response will contain the workflow that was passed, with the new validation errors:

HTTP 400 Bad request

{

"_extra_data": {

"validatior_errors": [

{ "path": ["path", "to", "error"], "message": "required: ['missing_key1', 'missing_key2']"

}

], ... rest of extra data content

}, "id": 910648, "metadata": {

"$schema": "https://labs.inspirehep.net/schemas/records/hep.json", ... record content

}

}

`inspirehep.modules.workflows.views.`**`callback_resolve_edit_article`**`(`**args*, ***kwargs*`)`

Resolve *edit_article* callback.

When the workflow needs to resolve conficts, the workflow stops in `HALTED` state, to continue this endpoint is called. If it's called and the conflicts are not resolved it will just save the workflow.

Parameters **`workflow_data`** (`dict`) – the workflow object send in the request's payload.

`inspirehep.modules.workflows.views.`**`callback_resolve_merge_conflicts`**`(`**args*, ***kwargs*`)`

Resolve merge callback.

When the workflow needs to resolve conficts, the workflow stops in `HALTED` state, to continue this endpoint is called. If it's called and the conflicts are not resolved it will just save the workflow.

Parameters **`workflow_data`** (`dict`) – the workflow object send in the request's payload.

`inspirehep.modules.workflows.views.`**`callback_resolve_validation`**`(`**args*, ***kwargs*`)`

Resolve validation error callback.

`inspirehep.modules.workflows.views.`**`error_handler`**`(`*error*`)`

Callback error handler.

`inspirehep.modules.workflows.views.`**`inspect_merge`**`(`*holdingpen_id*`)`

`inspirehep.modules.workflows.views.`**`robotupload_callback`**`()`

Handle callback from robotupload.

If robotupload was successful caches the workflow object id that corresponds to the uploaded record, so the workflow can be resumed when webcoll finish processing that record. If robotupload encountered an error sends an email to site administrator informing him about the error.

Examples

An example of failed callback that did not get to create a recid (the "nonce" is the workflow id):

```
$ curl \
    http://web:5000/callback/workflows/robotupload \
    -H "Host: localhost:5000" \
    -H "Content-Type: application/json" \
    -d '{
        "nonce": 1,
        "results": [
            {
                "recid":-1,
                "error_message": "Record already exists",
                "success": false
            }
        ]
    }'
```

One that created the recid, but failed later:

```
$ curl \
    http://web:5000/callback/workflows/robotupload \
    -H "Host: localhost:5000" \
    -H "Content-Type: application/json" \
    -d '{
        "nonce": 1,
```

```
        "results": [
            {
                "recid":1234,
                "error_message": "Unable to parse pdf.",
                "success": false
            }
        ]
    }'
```

A successful one:

```
$ curl \
    http://web:5000/callback/workflows/robotupload \
    -H "Host: localhost:5000" \
    -H "Content-Type: application/json" \
    -d '{
        "nonce": 1,
        "results": [
            {
                "recid":1234,
                "error_message": "",
                "success": true
            }
        ]
    }'
```

`inspirehep.modules.workflows.views.`**`start_edit_article_workflow`**`(`*recid*`)`

`inspirehep.modules.workflows.views.`**`webcoll_callback`**`()`

Handle a callback from webcoll with the record ids processed.

Expects the request data to contain a list of record ids in the recids field.

Example

An example of callback:

```
$ curl \
    http://web:5000/callback/workflows/webcoll \
    -H "Host: localhost:5000" \
    -F 'recids=1234'
```

Module contents

Workflows module.

Module contents

INSPIRE modules.

inspirehep.testlib package

Subpackages

inspirehep.testlib.api package

Submodules

inspirehep.testlib.api.author_form module

Literature suggestion form testlib.

class `inspirehep.testlib.api.author_form.`**`AuthorFormApiClient`**(*client*)

Bases: `object`

`SUBMIT_AUTHOR_FORM_URL` **= '/authors/new/submit'**

`submit`(*form_input_data*)

class `inspirehep.testlib.api.author_form.`**`AuthorFormInputData`**(*given_names, research_field, status='active', family_name=None, display_name=None*)

Bases: `object`

`request_data`()

inspirehep.testlib.api.base_resource module

Base resource class and utils.

class `inspirehep.testlib.api.base_resource.`**`BaseResource`**

Bases: `object`

inspirehep.testlib.api.callback module

/callback endpoint api client and resources.

class `inspirehep.testlib.api.callback.`**`CallbackClient`**(*client*)

Bases: `object`

Client for the Inspire callback

`CALLBACK_URL` **= '/callback/workflows'**

`robotupload`(*nonce, results*)

Parameters

- **`nonce`** (`int`) – nonce parameter passed to robotupload, usually the workflow id.
- **`results`** (`list[RobotuploadCallbackResult]`) – list of robotupload results.

