Django Heroku Connect

Thermondo GmbH

Sep 05, 2023

CONTENTS:

Django integration for Salesforce using Heroku Connect.

Model classes inheriting from `HerokuConnectModel` can easily be registered with Heroku Connect, which then keeps their tables in the Heroku database in sync with Salesforce.

CHAPTER

ONE

ALL CONTENTS

1.1 Quick Start

1.1.1 Deploy to Heroku

We have a Deploy to Heroku sample that gets you started in less than a minute.

1.1.2 Setup

Simply install the PyPi package…

```
pip install django-heroku-connect
```

…and add `heroku_connect` to the `INSTALLED_APP` settings.

You can add *`heroku_connect.db.router.HerokuConnectRouter`* to your `DATABASE_ROUTERS` setting, to prevent you from making write access errors.

Last but not least make sure to change the database engine, e.g.:

```
import dj_database_url

DATABASES['default'] = dj_database_url.config(
    engine='heroku_connect.db.backends.postgresql'
)

# or for PostGIS support:

DATABASES['default'] = dj_database_url.config(
    engine='heroku_connect.db.backends.postgis'
)
```

1.1.3 Example

This is what your `models.py` might look like:

```
from django.db import models
from heroku_connect.db import models as hc_models


class User(hc_models.HerokuConnectModel):
    sf_object_name = 'User'

    username = hc_models.Text(
        sf_field_name='Username', max_length=80)
    email = hc_models.Email(sf_field_name='Email')
    department = hc_models.Text(
        sf_field_name='Department', max_length=80)
    title = hc_models.Text(sf_field_name='Title', max_length=80)


class UserComment(models.Model):
    user = models.ForeignKey(User, to_field='sf_id',
                             on_delete=models.SET_NULL, null=True)
    comment = models.TextField()
```

In this example we read-only synchronize the `User` object to your Django application. We add another model, `UserComment` which is managed by Django you can write to it and set a foreign relation to the User object. Note that you should never use the internal primary key for foreign relations. This key may change when Heroku Connect re-synchronizes your table.

1.1.4 Deployment

Note: For convenience you will need the Heroku Connect CLI Plugin.

Make sure to set your Salesforce Organization ID in your Heroku application environment.

```
heroku config:set HEROKU_CONNECT_ORGANIZATION_ID=00Dxxx
# You want to add it to your local environment too,
# to execute some management commands locally.
export HEROKU_CONNECT_ORGANIZATION_ID=00Dxxx
```

Next deploy your code to Heroku, using your preferred method.

As a next step you will need to provision and setup the Heroku Connect add-on if you haven't already. Simply follow the Heroku Connect tutorial.

As a final step, import the correct mappings:

```
python manage.py makemappings -o hc_mappings.json
heroku connect:import hc_mappings.json
```

That's it, enjoy!

1.2 Checks

1.2.1 Model Checks

heroku_connect.E001

The class attribute `HerokuConnectModel.sf_object_name` has not been set.

heroku_connect.E002

`HerokuConnectModel.sf_access` must be either `read_only` or `read_write`.

heroku_connect.E003

The model has fields with duplicate `HerokuConnectFieldMixin.sf_field_name`.

heroku_connect.E004

More than one field is defined as `HerokuConnectFieldMixin.upsert`.

heroku_connect.E005

A related field (`ForeignKey` or `ManyToManyField`) points to the `id` field of a Heroku Connect model.

Heroku Connect uses the `id` column for internal purposes and may change it at any given time. The `id` column should not be referenced, since it does not represent a key for the Salesforce record. It is recommended to use an External ID or the Salesforce ID `sfid`.

Note: Always use an External ID, if you want to write to Heroku Connect, see: https://devcenter.heroku.com/articles/writing-data-to-salesforce-with-heroku-connect#simple-relationships-between-two-objects-and-relationship-external-ids

---

heroku_connect.E006

The Salesforce object name must be unique since an object can only mapped to a PostgreSQL table once.

heroku_connect.E007

Read-write mappings in Heroku Connect need to provide an upsert field to identify the record.

heroku_connect.E008

A related field (ForeignKey or ManyToManyField) pointing to a Heroku Connect model may not use database-constraints.

