sprockets.clients.postgresql

Release 2.0.1

November 14, 2014

Contents

The `sprockets.clients.postgresql` package wraps the Queries package providing environment variable based configuration for connecting to PostgreSQL.

CHAPTER 1

Installation

sprockets.clients.postgresql is available on the Python Package Index and can be installed via pip or easy_install:

```
pip install sprockets.clients.postgresql
```

CHAPTER 2

Requirements

- queries
- sprockets

CHAPTER 3

API Documentation

3.1 PostgreSQL Session API

The Session classes wrap the Queries `Session` and `TornadoSession` classes providing environment variable based configuration.

Environment variables should be set using the `PGSQL[_DBNAME]` format where the value is a PostgreSQL URI.

For PostgreSQL URI format, see:

http://www.postgresql.org/docs/9.3/static/libpq-connect.html#LIBPQ-CONNSTRING

As example, given the environment variable:

```
PGSQL_FOO = 'postgresql://bar:baz@foohost:6000/foo'
```

and code for creating a `Session` instance for the database name `foo`:

```
session = sprockets.postgresql.Session('foo')
```

A `queries.Session` object will be created that connects to Postgres running on `foohost`, port `6000` using the username `bar` and the password `baz`, connecting to the `foo` database.

3.2 Session Classes

class sprockets.clients.postgresql.**Session**(*dbname*, *cursor_factory=<class 'psycopg2.extras.RealDictCursor'>*, *pool_idle_ttl=60*, *pool_max_size=1*, *db_url=None*)

Extends queries.Session using configuration data that is stored in environment variables.

Utilizes connection pooling to ensure that multiple concurrent asynchronous queries do not block each other. Heavily trafficked services will require a higher `max_pool_size` to allow for greater connection concurrency.

Parameters

- **dbname** (*str*) – PostgreSQL database name
- **queries.cursor** – The cursor type to use
- **pool_idle_ttl** (*int*) – How long idle pools keep connections open
- **pool_max_size** (*int*) – The maximum size of the pool to use

- **db_url** (*str*) – Optional database connection URL. Use this when you need to connect to a database that is only known at runtime.

backend_pid
Return the backend process ID of the PostgreSQL server that this session is connected to.

Return type int

callproc(*name*, *args=None*)
Call a stored procedure on the server, returning the results in a `queries.Results` instance.

Parameters

- **name** (*str*) – The procedure name
- **args** (*list*) – The list of arguments to pass in

Return type queries.Results

Raises queries.DataError

Raises queries.DatabaseError

Raises queries.IntegrityError

Raises queries.InternalError

Raises queries.InterfaceError

Raises queries.NotSupportedError

Raises queries.OperationalError

Raises queries.ProgrammingError

close()
Explicitly close the connection and remove it from the connection pool if pooling is enabled. If the connection is already closed

Raises psycopg2.InterfaceError

connection
The current open connection to PostgreSQL.

Return type psycopg2.extensions.connection

cursor
The current, active cursor for the open connection.

Return type psycopg2.extensions.cursor

encoding
The current client encoding value.

Return type str

notices
A list of up to the last 50 server notices sent to the client.

Return type list

pid
Return the pool ID used for connection pooling

Return type str

query(*sql*, *parameters=None*)
A generator to issue a query on the server, mogrifying the parameters against the sql statement. Results are returned as a `queries.Results` object which can act as an iterator and has multiple ways to access the result data.

Parameters

- **sql** (*str*) – The SQL statement
- **parameters** (*dict*) – A dictionary of query parameters

Return type queries.Results

Raises queries.DataError

Raises queries.DatabaseError

Raises queries.IntegrityError

Raises queries.InternalError

Raises queries.InterfaceError

Raises queries.NotSupportedError

Raises queries.OperationalError

Raises queries.ProgrammingError

set_encoding(*value='UTF8'*)
Set the client encoding for the session if the value specified is different than the current client encoding.

Parameters **value** (*str*) – The encoding value to use

class sprockets.clients.postgresql.**TornadoSession**(*dbname*, *cursor_factory=<class 'psycopg2.extras.RealDictCursor'>*, *pool_idle_ttl=60*, *pool_max_size=25*, *io_loop=None*, *db_url=None*)
Extends queries.TornadoSession using configuration data that is stored in environment variables.

