LinuxFest Northwest

2016

An ORM Where the Database is the Boss

Featuring Simpycity

Eric Worden
eric@commandpromt.com

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

ORM in a Nutshell

```
create table animal
(
    id integer,
    genus text,
    species text
    extinct boolean
)

class Animal:
    property id
    property genus
    property species
    property extinct
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Some Example ORMs

- Rails**
- Django**
- SQLAlchemy
- peewee
- PetaPoco
- Simpycity++

**Rails and Django are application frameworks, with ORM features baked in

++Simpycity is awesome

Typical ORM Philosophy

- My program requires interaction with a database
- The database requires interface via SQL
- BUT, I don't like SQL


SQL

Typical ORM Philosophy, cont'd

- My program requires interaction with a database
- The database requires interface via SQL
- I don't like SQL
- I DO like … *DookieScript*!!! (????)
- I will use a DookieScript QL and forget all about SQL!

Dookie QL

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

SUCKA!

This DookieVision is even better than the real thing!!!

Dookie HTML

```
<p><FONT size="16pt" color="blue"
face="Times New Roman">

    MICROSOFT
FRONTPAGE    

</FONT></p>
```

Better HTML

```
<p class="banner">YO</p>
```


Dookie HTML

```
<a onclick="javascript:var
stuff_data = '99999'; if
(stuff_data > other_stuff)
{do_less_stuff();} else
{do_more_stuff();} return stuff;"
    Click Me
</a>
```

Better HTML

```
<a onclick="javascript:do_stuff()"
    Click Me
</a>
```


Dookie Python

```
sql = """WITH preferences(conversion_source,conversion_factor) AS
(
    VALUES
    (
        "user".get_preference(CAST(current_setting('acme.user_id') AS
INTEGER),in_prefix||'_conversion_source'),
        "user".get_preference(CAST(current_setting('acme.user_id') AS
INTEGER),in_prefix||'_conversion_factor')
    )
)
SELECT CASE WHEN conversion_source = 'ABC Value' THEN NULL
            WHEN conversion_factor !~ '^([0-9]{1,2}|[0-9]{1,2}[.][0-9]{0,4}|[.][0-9]{1,4})$' THEN 'Custom
Value'
            WHEN CAST(conversion_factor AS NUMERIC(6,4)) = 0 THEN 'Custom Value'
            ELSE conversion_factor
       END
FROM preferences;
"""
cursor.execute(sql, params)
```

Better Python

```
sql = “select * from get_data(%s)”
cursor.execute(sql, params)
```


Simpycity Philosophy

- The database requires interface via SQL
- My feelings about SQL are irrelevant
- But I like DookieScript !!!
- I will integrate DookieScript and SQL intelligently


SQL is TASTY!!!
SQL

“Models”

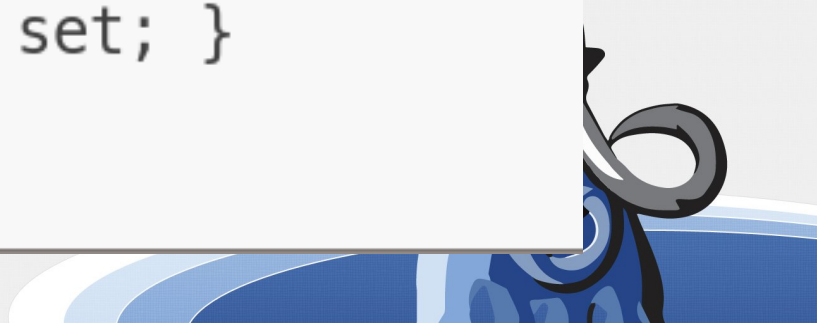
Non-SQL that translates to SQL ~table

SQLAlchemy

```
class User(Base):
    __tablename__ = 'user'

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    password = Column(String)
```

PetaPoco

```
public class article

{

    public long article_id { get; set; }

    public string title { get; set; }

    public DateTime date_created { get; set; }

    public bool draft { get; set; }

    public string content { get; set; }

}
```



Rails

```
class Product < ActiveRecord::Base
end
```

```
p = Product.new
p.name = "Some Book"
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Rails Deep Thoughts

“If you're used to using raw SQL to find database records, then you will generally find that there are better ways to carry out the same operations in Rails. Active Record ***insulates you from the need to use SQL*** in most cases.”

http://guides.rubyonrails.org/active_record_querying.html

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

SQLAlchemy Deep Thoughts

“SQL databases behave less like object collections the more size and performance start to matter; object collections behave less like tables and rows the more abstraction starts to matter.”

http://www.sqlalchemy.org/

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Dookie QL (Query Language)

- Examples...