(previously heroku_connect.W001)

Note: Sometimes Heroku Connect needs to recreate the table, which breaks when if there are database constraints pointing to them.

see: https://devcenter.heroku.com/articles/heroku-connect-logs-errors#error-during-sync-psycopg2-integrityerror-update-or-delete-on-_c-violates-foreign-key-constraint

---

heroku_connect.W001

Deprecated, changed to heroku_connect.E008.

There are situations where Heroku Connect sync breaks if there is a database constraint pointing to a Heroku-Connect table.

1.3 Management Commands

1.3.1 makemappings

class heroku_connect.management.commands.makemappings.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Return Heroku Connect mapping JSON for the entire project.

Example:

```
python manage.py makemappings -o hc_mappings.json
heroku connect:import hc_mappings.json
```

Note: For the example to work you will need the Heroku Connect CLI Plugin.

---

add_arguments(*parser*)

Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

1.3.2 import_mappings

class heroku_connect.management.commands.import_mappings.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Import Heroku Connect mappings to connection.

add_arguments(*parser*)

Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

wait_for_import(*connection_id*, *wait_interval*)

Wait until connection state is no longer `IMPORT_CONFIGURATION`.

Parameters

- **connection_id** (*str*) – Heroku Connect connection to monitor.
- **wait_interval** (*int*) – How frequently to poll in seconds.

Raises

CommandError – If fetch connection information fails.

1.3.3 load_remote_schema

class heroku_connect.management.commands.load_remote_schema.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Load schema from remote Heroku PostgreSQL database.

This command can be useful to load the Heroku Connect database schema into a local development environment.

Example:

```
python manage.py load_remote_schema --app ninja | psql -a
```

Note: This command requires the Heroku CLI and PostgreSQL to be installed.

add_arguments(*parser*)

Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

1.3.4 create_development_schema

class heroku_connect.management.commands.create_development_schema.**Command**(*stdout=None*, *stderr=None*, *no_color=False*, *force_color=False*)

Create Heroku Connect schema for local development.

add_arguments(*parser*)

Entry point for subclassed commands to add custom arguments.

handle(**args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

1.4 Contrib

1.4.1 Health Check

Health Check for Heroku Connect.

