transducer Documentation

Release 0.9

Sixty North

Jun 28, 2017

Contents

`transducer` is a Python package for ...

CHAPTER 1

Contents

Front Matter

Copyright

`transducer`

Official Website

https://github.com/sixty-north/python-transducers

License

Narrative Documentation

Read this to learn how to use `transducer`.

`transducer` Introduction

Current the best description of `transducer` can be found in our series of articles Understanding Transducers Through Python. The code developed over the course of these articles is substantially the same as in this `transducer` package, although the package uses some further abstractions and tools which are largely irrelevant to understanding how transducers work.

Reference Documentation

Descriptions and examples for every public function, class and method in `transducer`.

API Reference

`transducer`

`transducer.transducers`

Functions for creating transducers.

The functions in this module return transducers.

Blah blah blah

Transducer Factories

Call these functions to obtain transducer objects.

mapping	Create a mapping transducer with the given transform.

`mapping`(*transform*)

Create a mapping transducer with the given transform.

Parameters **`transform`** – A single-argument function which will be applied to each input element to produce the corresponding output element.

Returns A mapping transducer: A single argument function which, when passed a reducing function, returns a new reducing function which applies the specified mapping transform function *before* delegating to the original reducer.

Examples

Mapping a squaring function over a list:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import appending
```

```
>>> from transducer.transducers import mapping
>>> m = mapping(lambda x: x*x)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(m, appending(), a)
[1, 49, 9, 4, 3, 2]
```

filtering(*predicate*)

Create a filtering transducer with the given predicate.

Parameters **predicate** – A single-argument function which will be used to test each element and which must return True or False. Only those elements for which this function returns True will be retained.

Returns A filtering transducer: A single argument function which, when passed a reducing function, returns a new reducing function which passes only those items in the input stream which match satisfy the predicate to the original reducer.

Examples

Filtering even numbers from a list:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import appending
>>> from transducer.transducers import filtering
>>> f = filtering(lambda x: x % 2 == 0)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(f, appending(), a)
[4, 2]
```

reducing(*reducer*, *init=UNSET*)

Create a reducing transducer with the given reducer.

Parameters **reducer** – A two-argument function which will be used to combine the partial cumulative result in the first argument with the next item from the input stream in the second argument.

Returns A reducing transducer: A single argument function which, when passed a reducing function, returns a new reducing function which entirely reduces the input stream using 'reducer' before passing the result to the reducing function passed to the transducer.

Examples

Reducing a list of numbers to a single value by summing them:

```
>>> from transducer.eager import transduce
>>> from transducer.reducers import expecting_single
>>> from transducer.transducers import reducing
>>> r = reducing(lambda x, y: x + y)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(r, expecting_single(), a)
>>> 26
```

transducer.eager

Blah blah blah

---

transduce

---

transduce(*transducer*, *reducer*, *iterable*, *init=UNSET*)

Examples

Eager transduction of a mapping over a list:

```
>>> from transducer.eager import transducer
>>> from transducer.reducers import appending
>>> from transducer.transducers import mapping
>>> m = mapping(lambda x: x*x)
>>> a = [1, 7, 9, 4, 3, 2]
>>> transduce(mapping, appending(), a)
[1, 49, 9, 4, 3, 2]
```

Differences from Clojure's transducers

Although `transducer` is inspired by Clojure's transducers, there are inevitably some differences in order to accommodate the variance between Clojure and Python.

Frequently Asked Questions

So far, there are none.

Detailed Change History

Changes

transducer 0.8

Reorganised the code – particularly for the coroutine based event processing: sources, sinks and the react transduce now exist in their own modules.

Clarified the semantics of the sources, which now no longer call close() on the targets.

Single-sourced the version from the transducer/__init__.py file.

Adds a new reducer for producing sets.

A new reducer for completing regular Python reducing functions into objects supporting the full reducer protocol.

Numerous fixes throughout the code as a result of vastly improved test coverage.

transducer 0.7

Rework to make the code closer to the code in the blog series.

Much clearer reduction to single values for transducers such as first() and last() in conjunction with the expecting_single() reducer.

Simplified and clarified much of the implementation.

Dropped support for Python 3.2.

transducer 0.6

Correctness fixes by moving function locals in the transducer factories to be class attributes of the transducers. This allows transducers returned by the transducer factories to be safely used more than once.

Renamed transducer.transducer to transducer.transducers.

Relocate reducing functions to reducers.py

Move transducer infrastructure to infrastructure.py

transducer 0.5

Initial release

Samples

More complex examples of non-trivial usage of `transducer`:

Samples

One day ...

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index