`webcoll`(*recids*)

Parameters **`recids`** (`list(int)`) – list of recids that webcoll parsed.

class `inspirehep.testlib.api.callback.`**`RobotuploadCallbackResult`**(*recid*, *error_message*, *success*, *marcxml*, *url*)

Bases: `dict`

inspirehep.testlib.api.e2e module

/holdingpen endopint api client and resources.

class `inspirehep.testlib.api.e2e.`**`E2EClient`**(*client*)

Bases: `object`

Client for the Inspire E2E api.

`INIT_DB_URL` = **'/e2e/init_db'**

`INIT_ES_URL` = **'/e2e/init_es'**

`INIT_FIXTURES_URL` = **'/e2e/init_fixtures'**

`SCHEDULE_CRAWL_URL` = **'/e2e/schedule_crawl'**

`init_db`()

`init_es`()

`init_fixtures`()

`schedule_crawl`(***params*)

inspirehep.testlib.api.holdingpen module

/holdingpen endopint api client and resources.

class `inspirehep.testlib.api.holdingpen.`**`HoldingpenApiClient`**(*client*)

Bases: `object`

Client for the Inspire Holdingpen

`HOLDINGPEN_API_URL` = **'/api/holdingpen/'**

`HOLDINGPEN_EDIT_URL` = **'/api/holdingpen/{workflow_id}/action/edit'**

`HOLDINGPEN_RESOLVE_URL` = **'/api/holdingpen/{workflow_id}/action/resolve'**

`HOLDINGPEN_RESTART_URL` = **'/api/holdingpen/{workflow_id}/action/restart'**

`accept_core`(*holdingpen_id*)

`accept_non_core`(*holdingpen_id*)

`edit_workflow`(*holdingpen_entry*)

Helper method to edit a holdingpen entry.

Parameters **`holdingpen_entry`** (`HoldingpenResource`) – entry updated with the already changed data.

Returns

The actual http response to the last call (the actual /edit endpoint).

Return type requests.Response

Raises `requests.exceptions.BaseHttpError` – any error related to the http calls made.

Example

```
>>> my_entry = holdingpen_client.get_detail_entry(holdingpen_id=1234)
>>> my_entry.core = False    # do some changes
>>> holdingpen_client.edit_workflow(holdingpen_entry=my_entry)
<Response [200]>
```

get_detail_entry(*holdingpen_id*)

get_list_entries()

reject(*holdingpen_id*)

resolve_merge_conflicts(*hp_entry*)

restart_workflow(*holdingpen_entry_id*)

resume(*hp_entry*)

run_harvest(*spider*, *workflow='article'*, ***kwargs*)
Run a harvest scheduling a job in celery

class `inspirehep.testlib.api.holdingpen.`**HoldingpenAuthorResource**(*display_name*, ***kwargs*)

Bases: *inspirehep.testlib.api.holdingpen.HoldingpenResource*

Holdingpen for an author workflow.

to_json()

class `inspirehep.testlib.api.holdingpen.`**HoldingpenLiteratureResource**(*titles*, *auto_approved=None*, *doi=None*, *arxiv_eprint=None*, *approved_match=None*, ***kwargs*)

Bases: *inspirehep.testlib.api.holdingpen.HoldingpenResource*

Holdingpen entry for a literature workflow.

set_conflicts(*conflicts*)

to_json()

class `inspirehep.testlib.api.holdingpen.`**HoldingpenResource**(*workflow_id*, *approved*, *is_update*, *core*, *status*, *control_number*)

Bases: *inspirehep.testlib.api.base_resource.BaseResource*

Inspire holdingpen entry to represent a workflow

classmethod from_json(*json*, *workflow_id=None*)
Constructor for a holdingpen entry, it will be able to be mapped to and from json, and used to fully edit entries. Usually you pass to it the full raw json from the details of a holdingpen entry.

Parameters **json** (*dict*) – dictionary of a single entry as returned by the api.

set_action(*action*)

to_json()
Translates the current entry to a json applying any changes to the original json passed, or just with the info added to the entry since it's instantiation.