Simply add the following to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = [
    # ...
    'heroku_connect.contrib.heroku_connect_health_check',
    # ...
]
```

Note: This features requires django-health-check to be installed.

1.5 Error handling

1.5.1 Trigger Log

Heroku Connect uses the Trigger Log to track changes to connected model instances, as well as its own efforts to sync them to Salesforce.

django-heroku-connect exposes the trigger log tables, which are managed by Heroku Connect, as Django models *TriggerLog* and *TriggerLogArchive*. These models also offer access to database stored procedures provided by Heroku Connect to fix sync errors.

See also:

TriggerLogAbstract for how to use trigger log models.

Admin actions

The admins for the trigger log models (discussed below) offer actions to *ignore* or *retry* failed trigger log entries. The former simply sets the state of the entry to `IGNORED`, if you want to clean up for whatever reason.

Retrying means an attempt to re-sync the change represented by the trigger log. This entails creating an identical trigger log row in `NEW` state, either by simply changing the failed row's state (*TriggerLog*), or by copying an archived row back into the live TriggerLog and setting the archived state to `REQUEUED` (*TriggerLogArchive*). Heroku Connect will then normally process that row.

If this simple attempt at a fix does not work, try to manually call *capture_insert()* or *capture_update()* on a failed trigger log instance.

Models

heroku_connect.models.**TRIGGER_LOG_ACTION = {'DELETE': 'DELETE', 'INSERT': 'INSERT', 'UPDATE': 'UPDATE'}**

The type of change that a trigger log object represents.

heroku_connect.models.**TRIGGER_LOG_STATE = {'FAILED': 'FAILED', 'IGNORE': 'IGNORE', 'IGNORED': 'IGNORED', 'MERGED': 'MERGED', 'NEW': 'NEW', 'PENDING': 'PENDING', 'READONLY': 'READONLY', 'REQUEUE': 'REQUEUE', 'REQUEUED': 'REQUEUED', 'SUCCESS': 'SUCCESS'}**

The sync state of the change tracked by a trigger log entry.

class heroku_connect.models.**TriggerLog**(**args*, ***kwargs*)

Represents entries in the Heroku Connect trigger log.

See also:

TriggerLogAbstract

exception DoesNotExist

exception MultipleObjectsReturned

redo()

Re-sync the change recorded in this trigger log.

This MAY create new TriggerLog instances, or save changes to this instance.

Returns
A TriggerLog instance that represents the re-application; possibly `self`.

class heroku_connect.models.**TriggerLogAbstract**(**args*, ***kwargs*)

Support for accessing the Heroku Connect Trigger Log data and related actions.

Heroku Connect uses a Trigger Log table to track local changes to connected models (that is, in the Heroku database. Such changes are recorded as rows in the trigger log and, for read-write mappings, eventually written back to Salesforce.

Old logs are moved to an archive table (after being processed), from where they are purged eventually (currently 30 days for paid plans, 7 days for demo). Recent logs are modeled by *TriggerLog*; archived logs by *TriggerLogArchive*.

The data represented by these models is maintained entirely by Heroku Connect, and is instrumental to its operations; it should therefore not be modified. A possible exception is the `state` field, which may be changed as detailed in the error handling section in the Heroku Connect documentation.

See also:

- *TriggerLogQuerySet*
- Trigger Log in Heroku Connect docs

capture_insert(*, *exclude_fields=()*)

Apply *TriggerLogAbstract.capture_insert_from_model()* for this log.

classmethod capture_insert_from_model(*table_name*, *record_id*, *, *exclude_fields=()*)

Create a fresh insert record from the current model state in the database.

For read-write connected models, this will lead to the attempted creation of a corresponding object in Salesforce.

Parameters

- **table_name** (*str*) – The name of the table backing the connected model (without schema)
- **record_id** (*int*) – The primary id of the connected model
- **exclude_fields** (*Iterable[str]*) – The names of fields that will not be included in the write record

Returns

A list of the created TriggerLog entries (usually one).

Raises

LookupError – if `table_name` does not belong to a connected model

capture_update(*, *update_fields=()*, *update_columns=()*)

Apply *TriggerLogAbstract.capture_insert_from_model()* for this log.

classmethod capture_update_from_model(*table_name*, *record_id*, *, *update_fields=()*, *update_columns=()*)

Create a fresh update record from the current model state in the database.

For read-write connected models, this will lead to the attempted update of the values of a corresponding object in Salesforce.

