coercion

Release 1.0

November 03, 2015

Contents

This library provides functions that coerce datastructures into normalized forms. For example, converting an arbitrary `dict` into a form that is suitable for passing to `json.dumps`.

The tornado framework has a function called recursive_unicode in the `tornado.escape` module. It is a very simple recursive walk of datastructure that switches on type and transforms string values into unicode strings. I use this in production software regularly and it works like a charm. Or at least it did until my software encountered a deeply nested dictionary and I received a `RuntimeError: maximum recursion depth exceeded` error in my service log. This is one of the exceptions that strikes fear into most engineers when it rears it's head in production.

That is the primary reason for this library existing. It provides the same simple string encoding function iteratively instead of recursively. At the same time, the need to coerce values into a normalized string form is something that I've had to do repeatedly so it might as well be plopped into a reusable library.

CHAPTER 1

Examples

The following example shows one of the underlying reasons that this library was created. The commonly used msgpack implementation for python returns everything as byte strings which is problematic if you want to dump it as JSON since it will raise a `TypeError` if dictionary keys are not strings. (This is where recursive_unicode was so handy.)

```
>>> import json
>>> import coercion
>>> import msgpack
>>> bin_msg = msgpack.packb({u'\u00DCnicode': b'bytes', b'bytes': 'str'})
>>> decoded = msgpack.unpackb(bin_msg)
>>> decoded
{b'bytes': b'str', b'\xc3\x9cnicode': b'bytes'}
>>> json.dumps(decoded)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/daveshawley/opt/lib/python3.5/json/__init__.py", line 230, in dumps
    return _default_encoder.encode(obj)
  File "/Users/daveshawley/opt/lib/python3.5/json/encoder.py", line 199, in encode
    chunks = self.iterencode(o, _one_shot=True)
  File "/Users/daveshawley/opt/lib/python3.5/json/encoder.py", line 257, in iterencode
    return _iterencode(o, 0)
TypeError: keys must be a string
>>> json.dumps(coercion.normalize_collection(decoded))
'{"bytes": "str", "\\u00dcnicode": "bytes"}'
```

1.1 API Reference

`coercion.__version__` = '1.0.0'
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`coercion.version_info` = (1, 0, 0)
tuple() -> empty tuple tuple(iterable) -> tuple initialized from iterable's items

If the argument is a tuple, the return value is the same object.

`coercion.stringify`(*obj*)
Return the string representation of an object.

Parameters **obj** – object to get the representation of

Returns unicode string representation of *obj* or *obj* unchanged

This function returns a string representation for many of the types from the standard library. It does not convert numeric or Boolean values to strings – it only converts non-primitive instances such as `datetime.datetime`. The following table describes the types that are handled and describes how they are represented.

Class	Behavior
`uuid.UUID`	`str(obj)`
`datetime.datetime`	`obj.strftime('%Y-%m-%dT%H:%M:%S.%f%z')`
`memoryview`	`obj.tobytes().decode('utf-8')`
`bytearray`	`bytes(obj).decode('utf-8')`
`buffer`	`bytes(obj).decode('utf-8')`
`bytes`	`obj.decode('utf-8')`

Other types are returned unharmed.

`coercion.`**`normalize_collection`**(*coll*)

Normalize all elements in a collection.

Parameters **`coll`** – the collection to normalize. This is required to implement one of the following protocols: `collections.Mapping`, `collections.Sequence`, or `collections.Set`.

Returns a new instance of the input class with the keys and values normalized via *`stringify()`*

Raises `RuntimeError` if *coll* is not a collection

This function transforms the collection by recursively transforming each key and value contained in it. The action is recursive but the implementation is unrolled and iterative. If you are interested in the algorithm used, it is described as comments in the code.

1.2 Changelog

1.2.1 1.0.0 (2015-Nov-02)

- Add *`coercion.stringify()`*
- Add *`coercion.normalize_collection()`*

Symbols

N

S

V