swpt_lib Documentation

Release 0.1

Evgeni Pandurski

Jan 15, 2021

Contents:

CHAPTER 1

swpt_lib.endpoints

Build and match URLs to endpoints.

The available endpoints are:

authority /authority

debtor /debtors/<i64:debtorId>/

creditor /creditors/<i64:creditorId>/

account /creditors/<i64:creditorId>/debtors/<i64:debtorId>

exception swpt_lib.endpoints.**BuildError**
An URL can not be build for the endpoint.

exception swpt_lib.endpoints.**MatchError**
The URL does not match the endpoint.

swpt_lib.endpoints.**build_url**(*endpoint*, ***kw*)
Try to build an absolute URL for a given endpoint and arguments.

Parameters

- **endpoint** (*str*) – The name of the endpoint
- **kw** – The keyword arguments required by the particular endpoint

Returns The absolute URL

Raises *BuildError* if an URL can not be build for the endpoint.

swpt_lib.endpoints.**get_server_name**()
Return site's domain name (and maybe port), or *None* if not set.

The site's domain name and port can be configured by setting the *SWPT_SERVER_NAME* environment variable.

swpt_lib.endpoints.**get_url_scheme**()
Return site's URL scheme, or "http" if not set.

The site's URL scheme can be configured by setting the *SWPT_URL_SCHEME* environment variable.

swpt_lib.endpoints.**match_url**(*endpoint*, *url*)
Try to to match an absolute URL to given endpoint.

Parameters

- **endpoint** (*str*) – The name of the endpoint
- **url** (*str*) – The absolute URL that should be matched

Returns A *dict* of arguments extracted from the URL

Raises *MatchError* if the URL does not match the endpoint.

CHAPTER 2

swpt_lib.utils

class swpt_lib.utils.**Int64Converter**(*map*)
Flask URL converter for signed 64-bit integers.

The converter can be registered with the Flask app like this:

```
from flask import Flask
from swpt_lib.utils import Int64Converter

app = Flask(__name__)
app.url_map.converters['i64'] = Int64Converter
```

class swpt_lib.utils.**Seqnum**(*value: int*)
A signed 32-bit integer seqnum value.

Comparisions beteen *Seqnum* instances correctly deal with the possible 32-bit integer wrapping.

increment() → swpt_lib.utils.Seqnum
Return an incremented instance.

swpt_lib.utils.**date_to_int24**(*d: datetime.date*) → int
Return a non-negative 24-bit integer derived from a date.

The passed date must not be before January 1st, 1970. The returned integer equals the number of days passed since January 1st, 1970.

swpt_lib.utils.**get_config_value**(*key: str*) → Optional[str]
Get the value for the configuration variable with a name *key*.

The returned value is either a string or *None*. If there is a *Flask* application context, the app's config will be checked first. If that fails, the environment will be checked next. If that fails too, *None* will be returned.

swpt_lib.utils.**i64_to_u64**(*value: int*) → int
Convert a signed 64-bit integer to unsigned 64-bit integer.

Raises *ValueError* if the value is not in the range of signed 64-bit integers.

swpt_lib.utils.**increment_seqnum**(*n: int*) → int
Increment a 32-bit signed integer with wrapping.

swpt_lib.utils.**is_later_event**(*event: Tuple[datetime.datetime, int], other_event: Tuple[Optional[datetime.datetime], Optional[int]]*) → bool

Return whether *event* is later than *other_event*.

Each of the passed events must be a (*datetime*, *int*) tuple. The *datetime* must be the event timestamp, and the *int* must be the event sequential number (32-bit signed integer, with eventual wrapping).

An event with a noticeably later timestamp (>= 1s) is always considered later than an event with an earlier timestamp. Only when the two timestamps are very close (< 1s), the sequential numbers of the events are compared. When the timestamp of *other_event* is *None*, *event* is considered as a later event.

Note that sequential numbers are compared with possible 32-bit signed integer wrapping in mind. For example, compared to 2147483647, -21474836478 is considered a later sequential number.

swpt_lib.utils.**u64_to_i64**(*value: int*) → int

Convert an unsigned 64-bit integer to a signed 64-bit integer.

Raises *ValueError* if the value is not in the range of unsigned 64-bit integers.

CHAPTER 3

swpt_lib.scan_table

class swpt_lib.scan_table.**TableScanner**

A table-scanner super-class. Sub-classes may override class attributes.

Exapmle:

```
from swpt_lib.scan_table import TableScanner
from mymodels import Customer

class CustomerScanner(TableScanner):
    table = Customer.__table__
    columns = [Customer.id, Customer.last_order_date]

    def process_rows(self, rows):
        for row in rows:
            print(row['id'], row['last_order_date'])
```

blocks_per_query = 40

The number of database pages (blocks) to be retrieved per query. It might be a good idea to increase this number when the size of the table row is big.

columns = None

An optional list of `sqlalchemy.sql.expression.ColumnElement` instances to be be retrieved for each row. Most of the time it will be a list of `Column` instances. Defaults to all columns.

process_rows(*rows: list*) → None

Process a list or rows.

Must be defined in the subclass.

Parameters **rows** – A list of table rows.

run(*engine: sqlalchemy.engine.interfaces.Connectable*, *completion_goal: datetime.timedelta*, *quit_early: bool = False*)

Scan table continuously.

The table is scanned sequentially, starting from a random row. During the scan *process_rows()* will be continuously invoked with a list of rows. When the end of the table is reached, the scan continues from

the beginning, ad infinitum.

Parameters

- **engine** – SQLAlchemy engine
- **completion_goal** – The time interval in which the whole table should be processed. This is merely an approximate goal. In reality, scans can take any amount of time.
- **quit_early** – Exit after some time. This is mainly useful during testing.

table = None

The `sqlalchemy.schema.Table` that will be scanned. (`Model.__table__` if declarative base is used.)

Must be defined in the subclass.

target_beat_duration = 25

The scanning of the table is done in a sequence of "beats". This attribute determines the ideal duration in milliseconds of those beats. The value should be big enough so that, on average, all the operations performed on table's rows could be completed within this interval. Setting this value too high may have the effect of too many rows being processed simultaneously in one beat.

CHAPTER 4

swpt_lib.swpt_uris

swpt_lib.swpt_uris.**make_account_uri**(*debtor_id: int*, *account_id: str*) → str
Return a valid SWPT account URI.

Raises *ValueError* if the passed *debtor_id* or *account_id* is invalid.

swpt_lib.swpt_uris.**make_debtor_uri**(*debtor_id: int*) → str
Return a valid SWPT debtor URI.

Raises *ValueError* if the passed *debtor_id* is invalid.

swpt_lib.swpt_uris.**parse_account_uri**(*uri: str*) → Tuple[int, str]
Return a (debtor ID, account ID) tuple.

Raises *ValueError* if the passed URI does not represent a valid SWPT account.

swpt_lib.swpt_uris.**parse_debtor_uri**(*uri: str*) → int
Return a debtor ID.

Raises *ValueError* if the passed URI does not represent a valid SWPT debtor.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index