flask-simple Documentation

Release 0.0.1

Randall Degges

September 27, 2017

Contents

flask-simple is an extension for Flask that makes using Amazon's SimpleDB NoSQL database *incredibly simple* and enjoyable.

SimpleDB is a NoSQL database, 100% hosted by Amazon, which allows you to store any variable JSON data. All fields are indexed automatically, so no table schema is necessary – and the best part is that you can query records via SQL.

CHAPTER 1

User's Guide

This part of the documentation will show you how to get started with flask-simple. If you're a new flask-simple user, start here!

Quickstart

This section will guide you through everything you need to know to get up and running with flask-simple!

Installation

The first thing you need to do is install flask-simple. Installation can be done through pip, the Python package manager.

To install flask-simple, run:

```
$ pip install flask-simple
```

If you'd like to upgrade an existing installation of flask-simple to the latest release, you can run:

```
$ pip install -U flask-simple
```

Set Environment Variables

In order to run properly, flask-simple requires that you set several environment variables.

The required environment variables are:

- `AWS_ACCESS_KEY_ID` (*your Amazon access key ID*)
- `AWS_SECRET_ACCESS_KEY` (*your Amazon secret access key*)

There is also an optional variable you can set:

- `AWS_REGION` (*defaults to us-east-1*)

These credentials can be grabbed from your AWS Console.

Note: A full list of Amazon regions can be found here: http://docs.aws.amazon.com/general/latest/gr/rande.html#sdb_region

If you're unsure of how to set environment variables, I recommend you check out this StackOverflow question.

Specify Your Domains

The next thing you need to do is tell flask-simple which domains you'll be using.

If you're not sure how domains work with SimpleDB, you should read through the boto SimpleDB tutorial before continuing.

The way you can specify your domains is by creating an array called `SIMPLE_DOMAINS` (*this is what flask-simple uses to set everything up*).

Below is an example:

```
# app.py


from flask import Flask
from flask.ext.simple import Simple

app = Flask(__name__)
app.config['SIMPLE_DOMAINS'] = [
    'users',
    'groups',
]
```

In the above example, I'm defining two SimpleDB domains: `users` and `groups`.

flask-simple will respect *any* boto domains you define.

Initialize simple

Now that you've defined your domains, you can initialize flask-simple in your app.

All you need to do is pass your app to the `simple` constructor:

```
# app.py


from flask import Flask
from flask.ext.simple import Simple

app = Flask(__name__)
app.config['SIMPLE_DOMAINS'] = [
    'users',
    'groups',
]

simple = Simple(app)
```

From this point on, you can interact with SimpleDB through the global `simple` object.

Create Your Domains

If you haven't already created your SimpleDB domains, flask-simple can help you out!

After configuring flask-simple, you can use the following code snippet to create all of your predefined SimpleDB domains:

```
with app.app_context():
    simple.create_all()
```

This works great in bootstrap scripts.

Working with Domains

Now that you've got everything setup, you can easily access your domains in one of two ways: you can either access the domains directly from the `simple` global, or you can access the domains in a dictionary-like format through `simple.domains`.

Below is an example view which creates a new user account:

```
# app.py

@app.route('/create_user')
def create_user():
    simple.users.put_attributes('r@rdegges.com', {
        'username': 'rdegges',
        'first_name': 'Randall',
        'last_name': 'Degges',
        'email': 'r@rdegges.com',
    })

    # or ...

    simple.domains['users'].put_attributes('r@rdegges.com', {
        'username': 'rdegges',
        'first_name': 'Randall',
        'last_name': 'Degges',
        'email': 'r@rdegges.com',
    })
```

Either of the above will work the same.

Note: When storing items in SimpleDB, you need to specify two fields: an item name (*the first parameter*), and the item contents (*a Python dictionary*).

On a related note, you can also use the `simple.domains` dictionary to iterate through all of your domains (*this is sometimes useful*). Here's how you could iterate over your existing SimpleDB domains:

```
# app.py

with app.app_context():
    for domain_name, domain in simple.domains.iteritems():
        print domain_name, domain
```

Deleting Domains

If, for some reason, you'd like to destroy all of your predefined SimpleDB domains, flask-simple can also help you with that.

