alembic-offline Documentation

Release stable

August 28, 2015

Contents

alembic-offline is an extension for alemic to enrich offline functionality of the migrations

Contents

CHAPTER 1

Usage

1.1 Phased migrations

alembic-offline introduces a helper which allows to implement phased migrations, e.g. those which steps are divided into logical phases. For example, you can have steps to be executed before code deploy and those after.

In your alembic config file (main section):

```
phases = before-deploy after-deploy final
default-phase = after-deploy
```

In your version file:

```
from sqlalchemy import INTEGER, VARCHAR, NVARCHAR, TIMESTAMP, Column, func
from alembic import op

from alembic_offline import phased, execute_script

from tests.migrations.scripts import script

revision = '1'
down_revision = None


@phased
def upgrade():

    op.create_table(
        'account',
        Column('id', INTEGER, primary_key=True),
        Column('name', VARCHAR(50), nullable=False),
        Column('description', NVARCHAR(200)),
        Column('timestamp', TIMESTAMP, server_default=func.now())
    )
    yield
    op.execute("update account set name='some'")
    yield
    execute_script(script.__file__)


def downgrade():
    pass
```

Will give the sql output (for sqlite):

```
-- Running upgrade  -> 1

-- PHASE::before-deploy::;

CREATE TABLE account (
    id INTEGER NOT NULL,
    name VARCHAR(50) NOT NULL,
    description NVARCHAR(200),
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (id)
);

-- PHASE::after-deploy::;

update account set name='some';

-- PHASE::final::;

-- SCRIPT::scripts/script.py::;

INSERT INTO alembic_version (version_num) VALUES ('1');
```

As you see, phases are rendered as SQL comments to divide migration steps, so those who execute migration can see which phase's step it is. However, if migration procedure is highly customized, you can use alembic-offline API described below. *get_migration_data* returns migration phases in special form so you can automate their execution.

1.2 Arbitrary script as operation

For complex migrations, it's not enough to execute sql, you might need some script to be executed instead. For that, there's special operation:

```
from alembic_offline import execute_script

def upgrade():
    execute_script('scripts/script.py')
```

If you'll get migration sql, it will be rendered as SQL comment:

```
-- SCRIPT::scripts/script.py::;
```

For those who execute migrations it will be visible and they can execute the script manually. However, if migration procedure is highly customized, you can use alembic-offline API described below. *get_migration_data* returns script migration steps in special form so you can automate their execution. For online mode, the script will be executed as subprocess via python *subprocess* module.

1.3 Get migration data

alembic-offline provides specialized API to get certain migration data as dictionary:

```
from alembic_offline import get_migration_data

from alemic.config import Config

config = Config('path to alemic.ini')
```

```
data = get_migration_data(config, 'your-revision')

assert data == {
    'revision': 'your-revision',
    'phases': {
        'after-deploy': [
            {
                'type': 'mysql',
                'script': 'alter table account add column name VARCHAR(255)'
            },
            {
                'type': 'python',
                'script': 'from app.models import Session, Account; Session.add(Account()); Session.c
                'path': 'scripts/my_script.py'
            },
        ]
    }
}
```

get_migration_data requires both *phases* and *default-phase* configuration options to be set. *default-phase* is needed to be able to get migration data even for simple migrations without phases.

1.4 Get migration data in batch

alembic-offline provides an API call to get migration data for all revisions:

```
from alembic_offline import get_migrations_data

from alemic.config import Config

config = Config('path to alemic.ini')

data = get_migrations_data(config)

assert data == [
    {
        'revision': 'your-revision',
        'phases': {
            'after-deploy': [
                {
                    'type': 'mysql',
                    'script': 'alter table account add column name VARCHAR(255)'
                },
                {
                    'type': 'python',
                    'script': 'from app.models import Session, Account; Session.add(Account()); Sessi
                    'path': 'scripts/my_script.py'
                },
            ]
        }
    }
]
```

1.5 Command line utilities

Because with alembic revisions it's sometimes hard to find which the correct down revision should be; especially when there are multiple heads we added the alembic-offline graph command.

The graph command will generate a dot file of the revisions, this file can then be converted to an image for easy visualization.

Usage:

```
alembic-offline graph --filename revisions.dot --alembic-config path/to/alembic.ini
```

Then if you have graphviz installed you can run:

```
dot -Tpng -o revisions.png revisions.dot
```

To generate a png image.

CHAPTER 2

Contact

If you have questions, bug reports, suggestions, etc. please create an issue on the GitHub project page.

CHAPTER 3

License

This software is licensed under the MIT license

Please refer to the license file