aiida-hea

Release 0.1.0a0

Jun 17, 2019

Contents

`aiida-hea` is available at http://github.com/unkcpz/aiida-hea

CHAPTER 1

User guide

1.1 Getting started

This page should contain a short guide on what the plugin does and a short example on how to use the plugin.

1.1.1 Installation

Use the following commands to install the plugin:

```
git clone https://github.com/unkcpz/aiida-hea .
cd aiida-hea
pip install -e .  # also installs aiida, if missing (but not postgres)
#pip install -e .[pre-commit,testing] # install extras for more features
verdi quicksetup  # better to set up a new profile
verdi calculation plugins  # should now show your calclulation plugins
```

Then use `verdi code setup` with the `hea` input plugin to set up an AiiDA code for aiida-hea.

1.1.2 Usage

A quick demo of how to submit a calculation:

```
verdi daemon start         # make sure the daemon is running
cd examples
verdi run test_submit.py        # submit test calculation
verdi calculation list -a  # check status of calculation
```

If you have already set up your own aiida_hea code using `verdi code setup`, you may want to try the following command:

```
hea-submit  # uses aiida_hea.cli
```

1.2 Tutorial

This page can contain a simple tutorial for your code.

1.2.1 What we want to achieve

Step 1

Some text

Step 2

Some other text

1.2.2 The final result

Some text

CHAPTER 2

Developer guide

2.1 Running the tests

The following will discover and run all unit test:

```
pip install -e .[testing]
pytest -v
```

2.2 Automatic coding style checks

Enable enable automatic checks of code sanity and coding style:

```
pip install -e .[pre-commit]
pre-commit install
```

After this, the yapf formatter, the pylint linter and the prospector code analyzer will run at every commit.

If you ever need to skip these pre-commit hooks, just use:

```
git commit -n
```

2.3 Continuous integration

`aiida-hea` comes with a `.travis.yml` file for continuous integration tests on every commit using Travis CI. It will:

1. run all tests for the `django` and `sqlalchemy` ORM
2. build the documentation
3. check coding style and version number (not required to pass by default)

Just enable Travis builds for the `aiida-hea` repository in your Travis account.

`aiida-hea` also includes an `azure-pipelines.yml` file for continuous integration tests using Azure Pipelines.

2.4 Online documentation

The documentation of `aiida-hea` is ready for ReadTheDocs:

Simply add the `aiida-hea` repository on your RTD profile, preferably using `aiida-hea` as the project name - that's it!

2.5 PyPI release

Your plugin is ready to be uploaded to the Python Package Index. Just register for an account and:

```
pip install twine
python setup.py sdist bdist_wheel
twine upload dist/*
```

After this, you (and everyone else) should be able to:

```
pip install aiida-hea
```

CHAPTER 3

aiida_hea package

3.1 Subpackages

3.1.1 aiida_hea.data package

Module contents

Data types provided by plugin

Register data types via the "aiida.data" entry point in setup.json.

class aiida_hea.data.**DiffParameters**(*dict=None*, ***kwargs*)

Bases: aiida.orm.nodes.data.dict.Dict

Command line options for diff.

This class represents a python dictionary used to pass command line options to the executable.

__abstractmethods__ = frozenset([])

__init__(*dict=None*, ***kwargs*)

Constructor for the data class

Usage: `DiffParameters(dict{'ignore-case': True})`

Parameters

- **parameters_dict** (*type*) – dictionary with commandline parameters
- **parameters_dict** – dict

__module__ = 'aiida_hea.data'

__str__()

String representation of node.

Append values of dictionary to usual representation. E.g.:

```
uuid: b416cbee-24e8-47a8-8c11-6d668770158b (pk: 590)
{'ignore-case': True}
```

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 36

_abc_registry = <_weakrefset.WeakSet object>

_logger = <logging.Logger object>

_plugin_type_string = 'data.hea.DiffParameters.'

_query_type_string = 'data.hea.'

cmdline_params (*file1_name*, *file2_name*)

Synthesize command line parameters.

e.g. ['–ignore-case', 'filename1', 'filename2']

Parameters

- **file_name1** (*type*) – Name of first file
- **file_name1** – str
- **file_name2** (*type*) – Name of second file
- **file_name2** – str

schema = <Schema({'ignore-case': <type 'bool'>, 'ignore-tab-expansion': <type 'bool'

validate (*parameters_dict*)

Validate command line options.

