Using Storm to work with SQL databases

EuroPython 2011
Jamu Kakar
@jkakar

INTRODUCTION

Jamu

History

GOALS

Reliability

Simplicity

Focus

Portability

Flexibility

SESSIONS

Connections

```
from storm.locals import Store, create_database

database = create_database('sqlite:')
store = Store(database)
```

Statements

```
store.execute("""
   CREATE TABLE person (
      id INTEGER PRIMARY KEY,
      name TEXT NOT NULL UNIQUE)""")
```

Flushing

```
store.execute('''
    INSERT INTO person (id, name)
        VALUES (1, 'john')''')
store.flush()

result = store.execute('''
    SELECT id, name FROM person''')
assert list(result) == [(1, u'john')]
```

Transactions

```
store.commit()
store.rollback()
```

Disconnections

```
import time
from storm.exceptions import (
    DisconnectionError)

try:
    return function()
except DisconnectionError:
    time.sleep(1)
    store.rollback()
    return function()
```

Twisted

```
from storm.twisted.transact import Transactor
from twisted.python.threadpool import (
    ThreadPool)

threads = ThreadPool(maxthreads=4)
transaction = Transactor(threads)
deferred = transaction.run(function)
```

ORM

Python

```
class Person(object):

    def __init__(self, name):
        self.name = name
```

Storm

```
from storm.locals import (
    AutoReload, Int, Unicode)

class Person(object):
    __storm_table__ = 'person'
    id = Int(primary=True, default=AutoReload)
    name = Unicode(allow_none=False)

    def __init__(self, name):
        self.name = name
```

```
CREATE TABLE person (
   id INTEGER PRIMARY KEY,
   name TEXT NOT NULL UNIQUE)
```

```
from storm.locals import Store, create_database

database = create_database('sqlite:///test.db')
store = Store(database)
john = Person(u'John')
store.add(john)
store.commit()
```

Retrieval

```
result = store.find(Person)
john = result.one()
assert (john.id, john.name) == (1, u'John')
```

References

```
class Talk(object):
    __storm_table__ = 'talk'
    id = Int(primary=True, default=AutoReload)
    title = Unicode(allow_none=False)
    speaker_id = Int(allow_none=False)
    speaker = Reference(speaker_id, Person.id)

    def __init__(self, person, title):
        self.speaker_id = person.id
        self.title = title
```

```
CREATE TABLE talk (
    id INTEGER PRIMARY KEY,
    title TEXT NOT NULL UNIQUE,
    speaker_id INTEGER REFERENCES person)
```

```
john = store.add(Person(u'John'))
python = store.add(Talk(john, u'Python rocks!'))
assert python.speaker is john
```

```
john = store.add(Person(u'John'))
python = store.add(Talk(john, u'Python rocks!'))
result = store.find(
    Person, Person.id == python.speaker_id)
assert john is result.one()
```

Reference sets

```
class Person(object):
    __storm_table__ = 'person'
    id = Int(primary=True, default=AutoReload)
    name = Unicode(allow_none=False)
    talks = ReferenceSet(id, Talk.speaker_id)

    def __init__(self, name):
        self.name = name
```

```
john = store.add(Person(john))
italy = store.add(Talk(john, u'Italy rocks!'))
python = store.add(Talk(john, u'Python rocks!'))

result = john.talks.order_by(Talk.title)
assert [italy, python] == list(result)
```

```
john = store.add(Person(u'John'))
italy = store.add(Talk(john, u'Italy rocks!'))
python = store.add(Talk(john, u'Python rocks!'))

result = store.find(
    Person,
    Person.id == Talk.speaker_id,
    Talk.speaker_id == john.id)
result = result.order_by(Talk.title)
assert [italy, python] == list(result)
```

Dependencies

```
class Person(Storm):
    __storm_table__ = 'person'
    id = Int(primary=True)
    name = Unicode(allow_none=False)
    talks = ReferenceSet(id, 'Talk.speaker_id')

    def __init__(self, name):
        self.name = name
```

