sphinxalchemy Documentation

Release 0.5.1

Andrey Popp

September 09, 2016

Contents

This package provides a dialect for SQLAlchemy for interfacing with Sphinx search engine via SphinxQL. It allows reusing core SQLAlchemy parts to generate SphinxQL language constructs programmatically via `sqlalchemy.sql` and to maintain a pool of connections to Sphinx via `sqlalchemy.pool`.

CHAPTER 1

Basic usage

You need sphinxalchemy to be installed to allow SQLAlchemy to pick up SphinxQL dialect classes. To install sphinxalchemy do:

```
% pip install sphinxalchemy
```

or if you like `easy_install` command better:

```
% easy_install sphinxalchemy
```

Now, as Sphinx uses MySQL protocol for interfacing via SphinxQL, you need to decide which one of supported by sphinxalchemy MySQL connectivity libraries to use – MySQLdb or mysqlconnector-python. Install one of those and use:

```
sphinx+mysqldb://user@host:port
```

to connect to Sphinx using MySQLdb or:

```
sphinx+mysqlconnector://user@host:port
```

to use mysqlconnector-python. You can now create engine using `sqlalchemy.create_engine()` function as usually:

```
from sqlalchemy import create_engine, MetaData

engine = create_engine("sphinx+mysqlconnector://user@host:port")
metadata = MetaData(bind=engine)
```

To define indexes (analogs of tables in relational databases) you should use *sphinxalchemy.schema* module:

```
from sphinxalchemy.schema import Index, Attribute, ArrayAttribute

documents = Index("documents", metadata,
  Attribute("created"),
  ArrayAttribute("tag_ids"))
```

Now you can query your `documents` index:

```
results = engine.execute(
  documents.select()
    .match("We think in generalities, but we live in details")
    .where(documents.c.tag_ids.in_([42, 1])))
```

CHAPTER 2

API reference

class sphinxalchemy.schema.**Index**(**args*, ***kwargs*)
Sphinx index metadata

Accepted arguments are the same as sqlalchemy.schema.Table accepts.

class sphinxalchemy.schema.**Attribute**(**args*, ***kwargs*)
Sphinx index scalar attribute metadata

Accepted arguments are the same as sqlalchemy.schema.Column accepts.

class sphinxalchemy.schema.**ArrayAttribute**(**args*, ***kwargs*)
Sphinx index array attribute metadata

Accepted arguments are the same as sqlalchemy.schema.Column accepts.

class sphinxalchemy.sphinxql.**Select**(*columns*, **args*, ***kwargs*)
SphinxQL SELECT construct.

See corresponding doc section.

match(*query*)
Provide full text query for index

Sphinx uses extended query syntax in SphinxQL.

options(**args*, ***kwargs*)
Provide options to execute query

Available options described here.

class sphinxalchemy.sphinxql.**Replace**(*table*, *values=None*, *inline=False*, *bind=None*, *prefixes=None*, *returning=None*, ***kwargs*)
Represent an REPLACE construct.

The *Replace* object is created using the *replace()* function.

See also:

Insert Expressions

sphinxalchemy.sphinxql.**select**(*columns=None*, *whereclause=None*, *from_obj=[]*, ***kwargs*)
Factory for creating *Select* constructs.

Ressembles sqlalchemy.sql.expression.select().

sphinxalchemy.sphinxql.**replace**(*table*, *values=None*, *inline=False*, ***kwargs*)

S

A

I

M

O

R

S