nashvegas Documentation

Release 0.7

Patrick Altman

October 15, 2015

Contents

The purpose of this app is to enable a plug and play method for managing database changes.

Database migrations is a large topic with a lot of different approaches. This approach worked well for my needs and maybe it will for you as well.

CHAPTER 1

Installation

- pip install nashvegas
- Add the application to your INSTALLED_APPS list in your settings.py file.

CHAPTER 2

Settings

You can set the `NASHVEGAS_MIGRATIONS_DIRECTORY` to whatever absolute path you are using to store your migrations. It defaults to `migrations/` at the same level as your `settings.py`.

CHAPTER 3

Usage

nashvegas ships with two management commands, `upgradedb` and `comparedb`. The first, `upgradedb`, will manage the creation, listing, and execution of individual migrations. The second, `comparedb`, is currently an experimental command that attempts to help you discover missing migrations.

- Execute the command line:

```
    $ ./manage.py upgradedb --create
    $ ./manage.py upgradedb --list
    $ ./manage.py upgradedb --execute

    $ ./manage.py comparedb
```

It is useful to create an initial migration when starting any new project (or adding nashvegas to the project to start tracking schema changes). This can easily be done with:

```
$ ./manage.py upgradedb -c > migrations/001_initial.sql
```

Having this initial migration also helps when running tests.

CHAPTER 4

Options for upgradedb

- `--create` - Compares database with current models in apps that are installed and outputs the sql for them so that you can easily pipe the contents to a migration.
- `--list` - Lists all the scripts that will need to be executed.
- `--execute` - Executes all the scripts that need to be executed.
- `--seed` - Populates Migration model with scripts that have already been applied to your database and effectively want to skip execution. Provide a migration id to stop at. For instance, running *./manage.py upgradedb –seed 005* will skip migrations 000 to 005 but not 006.

CHAPTER 5

Conventions

Part of the simplicity of this solution is based on the naming conventions of the sql scripts. They should be named in a manner that enforces order. Some examples include:

```
0001_short_comment_about_migration.sql
0001.sql
```

The model, `nashvegas.Migration` will get synced into your database if it doesn't exist when you go to execute any of the `upgradedb` commands. In this model the scripts that have been executed will be recorded, effectively versioning your database.

In addition to sql scripts, `--execute` will also execute python scripts that are in the directory. This are run in filename order interleaved with the sql scripts. For example:

```
0001.sql
0002.py
0003.sql
```

The Python script will be executed 2nd between `0000.sql` and `0003.sql`. The script will only be executed if the module contains a callable named `migrate`. It is a good idea to put all your executing code within a class or series of functions or within a single `migrate()` function so as to avoid code executing upon import.

For example, your script might look like this if you need to update all your product codes on next release:

```python
from store.models import Product

def migrate():
    for product in Product.objects.all():
        product.code = "NEW-%s" % product.code
        product.save()
```

CHAPTER 6

Configuration for comparedb

The *comparedb* command is available only for advanced system administrators. It proceeds as such:

- get the SQL structure dump of the current database
- create a new database, the "compare" database
- syncdb in the "compare" database,
- get the SQL structure dump of the "compare" database
- output the diff

It executes a few raw shell commands which you might need to customize to add user credentials, encoding or specify database templates. This can be done through the *NASHVEGAS* dictionnary in your setting.

6.1 Example for PostgreSQL

By default, nashvegas executes raw *createdb*, *dropdb* or *pg_dump* commands, example customisation:

```
NASHVEGAS = {
    "createdb": "createdb -U postgres -T template0 -E UTF8 {dbname}",
    "dropdb": "dropdb -U postgres {dbname}",
    "dumpdb": "pg_dump -U postgres -s {dbname}",
}
```

If you add a field "test" on model "Foo", comparedb will output:

```
>>> ./manage.py comparedb
Getting schema for current database...
Getting schema for fresh database...
Outputing diff between the two...
---
+++
@@ -515,7 +515,8 @@

 CREATE TABLE testapp_foo (
     id integer NOT NULL,
-    bar character varying(100)
+    bar character varying(100),
+    test character varying(100)
 );
```

6.2 Example for MySQL

MySQL is not supported by default though such settings do work:

```
NASHVEGAS = {
    "createdb": "mysql -u root -p -e \"create database {dbname}\"",
    "dropdb": "mysql -u root -p -e \"drop database {dbname}\"",
    "dumpdb": "mysqldump -u root -p {dbname}",
}
```

If you add a field “test” on model “Foo”, comparedb will output:

```
>>> ./manage.py comparedb
Getting schema for current database...
Enter password:
Getting schema for fresh database...
Enter password:
Enter password:
Enter password:
Outputing diff between the two...
---
+++
@@ -1,6 +1,6 @@
 -- MySQL dump 10.13  Distrib 5.1.58, for debian-linux-gnu (x86_64)
 --
--- Host: localhost    Database: testproject
+-- Host: localhost    Database: testproject_compare
 -- ------------------------------------------------------
 -- Server version  5.1.58-1ubuntu1

@@ -419,6 +419,7 @@
 CREATE TABLE `testapp_foo` (
   `id` int(11) NOT NULL AUTO_INCREMENT,
   `bar` varchar(100) DEFAULT NULL,
+  `test` varchar(100) DEFAULT NULL,
   PRIMARY KEY (`id`)
 ) ENGINE=MyISAM DEFAULT CHARSET=latin1;
 /*!40101 SET character_set_client = @saved_cs_client */;
@@ -441,4 +442,4 @@
 /*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
 /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

--- Dump completed on 2012-03-07 12:58:15
+-- Dump completed on 2012-03-07 12:58:18
```

Typicall customisation would be to setup a *$HOME/.my.cnf* that contains credentials allowing to run this command without password prompt.

6.2.1 Indices and tables

- genindex
- modindex
- search