Returns Json view of the current status of the entry.

Return type dict

inspirehep.testlib.api.literature module

/literature endpoint api client and resources.

class `inspirehep.testlib.api.literature.`**`LiteratureApiClient`**(*client*)

Bases: `object`

Client for the Inspire Literature section

`LITERATURE_API_URL` = **'/api/literature/'**

`get_record`(*rec_id*)

class `inspirehep.testlib.api.literature.`**`LiteratureResource`**(*control_number*, *doi*, *arxiv_eprint*, *titles*)

Bases: *`inspirehep.testlib.api.base_resource.BaseResource`*

Inspire base entry to represent a literature record

classmethod **`from_json`**(*json*)

Parameters **`json`** (*`dict`*) – dictionary of a single entry as returned by the api.

class `inspirehep.testlib.api.literature.`**`LiteratureResourceTitle`**(*source*, *title*)

Bases: *`inspirehep.testlib.api.base_resource.BaseResource`*

classmethod **`from_json`**(*json*)

`to_json`()

inspirehep.testlib.api.literature_form module

Literature suggestion form testlib.

class `inspirehep.testlib.api.literature_form.`**`LiteratureFormApiClient`**(*client*)

Bases: `object`

`SUBMIT_LITERATURE_FORM_URL` = **'/literature/new/submit'**

`submit`(*form_input_data*)

class `inspirehep.testlib.api.literature_form.`**`LiteratureFormInputData`**(*title*, *language='en'*, *type_of_doc='article'*)

Bases: `object`

`add_author`(*name*, *affiliation=None*)

`request_data`()

inspirehep.testlib.api.mitm_client module

Client interface for INSPIRE-MITMPROXY.

class `inspirehep.testlib.api.mitm_client.`**`MITMClient`**(*proxy_host='http://mitm-manager.local'*)

Bases: `object`

`assert_interaction_used`(*service_name*, *interaction_name*, *times=None*)

get_interactions_for_service(*service_name*)

set_scenario(*scenario_name*)

start_recording()

stop_recording()

`inspirehep.testlib.api.mitm_client.`**with_mitmproxy**(**args*, ***kwargs*)

Decorator to abstract fixture recording and scenario setup for the E2E tests with mitmproxy.

Parameters

- **scenario_name** (`Optional[str]`) – scenario name, by default test name without '**test_**' prefix
- **should_record** (`Optional[bool]`) – is recording new interactions allowed during test run, by default *False*
- ***args** (`List[Callable]`) – list of length of either zero or one: decorated function. This is to allow the decorator to function both with and without calling it with parameters: if args is present, we can deduce that the decorator was used without parameters.

Returns

a decorator the can be used both with and without calling brackets (if all params should be default)

Return type Callable

Module contents

Main API client for Inspire

class `inspirehep.testlib.api.`**InspireApiClient**(*auto_login=True*, *base_url='http://inpirehep.local'*)

Bases: `object`

Inspire Client for end-to-end testing

LOCAL_LOGIN_URL = '/login/?next=%2F&local=1'

login_local(*user='admin@inspirehep.net'*, *password='123456'*)

Perform a local log-in in Inspire storing the session

class `inspirehep.testlib.api.`**Session**(**args*, ***kwargs*)

Bases: `requests.sessions.Session`

get(**args*, ***kwargs*)

get_full_url(**paths*)

post(**args*, ***kwargs*)

put(**args*, ***kwargs*)

static response_to_string(*res*)

Parameters res – `requests.Response` object

Parse the given request and generate an informative string from it

Module contents

Fake arXiv service module

inspirehep.utils package

Submodules

inspirehep.utils.citations module

`inspirehep.utils.citations.`**`get_and_format_citations`**(*record*)
Deprecated since version 2018-08-23.

inspirehep.utils.conferences module

`inspirehep.utils.conferences.`**`conferences_contributions_from_es`**(*cnum*)
Query ES for conferences in the same series.

`inspirehep.utils.conferences.`**`conferences_in_the_same_series_from_es`**(*seriesname*)
Query ES for conferences in the same series.

`inspirehep.utils.conferences.`**`render_conferences`**(*recid*, *conferences*)
Render a list of conferences to HTML.

