sqla$_{e}xt$Documentation

Release 0.0.0

Oliver Rice

Apr 16, 2021

CONTENTS

sqla_ext is a lightweight library providing extensions to SQLAlchemy.

CHAPTER

ONE

GETTING STARTED

Requirements

Python 3.6+

Installation

```
$ pip install sqla_ext
```

CHAPTER

TWO

CONTENTS

2.1 Welcome to sqla_ext

sqla_ext is a lightweight library providing extensions to SQLAlchemy.

2.1.1 Getting Started

Requirements

Python 3.6+

Installation

```
$ pip install sqla_ext
```

2.2 API Reference

2.2.1 JSON Functions

2.2.1.1 json.agg

`sqla_ext.func.json.`**`agg`**(*expression: Union[sqlalchemy.sql.elements.ColumnClause, sqlalchemy.sql.elements.TextClause]*) → None

JSON array aggregation

Dialects

- mysql
- postgresql
- sqlite

Parameters **`expression`** – A SQL expression, such as a `ColumnElement` expression or a `TextClause` with elements that are castable as JSON

Returns `FuntionElement`

E.g.:

```
from sqlalchemy import select, Integer, text, table, column
from sqla_ext import func as func_ext

t = table('some_table', column('q', Integer))

query = select([
    func_ext.json.agg(t.q)
])
```

The above statement will produce SQL resembling:

```
SELECT
    jsonb_agg(some_table.q)
FROM
    some_table
```

2.2.1.2 json.build_object

sqla_ext.func.json.**build_object**(**clauses*, ***kwargs*)

JSON object creation

Dialects

- mysql
- postgresql
- sqlite

Parameters **expression** – A SQL expression, such as a `ColumnElement` expression or a `TextClause` with elements that are castable as JSON

Returns `FuntionElement`

E.g.:

```
from sqlalchemy import select
from sqla_ext import func as func_ext

query = select([
    func_ext.json.build_object("first_key", "first_value", "second_key", 2)
])
```

The above statement will produce SQL resembling:

```
SELECT
    jsonb_build_object('first_key', 'first_value', 'second_key', 2)
```

2.2.1.3 json.to_array

sqla_ext.func.json.**to_array**(*expression: Union[ColumnClause, TextClause]*, *type_: TypeEngine*) → None

Cast a json array to a native array

Dialects

- postgresql

Parameters

- **expression** – A SQL expression, such as a `ColumnElement` expression or a `TextClause` that can be coerced into a JSONB array
- **type_** – A `TypeEngine` class or instance indicating the type to `CAST` elements of the native array as.

Returns `FuntionElement`

E.g.:

```
from sqlalchemy import select, Integer, text
from sqla_ext import func as func_ext

query = select([
    func_ext.json.to_array(text("'[1,2,3,4]'::jsonb"), Integer)
])
```

The above statement will produce SQL resembling:

```
SELECT
    array_agg(CAST(anon_1 AS INTEGER)) AS array_agg_1
FROM
    jsonb_array_elements(CAST('[1,2,3,4]'::jsonb AS JSONB)) AS anon_1
```

2.2.2 Datetime Functions

2.2.2.1 datetime.utc_now

sqla_ext.func.datetime.**utc_now**() → None

Current timestamp in UTC timezone

Dialects

- mysql
- postgresql
- sqlite

Returns `FuntionElement`

E.g.:

```
from sqlalchemy import select
from sqla_ext import func as func_ext

query = select([
    func_ext.datetime.utc_now()
])
```

The above statement will produce SQL resembling:

```
SELECT
    timezone('utc', current_timestamp)
```

2.2.3 Inspect

2.2.3.1 to_core_table

sqla_ext.inspect.**to_core_table**(*entity: Union[sqlalchemy.sql.schema.Table, sqla_ext.protocols.ORMTableProtocol, sqlalchemy.orm.mapper.Mapper, Callable[], Union[sqla_ext.protocols.ORMTableProtocol, sqlalchemy.sql.schema.Table]]]*) → sqlalchemy.sql.schema.Table

