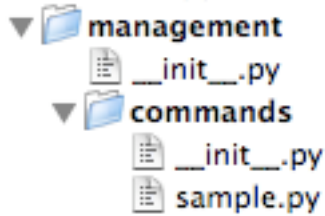


Custom Management Commands

BaseCommand
`def handle(self, *args, **options)`



NoArgsCommand
`def handle_noargs(self, **options)`

LabelCommand
`def handle_label(self, label, **options)`

AppCommand
`def handle_app(self, app, **options)`

Attributes

<code>requires_model_validation</code>	true if models are validated before the command is executed (True/False)
<code>output_transaction</code>	wraps sql commands in BEGIN; and COMMIT statements to execute in a transaction (True/False)
<code>can_import_settings</code>	assume we have a working settings file and that it can be imported (True/False)
<code>option_list</code>	a list of option arguments (optparse) for extending the behavior of a custom management command
<code>label</code>	used in error message for a LabelCommand when no label argument is provided
<code>help</code>	descriptive text to appears in the command help
<code>args</code>	the argument text to display in the command help

```

Usage: manage.py startapp [options] [appname]

Creates a Django app directory structure for the given app name in this project's directory.

Options:
  --settings=SETTINGS  The Python path to a settings module, e.g.
                        "myproject.settings.main". If this isn't provided, the
                        DJANGO_SETTINGS_MODULE environment variable will be
                        used.
  --pythonpath=PYTHONPATH
                        A directory to add to the Python path, e.g.
                        "/home/djangoprojects/myproject".
  --version            show program's version number and exit
  -h, --help          show this help message and exit
  
```