`inspirehep.utils.conferences.`**`render_conferences_contributions`**(*cnum*)
Conference export for single record in datatables format. :returns: list List of lists where every item represents a datatables row. A row consists of [conference_name, conference_location, contributions, date]

`inspirehep.utils.conferences.`**`render_conferences_in_the_same_series`**(*recid*, *seriesname*)
Conference export for single record in datatables format. :returns: list List of lists where every item represents a datatables row. A row consists of [conference_name, conference_location, contributions, date]

`inspirehep.utils.conferences.`**`render_contributions`**(*hits*)
Render a list of conferences to HTML.

inspirehep.utils.experiments module

`inspirehep.utils.experiments.`**`experiment_contributions_from_es`**(*experiment_name*)
Query ES for conferences in the same series.

`inspirehep.utils.experiments.`**`experiment_people_from_es`**(*experiment_name*)
Query ES for conferences in the same series.

`inspirehep.utils.experiments.`**`render_contributions`**(*hits*)
Render a list of conferences to HTML.

`inspirehep.utils.experiments.`**`render_experiment_contributions`**(*experiment_name*)
Conference export for single record in datatables format. :returns: list List of lists where every item represents a datatables row. A row consists of [conference_name, conference_location, contributions, date]

`inspirehep.utils.experiments.`**`render_experiment_people`**(*experiment_name*)
Conference export for single record in datatables format. :returns: list List of lists where every item represents a datatables row. A row consists of [conference_name, conference_location, contributions, date]

`inspirehep.utils.experiments.`**`render_people`**(*hits*)
Render a list of conferences to HTML.

inspirehep.utils.export module

class `inspirehep.utils.export.`**`Export`**(*record*, **args*, ***kwargs*)

Bases: `object`

Base class used for export formats.

`arxiv_field`

Return arXiv field if exists

exception `inspirehep.utils.export.`**`MissingRequiredFieldError`**(*field*)

Bases: `exceptions.LookupError`

Base class for exceptions in this module. The exception should be raised when the specific, required field doesn't exist in the record.

inspirehep.utils.ext module

Utils extension.

class `inspirehep.utils.ext.`**`INSPIREUtils`**(*app=None*)

Bases: `object`

Utils extension.

`configure_appmetrics`(*app*)

`create_rt_instance`(*app*)

Make a RT instance and return it.

`init_app`(*app*)

Initialize the application.

`inspirehep.utils.ext.`**`exception_hook`**(*response*, *exception*, *metric*, *func_args*, *func_kwargs*)

@time_execution hook to collect info about the raised exception.

inspirehep.utils.jinja2 module

`inspirehep.utils.jinja2.`**`render_template_to_string`**(*input*, *_from_string=False*, ***context*)

Render a template from the template folder with the given context. Code based on https://github.com/mitsuhiko/flask/blob/master/flask/templating.py :param input: the string template, or name of the template to be rendered, or an iterable with template names the first one existing will be rendered :param context: the variables that should be available in the context of the template. :return: a string

inspirehep.utils.latex module

`inspirehep.utils.latex.`**`decode_latex`**(*latex_text*)

Decode latex text.

Parameters **`latex_text`** (*`str`*) – a latex text.

Returns the latex text decoded.

Return type str

inspirehep.utils.lock module

Locking.

exception `inspirehep.utils.lock.`**`DistributedLockError`**
Bases: `exceptions.Exception`

`inspirehep.utils.lock.`**`distributed_lock`**(**args*, ***kwds*)
Context manager to acquire a lock visible by all processes.

This lock is implemented through Redis in order to be globally visible.

Parameters

- **`lock_name`** (*`str`*) – name of the lock to be acquired.
- **`expire`** (*`int`*) – duration in seconds after which the lock is released if not renewed in the meantime.
- **`auto_renewal`** (*`bool`*) – if `True`, the lock is automatically renewed as long as the context manager is still active.
- **`blocking`** (*`bool`*) – if `True`, wait for the lock to be released. If `False`, return immediately, raising *`DistributedLockError`*.

It is recommended to set `expire` to a small value and `auto_renewal=True`, which ensures the lock gets released quickly in case the process is killed without limiting the time that can be spent holding the lock.

Raises *`DistributedLockError`* – when `blocking` is set to `False` and the lock is already acquired.

inspirehep.utils.normalizers module

`inspirehep.utils.normalizers.`**`normalize_journal_title`**(*journal_title*)

inspirehep.utils.proxies module

Utils proxies.