Parameters

- **table_name** (*str*) – The name of the table backing the connected model (without schema)
- **record_id** (*int*) – The primary id of the connected model
- **update_fields** (*Iterable[str]*) – If given, the names of fields that will be included in the write record. These will be converted into database column names.
- **update_columns** (*Iterable[str]*) – If given, the names of database column names that will be included in the write record.

Returns

A list of the created TriggerLog entries (usually one).

Raises

LookupError – if `table_name` does not belong to a connected model

get_model()

Fetch the instance of the connected model referenced by this log record.

Returns

The connected instance, or `None` if it does not exists.

related(*, *exclude_self=False*)

Get a QuerySet for all trigger log objects for the same connected model.

Parameters
exclude_self (*bool*) – Whether to exclude this log object from the result list

class heroku_connect.models.**TriggerLogArchive**(**args*, ***kwargs*)

Represents entries in the Heroku Connect trigger log archive.

See also:

TriggerLogAbstract

exception DoesNotExist

exception MultipleObjectsReturned

redo()

Re-sync the change recorded in this trigger log.

Creates a `NEW` live trigger log from the data in this archived trigger log and sets the state of this archived instance to `REQUEUED`.

See also:

TriggerLog.redo()

Returns
The *TriggerLog* instance that was created from the data of this archived log.

class heroku_connect.models.**TriggerLogQuerySet**(*model=None*, *query=None*, *using=None*, *hints=None*)

A QuerySet for trigger log models.

failed()

Filter for log records with sync failures.

related_to(*instance*)

Filter for all log objects of the same connected model as the given instance.

1.6 Exceptions

exception heroku_connect.db.exceptions.**WriteNotSupportedError**

Write actions are not supported on read-only tables.

1.7 Fields

Salesforce fields for Django's ORM.

You can find a list of all Salesforce fields mapped to PostgreSQL here. See Heroku Connect's mapped data types.

class heroku_connect.db.models.fields.**AnyType**(**args*, ***kwargs*)

Salesforce `AnyType` field.

class heroku_connect.db.models.fields.**Checkbox**(**args*, ***kwargs*)

Salesforce `Checkbox` field.

class heroku_connect.db.models.fields.**Currency**(**args*, ***kwargs*)

Salesforce `Currency` field.

This is used for the money value. On the Salesforce side, the actual currency is specified in a `CurrencyIsoCode` field (see Currency Field Type), which is however (currently) not mapped by Heroku Connect.

class heroku_connect.db.models.fields.**Date**(**args*, ***kwargs*)

Salesforce `Date` field.

class heroku_connect.db.models.fields.**DateTime**(**args*, ***kwargs*)

Salesforce `DateTime` field.

Heroku connect create tables with time stamp fields but without time zones, and when it syncs to Salesforce it treats them as UTC. This field will be always making sure that the dates are aware and UTC.

db_type(*connection*)

Return the database column data type for this field, for the provided connection.

class heroku_connect.db.models.fields.**Email**(**args*, ***kwargs*)

Salesforce `Email` field.

class heroku_connect.db.models.fields.**EncryptedString**(**args*, ***kwargs*)

Salesforce `EncryptedString` field.

From the Heroku Connect doc:

> If the user credentials used to authorize Heroku Connect with Salesforce don't have View Encrypted Data permission, then encrypted strings will be received from Salesforce in masked format.
>
> It is possible to update the database with a new plain text value and Salesforce will take care of encryption when the new data is pushed from the database. The plain text value in the database will be overwritten with the masked format when the record is next updated with data from Salesforce.

class heroku_connect.db.models.fields.**ExternalID**(**args*, ***kwargs*)

External ID field for Salesforce objects.

This field uses *uuid.uuid4* as a default UUID function.

The corresponding field in Salesforce must be type `Text(32)`. In Salesforce it will display the UUID as a HEX. It should be set as `External ID` as well as `unique` (case insensitive).

The field should only be required on Salesforce if you want to insert new records only in your application.

Note: Django does not use Database defaults, should you create new records on Salesforce, you need to make sure Salesforce inserts UUIDs or handle empty External ID fields in your Django application.

---

get_db_prep_value(*value*, *connection*, *prepared=False*)

Return field's value prepared for interacting with the database backend.

