Watson - DB

Release 1.1.0

September 30, 2014

Contents

SqlAlchemy integration for Watson-Framework.

CHAPTER 1

Build Status

CHAPTER 2

Dependencies

- watson-framework
- sqlalchemy

CHAPTER 3

Installation

```
pip install watson-db
```

CHAPTER 4

Testing

Watson can be tested with py.test. Simply activate your virtualenv and run `python setup.py test`.

CHAPTER 5

Contributing

If you would like to contribute to Watson, please feel free to issue a pull request via Github with the associated tests for your code. Your name will be added to the AUTHORS file under contributors.

CHAPTER 6

Table of Contents

6.1 Usage

6.1.1 Configuration

Before being able to integrate SqlAlchemy with Watson, there are a few things that must be implemented first within your applications config.

1. Add the init event to your applications configuration.

```
'events': {
     events.INIT: [
          ('watson.db.listeners.Init', 1, True)
     ],
 }
```

2. Create a default configuration for a database session.

```
db = {
    'default': {
        'connection_string': 'sqlite:///:memory:',
        'engine_options': {},
        'session_options': {}
     }
 }
```

`engine_options` and `session_options` are optional values and can contain any kwarg values that `create_session` and `sessionmaker` take.

6.1.2 Example

Once configured, the session can be retrieved from the container via `container.get('sqlalchemy_session_[session name]')`.

watson.db also provides a paginator class for paginating a set of results back from SQLAlchemy. Basic usage includes:

```
# within controller
from watson.db import utils
query = session.query(Model)
paginator = utils.Pagination(query, limit=50)
```

```
# within view
{% for item in paginator %}
{% endfor %}
<div class="pagination">
{% for page in paginator.iter_pages() %}
    {% if page == paginator.page %}
    <a href="#" class="current">{{ page }}</a>
    {% else %}
    <a href="#">{{ page }}</a>
    {% endif %}
{% endfor %}
</div>
```

6.2 Reference Library

6.2.1 watson.db.contextmanagers

watson.db.contextmanagers.**transaction_scope**(*session*)

Provides a transactional scope for session calls.

See:

- http://docs.sqlalchemy.org/en/latest/orm/session.html

Example:

```
class MyController(controllers.Rest):

    def GET(self):
        with session(self.db):
            session.add(Model())
```

6.2.2 watson.db.engine

watson.db.engine.**make_engine**(***kwargs*)

Create a new engine for SqlAlchemy.

Remove the container argument that is sent through from the DI container.

6.2.3 watson.db.listeners

class watson.db.listeners.**Complete**

Cleanups the db session at the end of each request.

class watson.db.listeners.**Init**

Bootstraps watson.db into the event system of watson.

Each session and engine can be retrieved from the container by using sqlalchemy_engine_[name of engine] and sqlalchemy_session_[name of session] respectively.

6.2.4 watson.db.panels

6.2.5 watson.db.session

6.2.6 watson.db.utils

class watson.db.utils.**Pagination**(*query*, *page=1*, *limit=20*)

Provides simple pagination for query results.

query

Query

The SQLAlchemy query to be paginated

page

int

The page to be displayed

limit

int

The maximum number of results to be displayed on a page

total

int

The total number of results

items

list

The items returned from the query

Example:

```
# within controller
query = session.query(Model)
paginator = Pagination(query, limit=50)

# within view
{% for item in paginator %}
{% endfor %}
<div class="pagination">
{% for page in paginator.iter_pages() %}
    {% if page == paginator.page %}
    <a href="#" class="current">{{ page }}</a>
    {% else %}
    <a href="#">{{ page }}</a>
    {% endif %}
{% endfor %}
</div>
```

__init__(*query*, *page=1*, *limit=20*)

Initialize the paginator and set some default values.

has_next

Return whether or not there are more pages from the currently displayed page.

Returns boolean

has_previous
Return whether or not there are previous pages from the currently displayed page.

Returns boolean

iter_pages()
An iterable containing the number of pages to be displayed.

Example:

```
{% for page in paginator.iter_pages() %}{% endfor %}
```

pages
The total amount of pages to be displayed based on the number of results and the limit being displayed.

Returns int

W