`inspirehep.utils.proxies.`**`rt_instance`** **= <LocalProxy unbound>**
Helper proxy to access the state object.

inspirehep.utils.record module

`inspirehep.utils.record.`**`create_index_op`**(*record*, *version_type='external_gte'*)

`inspirehep.utils.record.`**`get_abstract`**(*record*)
Return the first abstract of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first abstract of the record.

Return type str

Examples

```
>>> record = {
...     'abstracts': [
...         {
...             'source': 'arXiv',
...             'value': 'Probably not.',
```

```
...         },
...     ],
... }
>>> get_abstract(record)
'Probably not.'
```

`inspirehep.utils.record.`**`get_arxiv_categories`**`(`*record*`)`

Return all the arXiv categories of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns all the arXiv categories of the record.

Return type list(str)

Examples

```
>>> record = {
...     'arxiv_eprints': [
...         {
...             'categories': [
...                 'hep-th',
...                 'hep-ph',
...             ],
...             'value': '1612.08928',
...         },
...     ],
... }
>>> get_arxiv_categories(record)
['hep-th', 'hep-ph']
```

`inspirehep.utils.record.`**`get_arxiv_id`**`(`*record*`)`

Return the first arXiv identifier of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first arXiv identifier of the record.

Return type str

Examples

```
>>> record = {
...     'arxiv_eprints': [
...         {
...             'categories': [
...                 'hep-th',
...                 'hep-ph',
...             ],
...             'value': '1612.08928',
...         },
...     ],
... }
>>> get_arxiv_id(record)
'1612.08928'
```

`inspirehep.utils.record.`**`get_collaborations`**`(`*record*`)`

Return the collaborations associated with a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the collaborations associated with the record.

Return type list(str)

Examples

```
>>> record = {'collaborations': [{'value': 'CMS'}]}
>>> get_collaborations(record)
['CMS']
```

`inspirehep.utils.record.`**`get_inspire_categories`**`(`*record*`)`

Return all the INSPIRE categories of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns all the INSPIRE categories of the record.

Return type list(str)

Examples

```
>>> record = {
...     'inspire_categories': [
...         {'term': 'Experiment-HEP'},
...     ],
... }
>>> get_inspire_categories(record)
['Experiment-HEP']
```

`inspirehep.utils.record.`**`get_keywords`**`(`*record*`)`

Return the keywords assigned to a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the keywords assigned to the record.

Return type list(str)

Examples

```
>>> record = {
...     'keywords': [
...         {
...             'schema': 'INSPIRE',
...             'value': 'CKM matrix',
...         },
...     ],
... }
>>> get_keywords(record)
['CKM matrix']
```

`inspirehep.utils.record.`**`get_method`**`(`*record*`)`

Return the acquisition method of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the acquisition method of the record.

Return type str

Examples

```
>>> record = {
...     'acquisition_source': {
...         'method': 'oai',
...         'source': 'arxiv',
...     }
... }
>>> get_method(record)
'oai'
```

`inspirehep.utils.record.`**`get_source`**`(`*record*`)`

Return the acquisition source of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the acquisition source of the record.

Return type str

Examples

```
>>> record = {
...     'acquisition_source': {
...         'method': 'oai',
...         'source': 'arxiv',
...     }
... }
>>> get_source(record)
'arxiv'
```

`inspirehep.utils.record.`**`get_subtitle`**`(`*record*`)`

Return the first subtitle of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first subtitle of the record.

Return type str

Examples

```
>>> record = {
...     'titles': [
...         {
...             'subtitle': 'A mathematical exposition',
...             'title': 'The General Theory of Relativity',
...         },
...     ],
... }
>>> get_subtitle(record)
'A mathematical exposition'
```

`inspirehep.utils.record.`**`get_title`**`(`*record*`)`

Return the first title of a record.

Parameters **`record`** (`InspireRecord`) – a record.

Returns the first title of the record.

Return type str

Examples

```
>>> record = {
...     'titles': [
...         {
...             'subtitle': 'A mathematical exposition',
...             'title': 'The General Theory of Relativity',
...         },
...     ],
... }
>>> get_title(record)
'The General Theory of Relativity'
```

inspirehep.utils.record_getter module

Resource-aware json reference loaders to be used with jsonref.

exception `inspirehep.utils.record_getter.`**`RecordGetterError`**(*message*, *cause*)
Bases: `exceptions.Exception`

`inspirehep.utils.record_getter.`**`get_db_record`**(**args*, ***kwargs*)

`inspirehep.utils.record_getter.`**`get_db_records`**(*pids*)
Get an iterator on record metadata from the DB.