The below code snippet will destroy all of your predefined SimpleDB domains:

```
# app.py

with app.app_context():
    simple.destroy_all()
```

Note: Please be *extremely* careful when running this – it has the potential to completely destroy your application's data!

Getting Help

Have a question you can't find an answer to? Things not working as expected?

If you've found a bug, or think something might not be working correctly, please file an issue on the official flask-simple issue tracker.

If you need specific help getting something working, please email me directly: r@rdegges.com or send me a tweet: @rdegges.

-Randall

Contributing

Want to contribute to flask-simple? **AWESOME!**

There's only a few things you need to know to get started:

1. All development is done on the Github repo.
2. When you send a pull request, please send it to the `develop` branch – this is where active development happens.
3. Please add tests if you can – it'll make accepting your pull requests a lot easier!

That's about it!

Setup Your Environment

To get started developing, you'll want to fork flask-simple on Github.

After that, you'll need to check out the `develop` branch, as this is where you should 'base' your development from:

```
$ git clone git@github.com:yourusername/flask-simple.git
$ cd flask-simple
$ git fetch origin develop:develop
$ git checkout develop
```

Next, create a new branch that describes the change you want to make:

```
$ git checkout -b bug-fix
```

Next, you'll want to install all of the local dependencies with pip:

```
$ pip install -r requirements.txt
```

After that, you'll want to install the flask-simple package in development mode:

```
$ python setup.py develop
```

Lastly, you'll want to configure your AWS access keys as environment variables so you can run the tests:

```
$ export AWS_ACCESS_KEY_ID=xxx
$ export AWS_SECRET_ACCESS_KEY=xxx
```

Running Tests

After writing some code, you'll need to run the tests to ensure everything is still working ok! This can be done by running:

```
$ python setup.py test
```

From the project's root directory.

Note: The tests take a while to run – this is on purpose, as Amazon rate limits your requests.

Submitting Your Pull Request

Now that you've added an awesome feature or fixed a bug, you probably want to submit your pull request, so let's do it!

First, you'll want to push your topic branch to your Github fork:

```
$ git push origin bug-fix
```

Then, go to Github on your fork, and submit a pull request from your topic branch into the `develop` branch on the main flask-simple repository.

That's it!

Thanks!

I'd also like to give you a big shout out for any contributions you make. You are totally fucking awesome and I love you.

-Randall

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

API

This part of the documentation documents all the public classes, functions, and API details in flask-simple. This documentation is auto generated, and is always a good up-to-date reference.

Configuration

class `flask.ext.simple.manager.`**`Simple`**`(`*app=None*`)`

SimpleDB wrapper for Flask.

`init_app``(`*app*`)`

Initialize this extension.

Parameters **app** (*obj*) – The Flask application.

`init_settings``()`

Initialize all of the extension settings.

`check_settings``()`

Check all user-specified settings to ensure they're correct.

We'll raise an error if something isn't configured properly.

Raises ConfigurationError

`connection`

Our SimpleDB connection.

This will be lazily created if this is the first time this is being accessed. This connection is reused for performance.

`domains`

Our SimpleDB domains.

These will be lazily initializes if this is the first time the tables are being accessed.

`create_all``()`

Create all user-specified SimpleDB domains.

We'll error out if the domains can't be created for some reason.

destroy_all()

Destroy all user-specified SimpleDB domains.

We'll error out if the domains can't be destroyed for some reason.

Errors

class flask.ext.simple.errors.**ConfigurationError**

This exception is raised if the user hasn't properly configured Flask-Simple.

CHAPTER 3

Additional Notes

This part of the documentation covers changes between versions and upgrade information, to help you migrate to newer versions of flask-simple easily.

flask-simple is in the public domain (UNLICENSE), so you can do whatever you want with it!

Change Log

All library changes, in descending order.

Version 0.0.1

Released on July 5, 2014.

- First release!
- Basic functionality.

Upgrade Guide

This page contains specific upgrading instructions to help you migrate between flask-simple releases.

Python Module Index

Index