Alembic Verify Documentation

Release 0.0.1

Student.com

Feb 11, 2019

Contents

Verify that your alembic migrations are valid and equivalent to your models.

CHAPTER 1

PyTest Example

1.1 Validating Migrations

Copy the following test suite into your project to verify that your migrations are valid. You will have to adapt the `alembic_root` and db related fixture to your structure. All other fixtures are included as part of alembic-verify.

```
# -*- coding: utf-8 -*-
import os

import pytest
from alembic import command
from sqlalchemydiff import compare
from sqlalchemydiff.util import (
    prepare_schema_from_models,
    get_temporary_uri,
)

from alembicverify.util import (
    get_current_revision,
    get_head_revision,
    prepare_schema_from_migrations,
)
from .models import Base


@pytest.fixture
def alembic_root():
    return os.path.join(
        os.path.dirname(__file__), 'migrations', 'alembic'
    )


@pytest.fixture(scope="module")
def uri_left(db_uri):
```

(continues on next page)

(continued from previous page)

```
    return get_temporary_uri(db_uri)


@pytest.fixture(scope="module")
def uri_right(db_uri):
    return get_temporary_uri(db_uri)


@pytest.mark.usefixtures("new_db_left")
def test_upgrade_and_downgrade(uri_left, alembic_config_left):
    """Test all migrations up and down.

    Tests that we can apply all migrations from a brand new empty
    database, and also that we can remove them all.
    """
    engine, script = prepare_schema_from_migrations(
        uri_left, alembic_config_left)

    head = get_head_revision(alembic_config_left, engine, script)
    current = get_current_revision(alembic_config_left, engine, script)

    assert head == current

    while current is not None:
        command.downgrade(alembic_config_left, '-1')
        current = get_current_revision(alembic_config_left, engine, script)
```

1.2 Testing Migrations against Models

You can copy this test in your code to make sure your migrations are always up to date with your models. If any change in either of those would leave them out of sync, this test will fail.

```
@pytest.mark.usefixtures("new_db_left")
@pytest.mark.usefixtures("new_db_right")
def test_model_and_migration_schemas_are_the_same(
        uri_left, uri_right, alembic_config_left):
    """Compare two databases.

    Compares the database obtained with all migrations against the
    one we get out of the models.
    """
    prepare_schema_from_migrations(uri_left, alembic_config_left)
    prepare_schema_from_models(uri_right, Base)

    result = compare(uri_left, uri_right, set(['alembic_version']))

    assert result.is_match
```

CHAPTER 2

Using unittest

If you prefer to use a testing approach based on unittest, you can find an example on how to do it here.

CHAPTER 3

Features

With this library you will be able to verify the consistency between two databases, one from Alembic migrations and one from SQLAlchemy models.

The library provides tools for handling migrations and creating all the required Alembic objects with ease. Comprehensive examples are included.

The foundation for the comparison is the SQLAlchemy Diff (todo: link) library.

CHAPTER 4

Installation

```
$ pip install alembic-verify
```