Parameters **pids** (*Iterable[Tuple[str, Union[str, int]]]*) – a list of (pid_type, pid_value) tuples.

Yields *dict* – metadata of a record found in the database.

Warning: The order in which records are returned is different from the order of the input.

`inspirehep.utils.record_getter.`**`get_es_record`**(**args*, ***kwargs*)

`inspirehep.utils.record_getter.`**`get_es_record_by_uuid`**(**args*, ***kwargs*)

`inspirehep.utils.record_getter.`**`get_es_records`**(*pid_type*, *recids*, ***kwargs*)
Get a list of recids from ElasticSearch.

`inspirehep.utils.record_getter.`**`raise_record_getter_error_and_log`**(*f*)

inspirehep.utils.references module

`inspirehep.utils.references.`**`get_and_format_references`**(*record*)
Format references.

Deprecated since version 2018-06-07.

`inspirehep.utils.references.`**`local_refextract_kbs_path`**(**args*, ***kwds*)
Get the path to the temporary refextract kbs from the application config.

`inspirehep.utils.references.`**`map_refextract_to_schema`**(*extracted_references*, *source=None*)
Convert refextract output to the schema using the builder.

inspirehep.utils.robotupload module

Utils for sending robotuploads to other Invenio instances.

`inspirehep.utils.robotupload.`**`make_robotupload_marcxml`**(*url*, *marcxml*, *mode*, ***kwargs*)

Make a robotupload request.

inspirehep.utils.schema module

`inspirehep.utils.schema.`**`ensure_valid_schema`**(*record*)

Make sure the `$schema` key of the record is valid.

This is done by setting the correct url to the schema, in case it only contains the schema filename.

inspirehep.utils.stats module

`inspirehep.utils.stats.`**`calculate_h_index`**(*citations*)

Calculate the h-index of a citation dictionary.

An author has h-index X if she has X papers with at least X citations each. See: https://en.wikipedia.org/wiki/H-index.

Parameters **`citations`** – a dictionary in the format {recid: citation_count}

Returns h-index of the dictionary of citations.

`inspirehep.utils.stats.`**`calculate_i10_index`**(*citations*)

Calculate the i10-index of a citation dictionary.

An author has i10-index X if she has X papers with at least 10 citations each. See: https://en.wikipedia.org/wiki/H-index#i10-index

Parameters **`citations`** – a dictionary in the format {recid: citation_count}

Returns i10-index of the dictionary of citations.

inspirehep.utils.template module

Utils related to Jinja templates.

`inspirehep.utils.template.`**`render_macro_from_template`**(*name*, *template*, *app=None*, *ctx=None*)

Render macro with the given context.

Parameters

- **`name`** (*`string.`*) – macro name.
- **`template`** (*`string.`*) – template name.
- **`app`** (*`object.`*) – Flask app.
- **`ctx`** (*`dict.`*) – parameters of the macro.

Returns unicode string with rendered macro.

inspirehep.utils.tickets module

Functions related to the main INSPIRE-HEP ticketing system.

exception `inspirehep.utils.tickets.`**`EditTicketException`**

Bases: `exceptions.Exception`

class `inspirehep.utils.tickets.`**`InspireRt`**(*url*, *default_login=None*, *default_password=None*, *proxy=None*, *default_queue='General'*, *basic_auth=None*, *digest_auth=None*, *skip_login=False*, *verify_cert=True*)

Bases: `rt.Rt`

`get_attachments`(*ticket_id*)

Get attachment list for a given ticket.

Copy-pased from rt library, only change is starting form 3rd line of response for attachments to look for attachments.

Parameters **`ticket_id`** – ID of ticket

Returns List of tuples for attachments belonging to given ticket. Tuple format: (id, name, content_type, size) Returns None if ticket does not exist.

`inspirehep.utils.tickets.`**`create_ticket`**(**args*, ***kwargs*)

Creates new RT ticket and returns new ticket id.

Parameters

- **`queue`** (`string`) – where the ticket will be created
- **`requestors`** (`string`) – username to set to requestors field of the ticket
- **`body`** (`string`) – message body of the ticket
- **`subject`** (`string`) – subject of the ticket
- **`recid`** (`integer`) – record id to be set custom RecordID field
- **`kwargs`** – Other arguments possible to set:

  Cc, AdminCc, Owner, Status,Priority, InitialPriority, FinalPriority, TimeEstimated, Starts, Due, ... (according to RT fields)

  Custom fields CF.{<CustomFieldName>} could be set with keywords CF_CustomFieldName.