SQLAlchemy DookieQL

http://docs.sqlalchemy.org/en/rel_1_0/orm/query.html

```
>>> jack = session.query(User).\
...                 options(subqueryload(User.addresses)).\
...                 filter_by(name='jack').one()
```

SQL

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

PetaPoco DookieQL

```
var sql=PetaPoco.Sql.Builder()

            .Select("*")

            .From("articles")

            .Where("date_created < @0", DateTime.UtcNow)

            .OrderBy("date_created DESC");
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Rails DookieQL

Client.joins(:orders).where(orders: { created_at: time_range })

http://api.rubyonrails.org/

Dookie QL Falls Over

```
begin
    v_null_result := (0,null::foo.found_bar[])::foo.search_result;
    create temporary table matching_bars
    (
        bar TEXT,
        type bar.type,
        found_via search.type[]
    ) on commit drop;
    insert into  matching_bars
        SELECT bar, type, found_via

              FROM bar.search(in_query,
                              in_extra_search_types,
                              ARRAY[in_bar_type::TEXT],
                              NULL, NULL)  -- limit after possible filtering on level

             WHERE (in_level IS NULL OR
                    in_level = foo.level(bar, type))
             ORDER BY bar
    ;
    GET DIAGNOSTICS count_extra = ROW_COUNT;
    create index ix_matching_bars on matching_bars (bar, type, found_via);

    create temporary table minor
    (like matching_bars) on commit drop;
    insert into minor (bar, type, found_via)
        SELECT bar, type, found_via
          FROM matching_bars
         WHERE found_via && '{bars,tools}'::search.type[]
         order by bar
    ;
    GET DIAGNOSTICS count_minor = ROW_COUNT;
    create index ix_minor_bar on minor (bar);

    v_extra_result := foo.make_search_result('matching_bars', count_extra, in_offset, in_limit);
    v_minor_result := foo.make_search_result('minor', count_minor, in_offset, in_limit);
    if in_bar_type::text like 'foo9%' then
        v_foo9_extra_result := v_extra_result;
        v_foo9_minor_result := v_minor_result;
        v_foo10_extra_result := v_null_result;
        v_foo10_minor_result := v_null_result;
    elsif in_bar_type::text like 'foo10%' then
        v_foo10_extra_result := v_extra_result;
        v_foo10_minor_result := v_minor_result;
        v_foo9_extra_result := v_null_result;
        v_foo9_minor_result := v_null_result;
```

Complex SQL

Simpycity Tutorial
Step 1: Define your model in the *Database*

```
create table animal
(
    id int primary key,
    genus_id int,
    species text,
    common_name text,
    extinct boolean
);
--"animal" is a table = a type
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Create matching Python model

```
import simpycity.model as sm
import simpycity.core as sc
class Animal(sm.SimpleModel):
    table = ['id', 'genus_id',
             'species', 'common_name',
             'extinct']
    __load__ = \
sc.QuerySingle('public.animal', ['id'])
```

Configure and instantiate

```
import simpycity.config
import animals

simpycity.config.port = 5434
simpycity.config.database = 'animals'
simpycity.config.user = 'ormboss'

animal = animals.Animal(id=3)
print(animal.common_name)
>>>racoon
```

Query the model

```
CREATE FUNCTION mammals() RETURNS
setof animal as
$body$
    SELECT * FROM animal
    WHERE id in (1,3,7);
$body$ language sql;
```

Py function <=> Pg function

```
class Animal(sm.SimpleModel):
    ….
    mammals = sc.Function('public.mammals', [])

for mammal in animal.mammals():
    print(mammal)
```

```
>>>[1, 2, 'sapiens', 'human', False]
>>>[3, 3, 'lotor', 'racoon', False]
>>>[7, 7, 'hemionus', 'mule deer', False]
```

Lists suck. Return model objects instead

```
animals.Animal.register_composite(\
'public.animal')


mammals = \
sc.FunctionTyped('public.mammals', [])


print(mammal['common_name'])
print(mammal.common_name)
```

Model properties from pg functions

```
CREATE FUNCTION human() RETURNS animal AS
$$ SELECT * FROM animal where id = 1; $$
language sql;

class Animal(simpycity.model.SimpleModel):
    ...
    master = sc.Property('public.human', [])

animal = animals.Animal(id=3)
print(animal.master.species)
>>>sapiens
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Automatic table attribute

```
class Animal(sm.SimpleModel):
    table = ['id',
             'genus_id',
             'species',
             'common_name',
             'extinct']
    pg_type = ('public', 'animal')
#Model attributes are queried/defined at
runtime
```

Postgres table inheritance

```
create table predator
(
    tactic text,
    primary key (id)
) inherits (animal);
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Pg Type/Py Class Inheritance

```
create type predator_type as
(
    base_ animal,
    tactic text,
    prey animal[]
);
--Yes, Postgres can do that!
```

SQL Power!!!

```
create or replace function predator(id int)
returns predator_type language sql as $$
select
    row(p1.id, p1.genus_id, p1.species,
p1.common_name, p1.extinct)::animal,
    p1.tactic,
    array_agg(row(prey.*)::animal)::animal[]
from
…
$$;
```

Command Prompt, Inc.
http://www.commandprompt.com/
@cmdpromptinc

Python Model Inheritance

```
class Predator(Animal):
    pg_type = ('public', 'predator_type')
    __load__ = \
sc.FunctionTypedSingle('public.predator', ['id'])
```

Composite types unpacked recursively!

```
animals.Predator.register_composite(\
“public.predator_type”)
trex = animals.Predator(id=10)
print(trex.common_name + “, “ + trex.tactic)
for victim in trex.prey:
    print(“Victim: “ + victim.common_name)
```

```
>>>tyranosaurus rex, chomp!
>>>Victim: mule deer
>>>Victim: elephant
```

Get Simpycity

http://github.com/commandprompt/Simpycity

- Type mapping features are brand new, *only* on development branch *type_registration.*
- New stable release soon
- No Python 3 yet
- Requires psycopg2 2.5+
- Development help welcome!

Evaluate this session

An ORM Where the Database is the Boss

Session videos will be posted to the LinuxFest Northwest YouTube channel.

Thank you!