DjHDGutils.PatchManager

patch command

Introduction

PatchManager makes it easy to support large scale project migrations in the well controlled manner.
Patches can be different types, namely:

- .sql
- .py

and they are run in sequentual manner one after one. The migrations are made that way that several developers in a different branches can run their patches, merge them without conflicts (a random number at the end of the patch name makes it low probability to patch names to interfere) and test well before going to production.

Patch names

Patches can be named automatically that is for **next** command is made:

```
(env)djyp $ ./manage.py patch next
File /someproject/migrations/00636-avk-654.sql has been created
size: 0
```

Filename consists from:

- 00636 - patch number
- avk - developer name
- 654 - the random part to ensure that patch names do not interfer with same developer different branch

Commands

- **list** - list pending patches
- **install** - install PatchManager track model table
- **up** - update project to latest migration
- **skip** - skip specific patch
- **next** - auto-generate the next available patch name

Compare to other systems

Most of other systems differ by their primary usage. South for example mainly used for distributed project support. Our system is simple and straightforward to use, we always know what patch is going to be installed on a production database.
And so our advantages:

- Very simple, robust, reliable application
- No conflicts during merge

Typical migrate session

In a typical usage session you do 2 things:

- expect what patches are pending
- apply them

Or even just directly apply them, as we usually do in our upgrade scripts.

```
./manage.py patch list
./manage.py patch up
```