Returns ID of the new ticket or `-1`, if it fails

Return type integer

`inspirehep.utils.tickets.`**`create_ticket_with_template`**(*queue*, *requestors*, *template_path*, *template_context*, *subject*, *recid=None*, ***kwargs*)

Creates new RT ticket with a body that is rendered template

Parameters

- **`queue`** (`string`) – where the ticket will be created
- **`requestors`** (`string`) – username to set to requestors field of the ticket
- **`template_path`** (`string`) – path to the template for the ticket body
- **`template_context`** (`dict`) – context object to be used to render template
- **`subject`** (`string`) – subject of the ticket
- **`recid`** (`integer`) – record id to be set custom RecordID field
- **`kwargs`** – Other arguments possible to set:

  Cc, AdminCc, Owner, Status,Priority, InitialPriority, FinalPriority, TimeEstimated, Starts, Due, ... (according to RT fields)

  Custom fields CF.{<CustomFieldName>} could be set with keywords CF_CustomFieldName.

Returns ID of the new ticket or `-1`, if it fails

Return type integer

`inspirehep.utils.tickets.`**`get_queues`**()

Returns list of all queues as {id, name} dict

Return type dict - with `name (string)`, `id (integer)` properties `inspirehep.utils.tickets.`**`get_rt_link_for_ticket`**`(`*ticket_id*`)`

Returns rt system display link to given ticket

Return type string

`inspirehep.utils.tickets.`**`get_tickets_by_recid`**`(`**args*, ***kwargs*`)`

Returns all tickets that are associated with the given recid

`inspirehep.utils.tickets.`**`get_users`**`()`

Returns list of all users as {id, name} dict

Return type dict - with `name (string)`, `id (integer)` properties

`inspirehep.utils.tickets.`**`relogin_if_needed`**`(`*f*`)`

Repeat RT call after explicit login, if needed.

In case a call to RT fails, due session expired, this decorator will explicitly call .login() on RT, in order to refresh the session, and will replay the call.

This decorator should be used to wrap any function calling into RT.

FIXME: The real solution would be to enable auth/digest authentication on RT side. Then this trick would no longer be needed, as long as the extension is properly initialized in ext.py.

`inspirehep.utils.tickets.`**`reply_ticket`**`(`**args*, ***kwargs*`)`

Replies the given ticket with the message body

Parameters

- **`body`** (*`string`*) – message body of the reply
- **`keep_new`** – flag to keep ticket `Status`, `'new'`

`inspirehep.utils.tickets.`**`reply_ticket_with_template`**`(`*ticket_id*, *template_path*, *template_context*, *keep_new=False*`)`

Replies the given ticket with a body that is rendered template

Parameters

- **`template_path`** (*`string`*) – path to the template for the ticket body
- **`template_context`** (*`dict`*) – context object to be used to render template
- **`keep_new`** – flag to keep ticket `Status`, `'new'`

`inspirehep.utils.tickets.`**`resolve_ticket`**`(`**args*, ***kwargs*`)`

Resolves the given ticket

inspirehep.utils.url module

Helpers for handling with http requests and URL handling.

`inspirehep.utils.url.`**`copy_file`**`(`*src_file*, *dst_file*, *buffer_size=8192*`)`

Dummy buffered copy between open files.

`inspirehep.utils.url.`**`get_legacy_url_for_recid`**`(`*recid*`)`

Get a URL to a record on INSPIRE.

Parameters

- **`recid`** (*`Union[int, string]`*) – record ID
- **`pattern_config_var`** (*`string`*) – config var with the pattern

Returns URL

Return type text_type

`inspirehep.utils.url.`**`is_pdf_link`**`(`*url*`)`

Return `True` if `url` points to a PDF.

Returns `True` if the first non-whitespace characters of the response are `%PDF`.

Parameters **`url`** (*string*) – a URL.
Returns whether the url points to a PDF.
Return type bool

`inspirehep.utils.url.`**`make_user_agent_string`**(*component=’‘*)
Return a nice and uniform user-agent string to be used by INSPIRE.

`inspirehep.utils.url.`**`retrieve_uri`**(**args*, ***kwds*)
Retrieves the given uri and stores it in a temporary file.

