
sqli-filters-yaml Documentation

Release 0.0.1

Marc-Aurele Coste

Dec 18, 2019

Contents:

1	YAML Parser	3
2	Usage	5
3	Indices and tables	7

Warning: This project is not ready for production so use it carefully because it's not stable.

CHAPTER 1

YAML Parser

This parser is used to create a filters tree using a yaml file.

CHAPTER 2

Usage

For the example we will take a simple entity like the following:

```
class Post(Base):
    p_id = sa.Column(sa.Integer, primary_key=True)
    title = sa.Column(sa.String(100))
    content = sa.Column(sa.String)

    def __str__(self):
        return '{} | {}'.format(self.title, self.content)
```

and your YAML looks like:

```
type: or
data:
- type: operator
  data:
    attribute: title
    operator: eq
    value: Post_01
- type: operator
  data:
    attribute: title
    operator: eq
    value: Post_03
```

you can now use `sqla-filter` to filter your query and get only post with title equal to 'Post_01' or 'Post_02'.

```
# Create a YAML parser instance
parser = YAMLFiltersParser(raw_yaml_string)

# A tree is generated with the YAML received.
# If you set the YAML the tree is automatically updated.
print(parser.tree)
```

(continues on next page)

(continued from previous page)

```
# Now you can filter a query
query = session.query(Post)
filtered_query = parser.tree.filter(query)

# Get the results
# you can also directly call the `all()` from previous step
# results = filtered_query = parser.tree.filter(query).all()
results = query.all()
```

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`