Coerces multiple SQLA Table-like entities to a core sqlalcehmy.Table

Parameters **entity** – A sqlalchemy `Table`, `DeclarativeBase` or `Mapper`

Returns `Table`

2.2.3.2 to_schema_name

sqla_ext.inspect.**to_schema_name**(*entity: Union[sqlalchemy.sql.schema.Table, sqla_ext.protocols.ORMTableProtocol, sqlalchemy.orm.mapper.Mapper, Callable[], Union[sqla_ext.protocols.ORMTableProtocol, sqlalchemy.sql.schema.Table]]]*) → Optional[str]

Get the schema of a sqlalchemy table-like entity

Parameters **entity** – A sqlalchemy `Table`, `DeclarativeBase` or `Mapper`

Returns `Optional[str]`

2.2.3.3 to_table_name

sqla_ext.inspect.**to_table_name**(*entity: Union[sqlalchemy.sql.schema.Table, sqla_ext.protocols.ORMTableProtocol, sqlalchemy.orm.mapper.Mapper, Callable[], Union[sqla_ext.protocols.ORMTableProtocol, sqlalchemy.sql.schema.Table]]]*) → str

Get the name of a sqlalchemy table-like entity

Parameters **entity** – A sqlalchemy `Table`, `DeclarativeBase` or `Mapper`

Returns `str`

2.2.4 Types

2.2.4.1 CITEXT

class sqla_ext.types.postgresql.**CITEXT**

Case Insensitive Text Type

Dialects

- postgresql

https://www.postgresql.org/docs/current/citext.html

E.g.:

```
from sqlalchemy import Column, Integer
from sqlalchemy.orm import declarative_base
from sqla_ext.types.postgresql import CITEXT


class User(Base):
    __tablename__ = 'user'
    id = sa.Column(Integer, primary_key=True)
    email = sa.Column(CITEXT())


user = User(id=1, email='John.Smith@example.com')
session.add(user)
session.commit()

# Note: query email is lowercase
user = (
    session.query(User)
    .filter(User.email == 'john.smith@example.com')
    .one()
)
assert user.id == 1
```

2.3 License

sqla_ext is under the MIT License. See the LICENSE file.

2.3.1 Conditions for Contributors

By contributing to this software project, you are agreeing to the following terms and conditions for your contributions: First, you agree your contributions are submitted under the MIT license. Second, you represent you are authorized to make the contributions and grant the license. If your employer has rights to intellectual property that includes your contributions, you represent that you have received permission to make contributions and grant the required license on behalf of that employer.

2.4 Testing

Install:

```
$ pip install -e ".[dev]"
```

Run the tests:

```
$ pytest
```

2.5 Changelog

2.5.1 0.0.0

- Added func.json.build_object for building json objects in the database
- Added func.json.agg for aggregating data as json on the database
- Added func.datetime.utc_now as a unified interface for current timestamp in UTC timezone across dbms

2.5.2 0.0.1

- Test all dialects of *func* functions
- Minimal docstrings
- Initail RTD documentation

2.5.3 0.0.2

- Added func.json.to_array for casting json/jsonb arrays to native arrays

2.5.4 0.0.3

- Doc fixes
- Added *sqla_ext.types* module for sharing type definitions
- Added *sqla_ext.inspect* module for extracting info from sqlalchemy entities
- Added *inspect.to_core_table* for casting multiple table-like entities to a *sqlalchemy.Table*
- Added *inspect.to_table_name* for getting a table name from a table-like entity
- Added *inspect.to_schema_name* for getting a schema name from a table-like entity

2.5.5 0.0.4

- Added *sqla_ext.types.postgresql* module for sharing postgres type definitions
- Added *sqla_ext.types.postgresql.CITEX* for case insensitive strings

2.5.6 Master

- Nothing yet

PYTHON MODULE INDEX

S

INDEX