Used by the default implementations of get_db_prep_save().

class heroku_connect.db.models.fields.**HerokuConnectFieldMixin**(**args*, ***kwargs*)

Base mixin for Heroku Connect fields.

sf_field_name = None

Field's Salesforce API name.

Type
(str)

upsert = False

Whether or not a field is an `externalId`.

Type
(bool)

class heroku_connect.db.models.fields.**ID**(**args*, ***kwargs*)

Salesforce `ID` field.

class heroku_connect.db.models.fields.**Number**(**args*, ***kwargs*)

Salesforce `Number` field.

Allows users to enter any number. Leading zeros are removed.

Numbers in Salesforce are constrained by length and decimal places. Heroku Connect maps those decimal values to `double precision` floats. To have the same accuracy and avoid Salesforce validation rule issues this field uses `Decimal` values and casts them to floats when persisting them to PostgreSQL.

get_db_prep_save(*value*, *connection*)

Return field's value prepared for saving into a database.

get_db_prep_value(*value*, *connection*, *prepared=False*)

Return field's value prepared for interacting with the database backend.

Used by the default implementations of get_db_prep_save().

class heroku_connect.db.models.fields.**Percent**(**args*, ***kwargs*)

Salesforce `Percent` field.

class heroku_connect.db.models.fields.**Phone**(**args*, ***kwargs*)

Salesforce `Phone` field.

class heroku_connect.db.models.fields.**Picklist**(**args*, ***kwargs*)

Salesforce `Picklist` field.

class heroku_connect.db.models.fields.**Text**(**args*, ***kwargs*)

Salesforce `Text` field.

class heroku_connect.db.models.fields.**TextArea**(**args*, ***kwargs*)

Salesforce `Text Area` field.

formfield(***kwargs*)

Return a django.forms.Field instance for this field.

class heroku_connect.db.models.fields.**TextAreaLong**(**args*, ***kwargs*)

Salesforce `Text Area (Long)` and `Text Area (Rich)` field.

class heroku_connect.db.models.fields.**Time**(**args*, ***kwargs*)

Salesforce `Time` field.

class heroku_connect.db.models.fields.**URL**(**args*, ***kwargs*)

Salesforce `URL` field.

1.7.1 Relationship fields

class heroku_connect.db.models.related.**ConstraintlessForeignObjectMixin**(**args*, ***kwargs*)

Ensure Django does not add foreign key database constraints.

class heroku_connect.db.models.related.**Lookup**(**args*, ***kwargs*)

Salesforce `Lookup` field.

class heroku_connect.db.models.related.**MasterDetail**(**args*, ***kwargs*)

Salesforce `Master-Detail` field.

1.8 Models

class heroku_connect.db.models.**HerokuConnectModel**(**args*, ***kwargs*)

Base model for Heroku Connect enabled ORM models in Django.

Example::

from heroku_connect.db import models as hc_models

class MyCustomObject(hc_models.HerokuConnectModel):

sf_object_name = 'My_Custom_Object__c'

custom_date = hc_models.DateTime(sf_field_name='Custom_Date__c')

Note: Subclasses have `Meta.managed` set to `False`.

A default value for `Meta.db_table` is set based on *settings.HEROKU_CONNECT_SCHEMA* and *sf_object_name*.

Note: A model mixin must inherit from `Meta.managed` not *.HerokuConnectModel*. Only the final (not abstract) models should inherit from *.HerokuConnectModel* otherwise build-in fields will clash.

Warning: The Salesforce User and RecordType objects have no `IsDeleted` field. Therefore if *sf_object_name* is set to `User` or `RecordType` the Django ORM representation does not have this field either. You can add the `IsActive` field to your User or RecordType object, but it is not required by Heroku Connect.

classmethod get_heroku_connect_table_name()

Return the table name (without schema) associated with a model class.

Parameters

model_cls – A connected model class object

Raises

`LookupError` – if no table name is associated with the given class

sf_access = 'read_only'

Heroku Connect Object access level.

Access to Heroku Connect tables can be either read or write. Read will only sync from Salesforce to PostgreSQL where write will sync both ways.

Default value is `read_only`.

Accepted values:

- **`read_only`**
  Synchronize only from Salesforce to PostgreSQL.
- **`read_write`**
  Synchronize both ways between Salesforce and PostgreSQL.

`sf_notify_enabled = False`

Use the Salesforce Streaming API to trigger polls when data changes.

With event-based polling enabled, Heroku Connect will continue to poll at the set polling frequency.

`sf_object_name = ''`

Salesforce object API name.

`sf_polling_seconds = 600`

Poll frequency in seconds.

Default: 10 minutes

1.8.1 Model inheritance

Model inheritance in with Heroku Connect models is almost identical to Django's Model inheritance feature. For example you can build model mixins as followed:

```
from heroku_connect.db import models as hc_models


class ExternalIDModelMixin(hc_models.HerokuConnectModel):
    sf_access = hc_model.READ_WRITE

    external_id = hc_models.ExternalID(sf_field_name='External_ID__c')

    class Meta:
        abstract = True


class MyModel(ExternalIDModelMixin):
    sf_object_name = 'My_Object__c'

    data = hc_models.Text(sf_field_name='Data__c')
```

1.8.2 Multi-table inheritance

Multi-table inheritance is a concept where each superclass is a model by itself. It allows building hybrid models that are partly managed by Heroku Connect partly by Django.

Example:

```
from django.db import models
from heroku_connect.db import models as hc_models


class SFObjectModel(hc_models.HerokuConnectModel):
    sf_object_name = 'My_Object__c'
    sf_access = hc_model.READ_WRITE

    external_id = hc_models.ExternalID(sf_field_name='External_ID__c')
    data = hc_models.Text(sf_field_name='Data__c')


class CompoundModel(SFObjectModel):
    hc_model = models.OneToOneField(SFObjectModel, on_delete=models.CASCADE,
                                    to_field='external_id', parent_link=True,
                                    db_constraint=False)
    more_data = models.TextField()

    class Meta:
        managed = True
```

In this scenario `SFObjectModel` is managed by Heroku Connect and `CompoundModel` a hybrid where `CompoundModel.data` is manged by Heroku Connect and `CompoundModel.more_data` is managed by Django.

> **Warning:** You should use your own `parent_link` `OneToOneField` that points to the `upsert`-field of your parent Heroku Connect model as the `id`-field is not not guarantied in to be consistent.

It is also possible to have two concrete Heroku Connect models to Inherit from each other.

1.8.3 Signals

Standard Django model signals won't be triggered for changes made on Salesforce, because Heroku Connect operates on a database-level. If you want to hook into to changes coming from Salesforce, there are several possible solutions:

- Use Salesforce PushTopic Events
- Set up a Salesforce Outbound message (triggered on object change) and a Django SOAP endpoint listening to it. From there, a custom Django signal can be triggered.
- Use PostgreSQL `LISTEN` and `NOTIFY` features. See related documentation and an example Python library

1.9 Router

class heroku_connect.db.router.**HerokuConnectRouter**

Router that prevents write actions on read-only Heroku Connect tables.

The router will raise a *WriteNotSupportedError* error when `save`, `create`, `delete`, `update` or other model or QuerySet methods are called on read-only tables.

Note: You will need to add the router to your `DATABASE_ROUTERS` setting. For example:

```
DATABASE_ROUTERS = ['heroku_connect.db.router.HerokuConnectRouter']
```

See also:

Automatic database routing

db_for_write(*model*, ***hints*)

Prevent write actions on read-only tables.

Raises

WriteNotSupportedError – If models.sf_access is `read_only`.

1.10 Settings

class heroku_connect.conf.**HerokuConnectAppConf**(***kwargs*)

HEROKU_AUTH_TOKEN = ''

Heroku Platform API's Direct Authorization token.

This setting is OPTIONAL. It is required only if you are using the `health-check` application.

To obtain this token, first we need to have a Heroku API token.

API token can be fetched using command *heroku auth:token*. More about Heroku API tokens here - https://devcenter.heroku.com/articles/authentication#retrieving-the-api-token.

After we have our API token, we can fetch the Direct Authentication token by using a process called Token Exchange.

Complete details about this process are provided in this link - https://devcenter.heroku.com/articles/oauth#direct-authorization-token-exchange.

HEROKU_CONNECT_API_ENDPOINT = 'https://connect-eu.heroku.com/api/v3'

Heroku Connect API Endpoint.

This setting is OPTIONAL. It is required only if you are using the `health-check` application. Default is `https://connect-eu.heroku.com/api/v3`.

Check your endpoints at this link - https://devcenter.heroku.com/articles/heroku-connect-api#endpoints.

HEROKU_CONNECT_APP_NAME = 'ninja'

Heroku application name.

This setting will have default based on your environment should you have Dyno Metadata enabled.

HEROKU_CONNECT_ORGANIZATION_ID = '1234567890'

Salesforce Organization ID.

This setting will have default based on the environment variable `HEROKU_CONNECT_ORGANIZATION_ID`.

Note: This is not preset on your Heroku application. You will need to either add a setting or set the environment variable manually.

HEROKU_CONNECT_SCHEMA = 'salesforce'

Database schema used by the Heroku Connect add-on.

This setting is OPTIONAL and based on the environment variable `HEROKU_CONNECT_SCHEMA`. It is required only if you chose a different schema for Heroku Connect tables. Default is `salesforce`.

See also:

https://devcenter.heroku.com/articles/heroku-connect#completing-configuration

1.11 Test Framework

Extensions for Django's test framework to add support for Heroku Connect.

The custom database engine will create the Heroku Connect schema for you. The schema is created right via the `pre_migrate` signal, only when a test database is created. This will work for both Django's build in test suite as well as pytest-django and maybe others.

1.11.1 Utilities

Test utilities for Django and Heroku Connect.

heroku_connect.test.utils.**heroku_cli**(*stdout=''*, *stderr=''*, *exit_code=0*)

Context manager to mock the Heroku CLI command.

Example:

```
import subprocess
from heroku_connect.test.utils import heroku_cli

with heroku_cli(stdout='success', stderr='warning', exit=0):
    process = subprocess.run(
        ['heroku'],
        stdout=subprocess.PIPE,
        stderr=subprocess.PIPE,
    )
assert process.returncode == 0
assert process.stdout == b'success\n'
assert process.stderr == b'warning\n'
```

Parameters

- **stdout** (*str*) – String the command writes to `stdout`.
- **stderr** (*str*) – String the command writes to `stderr`.

- **exit_code** (*int*) – Code that the command will exit with, default 1.

heroku_connect.test.utils.**no_heroku_connect_write_restrictions**()

Allow writing to read-only Heroku Connect tables.

Can be used to create test data, e.g.:

```
def get_test_data():
    with no_heroku_connect_write_restrictions():
        return MyReadOnlyModel.objects.create()
```

1.12 Utilities

Utility methods for Django Heroku Connect.

class heroku_connect.utils.**WriteAlgorithm**(*value*, *names=None*, ***, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

heroku_connect.utils.**create_heroku_connect_schema**(*using='default'*)

Create Heroku Connect schema.

Note: This function is only meant to be used for local development. In a production environment the schema will be created by Heroku Connect.

Parameters
using (*str*) – Alias for database connection.

Returns

True if the schema was created, False if the schema already exists.

Return type
bool

heroku_connect.utils.**get_connected_model_for_table_name**(*table_name*)

Return a connected model's table name (which read and written to by Heroku Connect).

heroku_connect.utils.**get_connection**(*connection_id*, *deep=False*)

Get Heroku Connection connection information.

For more details check the link - https://devcenter.heroku.com/articles/heroku-connect-api#step-8-monitor-the-connection-and-mapping-status

Sample response from API call is below:

```
{
    "id": "<connection_id>",
    "name": "<app_name>",
    "resource_name": "<resource_name>",
    "schema_name": "salesforce",
    "db_key": "DATABASE_URL",
    "state": "IDLE",
```

(continues on next page)

(continued from previous page)

```
    "mappings":[
        {
            "id": "<mapping_id>",
            "object_name": "Account",
            "state": "SCHEMA_CHANGED",
            ...
        },
        {
            "id": "<mapping_id>",
            "object_name": "Contact",
            "state": "SCHEMA_CHANGED",
            ...
        },
        ...
    ]
    ...
}
```

Parameters

- **connection_id** (*str*) – ID for Heroku Connect's connection.
- **deep** (*bool*) – Return information about the connection's mappings, in addition to the connection itself. Defaults to `False`.

Returns

Heroku Connection connection information.

Return type

dict

Raises

- **requests.HTTPError** – If an error occurred when accessing the connection detail API.
- **ValueError** – If response is not a valid JSON.

heroku_connect.utils.**get_connections**(*app*)

Return all Heroku Connect connections setup with the given application.

For more details check the link - https://devcenter.heroku.com/articles/heroku-connect-api#step-4-retrieve-the-new-connection-s-id

Sample response from the API call is below:

```
{
    "count": 1,
    "results":[{
        "id": "<connection_id>",
        "name": "<app_name>",
        "resource_name": "<resource_name>",
        ...
    }],
    ...
}
```

Parameters
app (*str*) – Heroku application name.

Returns
List of all Heroku Connect connections associated with the Heroku application.

Return type
List[dict]

Raises

- **requests.HTTPError** – If an error occurred when accessing the connections API.
- **ValueError** – If response is not a valid JSON.

heroku_connect.utils.**get_heroku_connect_models**()

Return all registered Heroku Connect Models.

Returns
All registered models that are subclasses of *.HerokuConnectModel*. Abstract models are excluded, since they are not registered.

Return type
(Iterator)

heroku_connect.utils.**get_mapping**(*version=1*, *exported_at=None*, *app_name=None*)

Return Heroku Connect mapping for the entire project.

Parameters

- **version** (*int*) – Version of the Heroku Connect mapping, default: 1.
- **exported_at** (*datetime.datetime*) – Time the export was created, default is now().
- **app_name** (*str*) – Name of Heroku application associated with Heroku Connect the add-on.

Returns
Heroku Connect mapping.

Return type
dict

Note: The version does not need to be incremented. Exports from the Heroku Connect website will always have the version number 1.

---

heroku_connect.utils.**import_mapping**(*connection_id*, *mapping*)

Import Heroku Connection mapping for given connection.

Parameters

- **connection_id** (*str*) – Heroku Connection connection ID.
- **mapping** (*dict*) – Heroku Connect mapping.

Raises

- **requests.HTTPError** – If an error occurs uploading the mapping.
- **ValueError** – If the mapping is not JSON serializable.

heroku_connect.utils.**link_connection_to_account**(*app*)

Link the connection to your Heroku user account.

https://devcenter.heroku.com/articles/heroku-connect-api#step-3-link-the-new-add-on-to-your-heroku-user-account

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX

S

T

U

W