Utilizes connection pooling to ensure that multiple concurrent asynchronous queries do not block each other. Heavily trafficked services will require a higher `max_pool_size` to allow for greater connection concurrency.

`query` and `callproc` must call `Results.free`

Parameters

- **dbname** (*str*) – PostgreSQL database name
- **queries.cursor** – The cursor type to use
- **pool_idle_ttl** (*int*) – How long idle pools keep connections open
- **pool_max_size** (*int*) – The maximum size of the pool to use
- **ioloop** (*tornado.ioloop.IOLoop*) – Pass in the instance of the tornado IOLoop you would like to use. Defaults to the global instance.
- **db_url** (*str*) – Optional database connection URL. Use this when you need to connect to a database that is only known at runtime.

backend_pid
Return the backend process ID of the PostgreSQL server that this session is connected to.

Return type int

callproc(*name*, *args=None*)
Call a stored procedure asynchronously on the server, passing in the arguments to be passed to the stored procedure, yielding the results as a `Results` object.

You **must** free the results that are returned by this method to unlock the connection used to perform the query. Failure to do so will cause your Tornado application to run out of connections.

Parameters

- **name** (*str*) – The stored procedure name
- **args** (*list*) – An optional list of procedure arguments

Return type Results

Raises queries.DataError

Raises queries.DatabaseError

Raises queries.IntegrityError

Raises queries.InternalError

Raises queries.InterfaceError

Raises queries.NotSupportedError

Raises queries.OperationalError

Raises queries.ProgrammingError

close()
Explicitly close the connection and remove it from the connection pool if pooling is enabled. If the connection is already closed

Raises psycopg2.InterfaceError

connection
Do not use this directly with Tornado applications

Returns

encoding
The current client encoding value.

Return type str

notices
A list of up to the last 50 server notices sent to the client.

Return type list

pid
Return the pool ID used for connection pooling

Return type str

query(*sql*, *parameters=None*)
Issue a query asynchronously on the server, mogrifying the parameters against the sql statement and yielding the results as a `Results` object.

You **must** free the results that are returned by this method to unlock the connection used to perform the query. Failure to do so will cause your Tornado application to run out of connections.

Parameters

- **sql** (*str*) – The SQL statement

- **parameters** (*dict*) – A dictionary of query parameters

Return type Results

Raises queries.DataError

Raises queries.DatabaseError

Raises queries.IntegrityError

Raises queries.InternalError

Raises queries.InterfaceError

Raises queries.NotSupportedError

Raises queries.OperationalError

Raises queries.ProgrammingError

set_encoding(*value='UTF8'*)

Set the client encoding for the session if the value specified is different than the current client encoding.

Parameters **value** (*str*) – The encoding value to use

validate()

Validate the session can connect or has open connections to PostgreSQL

Return type bool

3.3 Examples

The following example sets the environment variables for connecting to PostgreSQL on localhost to the `postgres` database and issues a query.

```
import os

from sprockets.clients import postgresql

os.environ['PGSQL'] = 'postgresql://postgres@localhost:5432/postgres'

session = postgresql.Session()
result = session.query('SELECT 1')
print(repr(result))
```

The following example shows how to use the `TornadoSession` class in a Tornado `RequestHandler`.

```
import os

from tornado import gen
from sprockets.clients import postgresql
from tornado import web

os.environ['PGSQL_FOO'] = 'postgresql://postgres@localhost:5432/foo'

class RequestHandler(web.RequestHandler):

    def initialize(self):
        self.session = postgresql.TornadoSession('foo')

    @gen.coroutine
```

```
    def get(self, *args, **kwargs):
        result = yield self.session.query('SELECT 1')
        self.write(result.as_dict())
        result.free()
```

CHAPTER 4

Version History

See `history`

CHAPTER 5

Issues

Please report any issues to the Github project at https://github.com/sprockets/sprockets.clients.postgresql/issues

CHAPTER 6

Source

sprockets.clients.postgresql source is available on Github at https://github.com/sprockets/sprockets.clients.postgresql

CHAPTER 7

License

sprockets.clients.postgresql is released under the 3-Clause BSD license.

CHAPTER 8

Indices and tables

- *genindex*
- *modindex*
- *search*

S

B

C

E

N

P

Q

S

T

V