Module contents

2.10.2 Submodules

2.10.3 inspirehep.celery module

2.10.4 inspirehep.celery_tests module

2.10.5 inspirehep.cli module

INSPIREHEP CLI app instantiation.

2.10.6 inspirehep.config module

INSPIREHEP app configuration.

`inspirehep.config.`**`COLLECTIONS_DELETED_RECORDS`** **= ‘{dbquery} AND NOT deleted:True’**
Enhance collection query to exclude deleted records.

`inspirehep.config.`**`COLLECTIONS_REGISTER_RECORD_SIGNALS`** **= False**
Don’t register the signals when instantiating the extension.

Since we are instantiating the *invenio-collections* extension two times we don’t want to register the signals twice, but we want to explicitly call *register_signals()* on our own.

`inspirehep.config.`**`COLLECTIONS_USE_PERCOLATOR`** **= False**
Define which percolator you want to use.

Default value is *False* to use the internal percolator. You can also set True to use ElasticSearch to provide percolator resolver. NOTE that ES percolator uses high memory and there might be some problems when creating records.

`inspirehep.config.`**`FEATURE_FLAG_ENABLE_UPDATE_TO_LEGACY`** **= False**
This feature flag will prevent to send a `replace` update to legacy.

`inspirehep.config.`**`HEP_ONTOLOGY_FILE`** **= ‘HEPont.rdf’**
Name or path of the ontology to use for hep articles keyword extraction.

`inspirehep.config.`**`INSPIRE_FULL_THEME`** **= True**
Allows to switch between labs.inspirehep.net view and full version.

`inspirehep.config.`**`INSPIRE_REF_UPDATER_WHITELISTS`** **= {‘literature’: [’accelerator_experiments.record’, ‘authors**
Controls which fields are updated when the referred record is updated.

`inspirehep.config.`**`RECORDS_DEFAULT_FILE_LOCATION_NAME`** **= ‘records’**
Name of default records Location reference.

`inspirehep.config.`**`RECORDS_DEFAULT_STORAGE_CLASS`** **= ‘S’**
Default storage class for record files.

`inspirehep.config.`**`RECORDS_MIGRATION_SKIP_FILES`** **= False**
Disable the downloading of files at record migration time.

Note: This variable takes precedence over `RECORDS_SKIP_FILES`, but can be overriden by the tasks in the `inspirehep.modules.migrator.tasks` module.

`inspirehep.config.`**`RECORDS_SKIP_FILES`** **= False**
Disable the downloading of files at record creation and update times.

Note: The `skip_files` parameter passed to `InspireRecord.create` or `InspireRecord.update` takes precedence on this config variable.

`inspirehep.config.`**`REMEMBER_COOKIE_HTTPONLY`** **= True**
Prevents the “Remember Me” cookie from being accessed by client-side scripts

`inspirehep.config.`**`WORKFLOWS_DEFAULT_FILE_LOCATION_NAME`** **= ‘holdingpen’**
Name of default workflow Location reference.

`inspirehep.config.`**`WORKFLOWS_OBJECT_CLASS`** **= ‘invenio_workflows_files.api.WorkflowObject’**
Enable obj.files API.

2.10.7 inspirehep.factory module

INSPIREHEP app factories.

`inspirehep.factory.`**`api_config_loader`**`(`*app*, ***kwargs_config*`)`

`inspirehep.factory.`**`config_loader`**`(`*app*, ***kwargs_config*`)`

`inspirehep.factory.`**`instance_path`** **= ‘/home/docs/checkouts/readthedocs.org/user_builds/inspirehep/envs/stable/var/i**
Instance path for Invenio.

Defaults to `<env_prefix>_INSTANCE_PATH` or if environment variable is not set `<sys.prefix>/var/<app_name>-instance`.

`inspirehep.factory.`**`static_folder`** **= ‘/home/docs/checkouts/readthedocs.org/user_builds/inspirehep/envs/stable/var/i**
Static folder path.

Defaults to `<env_prefix>_STATIC_FOLDER` or if environment variable is not set `<sys.prefix>/var/<app_name>-instance/static`.

2.10.8 inspirehep.version module

2.10.9 inspirehep.wsgi module

2.10.10 inspirehep.wsgi_with_coverage module

2.10.11 Module contents

INSPIREHEP.

Happy hacking!

INSPIRE Development Team
Email: admin@inspirehep.net
Twitter: @inspirehep
URL: http://inspirehep.net

Python Module Index

Index