ascetic Documentation

Release stable

September 05, 2015

Contents

Autumn exists as a super-lightweight Object-relational mapper (ORM) for Python. Autumn ORM follows the KISS principle. Has automatic population of fields from database (see the example below), and minimal size. You do not have to specify the columns in the class. This follows the DRY principle. Autumn as small as possible.

It is released under the MIT License (see LICENSE file for details).

This project is currently considered beta software.

CHAPTER 1

PostgreSQL Example

Using these tables:

```
CREATE TABLE autumn_tests_models_author (
    id serial NOT NULL PRIMARY KEY,
    first_name VARCHAR(40) NOT NULL,
    last_name VARCHAR(40) NOT NULL,
    bio TEXT
);
CREATE TABLE books (
    id serial NOT NULL PRIMARY KEY,
    title VARCHAR(255),
    author_id integer REFERENCES autumn_tests_models_author(id) ON DELETE CASCADE
);
```

Put in your PYTHONPATH file autumn_settings.py with your settings. See file autumn/settings.py for more details.

You can also set the settings directly:

```
import autumn.settings
autumn.settings.DATABASES = {
    'default': {
        'engine': "postgresql",
        'user': "devel",
        'database': "devel_autumn",
        'password': "devel",
        'debug': True,
        'initial_sql': "SET NAMES 'UTF8';",
        'thread_safe': True,
    }
}
```

We setup our objects like so:

```
from autumn.model import Model, ForeignKey, OneToMany

class Author(Model):

    class Meta:
        defaults = {'bio': 'No bio available'}
        validations = {'first_name': (
            lambda v: len(v) > 1 or "Too short first name",
            lambda self, key, value: value != self.last_name or "Please, enter another first name",
        )}
```

```
class Book(Model):
    author = ForeignKey(Author, related_name='books')

    class Meta:
        db_table = 'books'
```

Now we can create, retrieve, update and delete entries in our database. Creation

```
james = Author(first_name='James', last_name='Joyce')
james.save()

u = Book(title='Ulysses', author_id=james.id)
u.save()
```

CHAPTER 2

Retrieval

```
a = Author.get(1)
a.first_name # James
a.books      # Returns list of author's books

# Returns a list, using LIMIT based on slice
a = Author.get()[:10]   # LIMIT 0, 10
a = Author.get()[20:30] # LIMIT 20, 10
```

CHAPTER 3

Updating

```
a = Author.get(1)
a.bio = 'What a crazy guy! Hard to read but... wow!'
a.save()
```

CHAPTER 4

Deleting

```
a.delete()
```

CHAPTER 5

SQLBuilder integration

```
object_list = Book.qs.tables(
    (Book.s & Author.s).on(Book.s.author_id == Author.s.id)
).where(
    (Author.s.first_name != 'James') & (Author.s.last_name != 'Joyce')
)[:10]
```

QuerySet based on sqlbuilder.smartsql, see more info.

CHAPTER 6

Signals support

- pre_init
- post_init
- pre_save
- post_save
- pre_delete
- post_delete
- class_prepared

CHAPTER 7

Web

You can use Autumn ORM with lightweight web-frameworks, like wheezy.web, Bottle, Tornado, pysi, etc.

CHAPTER 8

Gratitude

Forked from https://github.com/lucky/autumn
Thanks to Jared Kuolt (lucky)

CHAPTER 9

Other projects

See also:

- SQLAlchemy (scheme from class or database, see “autoload” option)
- SQLObject (scheme from class or database, see “fromDatabase” option)
- Storm (properties from class)
- Peewee (scheme from class)
- Twistar (scheme from database), provides asynchronous DB interaction
- Openorm (lightweight datamapper)
- Activemodel (scheme from database)
- ActiveRecord like ORM under 200 lines (scheme from database)