Find

```
SELECT * FROM person, talk
  WHERE person.id = talk.speaker_id
  ORDER BY person.name
```

```
result = store.find(
    Person, Person.id == Talk.speaker_id)
for person in result.order_by(Person.name):
    print person.name
```

Find faster

```
result = store.find(
    Person.name, Person.id == Talk.speaker_id)
store.config(distinct=True)
for name in result:
    print name
```

```
result = store.find(
    Person, Person.id == Talk.speaker_id)
store.config(distinct=True)
for name in result.values(Person.name):
    print name
```

Subselects

```
result = store.find(
    Talk.id, Talk.speaker_id != None)
subselect = result.get_subselect_expr(Talk.id)
result = store.find(
    Person.name, Person.id.is_in(subselect))
for name in result:
    print name
```

```
SELECT person.name FROM person
  WHERE person.id IN (
    SELECT talk.speaker_id FROM talk
      WHERE talk.speaker_id IS NOT NULL)
```

Subfinds

```
result = store.find(Person)
result = result.find(Person.name.like('J%'))
```

Functions

```
from storm.expr import Lower

result = store.find(Lower(Person.name))
for name in result:
    print name
```

```
SELECT LOWER(person.name) FROM person
```

```
from storm.expr import NamedFunc

class Lower(NamedFunc):
    __slots__ = ()
    name = 'LOWER'
```

Aggregating

```
john = store.add(Person(u'John'))
store.add(Talk(john, u'Italy rocks!'))
store.add(Talk(john, u'Python rocks!'))

jane = store.add(Person(u'Jane'))
Store.add(Talk(jane, u'Florence rocks!'))
```

```
SELECT COUNT(talk.id) AS count, person.name
  FROM person, talk
  WHERE person.id = talk.speaker_id
  GROUP BY count
  ORDER BY count DESC
```

```
from storm.expr import Alias
from storm.locals import Count, Desc

count = Alias(Count(Talk.id))
result = store.find(
    (count, Person.name),
    Person.id == Talk.speaker_id)
result = result.group_by(count)
result = result.order_by(Desc(count))
assert [(2, u'John'), (1, u'Jane')] == list(result)
```

Validation

```
def no_spaces(obj, attribute, value):
    if ' ' in value:
        raise ValueError('No spaces allowed!')
    return value

class Person(object):
    __storm_table__ = 'person'
    id = Int(primary=True, default=AutoReload)
    name = Unicode(
        allow_none=False, validator=no_spaces)
```

Hooks

```
class Person(Storm):
    def __storm_invalidate__(self): pass
    def __storm_loaded__(self): pass
    def __storm_flushed__(self): pass
```

Tracing

```python
from storm.tracer import debug

debug(True)
```

MIDDLEWARE

Django

```
DATABASE_ENGINE = 'postgresql_psycopg2'
DATABASE_NAME = 'europython'
DATABASE_USER = 'europython'
DATABASE_PASSWORD = 'europython'
DATABASE_HOST = 'localhost'
DATABASE_PORT = ''
STORM_STORES = {
    'main': 'postgres:///europython'
}
```

```
from storm.django.stores import get_store

store = get_store('main')
```

Zope 3

```
<include package='storm.zope' />
<store name='main' uri='sqlite:///europython' />
```

```python
from storm.zope.interfaces import IZStorm
from zope.component import getUtility

zstorm = getUtility(IZStorm)
store = zstorm.get('main')
```

PATTERNS

Functions

```
def get_person_by_name(name)
def get_people_with_talks()
def get_people_without_talks()
def get_talks()
def get_talks_matching(text)
```

Collections

```
class PersonCollection(object):
    def with_name(self, name): pass
    def with_talk(self): pass
    def without_talk(self): pass
    def find(self): pass
```

Thanks

#storm on chat.freenode.net

https://storm.canonical.com
https://launchpad.net/storm

@jkakar
jkakar@kakar.ca