Uses the voluptuous package for validation. Find out about allowed keys using:

```
print(DiffParameters).schema.schema
```

Parameters

- **parameters_dict** (*type*) – dictionary with commandline parameters
- **parameters_dict** – dict

Returns validated dictionary

3.1.2 aiida_hea.tests package

Subpackages

aiida_hea.tests.input_files package

Module contents

Submodules

aiida_hea.tests.test_calculations module

Tests for calculations

aiida_hea.tests.test_calculations.**test_process**(*aiida_code*)
Test running a calculation note this does not test that the expected outputs are created of output parsing

aiida_hea.tests.test_cli module

Module contents

Tests for the plugin.

Includes both tests written in unittest style (test_cli.py) and tests written in pytest style (test_calculations.py).

3.2 Submodules

3.3 aiida_hea.calculations module

Calculations provided by aiida_hea.

Register calculations via the "aiida.calculations" entry point in setup.json.

class aiida_hea.calculations.**DiffCalculation**(**args*, ***kwargs*)
Bases: aiida.engine.processes.calcjobs.calcjob.CalcJob

AiiDA calculation plugin wrapping the diff executable.

Simple AiiDA plugin wrapper for 'diffing' two files.

__abstractmethods__ = frozenset([])

__module__ = 'aiida_hea.calculations'

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 36

_abc_registry = <_weakrefset.WeakSet object>

classmethod define(*spec*)
Define inputs and outputs of the calculation.

prepare_for_submission(*folder*)
Create input files.

Parameters **folder** – an *aiida.common.folders.Folder* where the plugin should temporarily place all files needed by the calculation.

Returns *aiida.common.datastructures.CalcInfo* instance

3.4 aiida_hea.cli module

3.5 aiida_hea.helpers module

Helper functions for automatically setting up computer & code.

Helper functions for setting up

1. An AiiDA localhost computer
2. A "diff" code on localhost

Note: Point 2 is made possible by the fact that the `diff` executable is available in the PATH on almost any UNIX system.

`aiida_hea.helpers.`**`get_code`**(*entry_point*, *computer*)
Get local code.

Sets up code for given entry point on given computer.

Parameters

- **`entry_point`** – Entry point of calculation plugin
- **`computer`** – (local) AiiDA computer

Returns The code node

Return type `aiida.orm.Code`

`aiida_hea.helpers.`**`get_computer`**(*name='localhost-test'*, *workdir=None*)
Get AiiDA computer.

Loads computer 'name' from the database, if exists. Sets up local computer 'name', if it isn't found in the DB.

Parameters

- **`name`** – Name of computer to load or set up.
- **`workdir`** – path to work directory Used only when creating a new computer.

Returns The computer node

Return type `aiida.orm.Computer`

`aiida_hea.helpers.`**`get_path_to_executable`**(*executable*)
Get path to local executable.

Parameters **`executable`** (*`str`*) – Name of executable in the $PATH variable

Returns path to executable

Return type str

3.6 aiida_hea.parsers module

Parsers provided by aiida_hea.

Register parsers via the "aiida.parsers" entry point in setup.json.

class aiida_hea.parsers.**DiffParser**(*node*)
Bases: aiida.parsers.parser.Parser

Parser class for parsing output of calculation.

__abstractmethods__ = frozenset([])

__init__(*node*)
Initialize Parser instance

Checks that the ProcessNode being passed was produced by a DiffCalculation.

Parameters

- **node** (*type*) – ProcessNode of calculation
- **node** – aiida.orm.ProcessNode

__module__ = 'aiida_hea.parsers'

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 36

_abc_registry = <_weakrefset.WeakSet object>

parse(***kwargs*)
Parse outputs, store results in database.

Returns an exit code, if parsing fails (or nothing if parsing succeeds)

3.7 Module contents

aiida_hea

AiiDA plugin for generating special quasi-random structures by ATAT/sqs and enumerating derivative superstructures by enumlib.

If you use this plugin for your research, please cite the following work:

> Author Name1, Author Name2, *Paper title*, Jornal Name XXX, YYYY (Year).

If you use AiiDA for your research, please cite the following work:

> Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky, *AiiDA: automated interactive infrastructure and database for computational science*, Comp. Mat. Sci 111, 218-230 (2016); https://doi.org/10.1016/j.commatsci.2015.09.013; http://www.aiida.net.

aiida-hea is released under the MIT license.

Please contact morty.yu@yahoo.com for information concerning aiida-hea and the AiiDA mailing list for questions concerning aiida.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

Symbols

A

C

D