Flask-PyMongo Documentation

Release 2.3.0

Dan Crosta

Apr 30, 2019

Contents

MongoDB is an open source database that stores flexible JSON-like "documents," which can have any number, name, or hierarchy of fields within, instead of rows of data as in a relational database. Python developers can think of MongoDB as a persistent, searchable repository of Python dictionaries (and, in fact, this is how PyMongo represents MongoDB documents).

Flask-PyMongo bridges Flask and PyMongo and provides some convenience helpers.

Quickstart

First, install Flask-PyMongo:

```
$ pip install Flask-PyMongo
```

Next, add a *PyMongo* to your code:

```python
from flask import Flask
from flask_pymongo import PyMongo

app = Flask(__name__)
app.config["MONGO_URI"] = "mongodb://localhost:27017/myDatabase"
mongo = PyMongo(app)
```

PyMongo connects to the MongoDB server running on port 27017 on localhost, to the database named `myDatabase`. This database is exposed as the *db* attribute.

You can use *db* directly in views:

```python
@app.route("/")
def home_page():
    online_users = mongo.db.users.find({"online": True})
    return render_template("index.html",
        online_users=online_users)
```

Note: Previous versions of Flask-PyMongo required that the MongoDB URI contained a database name; as of 2.2, this requirement is lifted. If there is no database name, the *db* attribute will be `None`.

Compatibility

Flask-PyMongo depends on recent versions of Flask and PyMongo, where "recent" is defined to mean "was released in the last 3 years". Flask-PyMongo *may* work with older versions, but compatibility fixes for older versions will not be accepted, and future changes may break compatibility in older versions.

Flask-PyMongo is tested against supported versions of MongoDB, and Python 2.7 and 3.4+. For the exact list of version combinations that are tested and known to be compatible, see the *envlist* in tox.ini.

Helpers

Flask-PyMongo provides helpers for some common tasks:

Collection.**find_one_or_404**(*args*, ***kwargs*)

Find a single document or raise a 404.

This is like find_one(), but rather than returning None, cause a 404 Not Found HTTP status on the request.

```python
@app.route("/user/<username>")
def user_profile(username):
    user = mongo.db.users.find_one_or_404({"_id": username})
    return render_template("user.html",
        user=user)
```

PyMongo.**send_file**(*filename*, *base='fs'*, *version=-1*, *cache_for=31536000*)

Respond with a file from GridFS.

Returns an instance of the response_class containing the named file, and implement conditional GET semantics (using make_conditional()).

```python
@app.route("/uploads/<path:filename>")
def get_upload(filename):
    return mongo.send_file(filename)
```

Parameters

- **filename** (*str*) – the filename of the file to return

- **base** (*str*) – the base name of the GridFS collections to use

- **version** (*bool*) – if positive, return the Nth revision of the file identified by filename; if negative, return the Nth most recent revision. If no such version exists, return with HTTP status 404.

- **cache_for** (*int*) – number of seconds that browsers should be instructed to cache responses

PyMongo.**save_file**(*filename*, *fileobj*, *base='fs'*, *content_type=None*, ***kwargs*)

> Save a file-like object to GridFS using the given filename.

```python
@app.route("/uploads/<path:filename>", methods=["POST"])
def save_upload(filename):
    mongo.save_file(filename, request.files["file"])
    return redirect(url_for("get_upload", filename=filename))
```

> ### Parameters
>
> - **filename** (*str*) – the filename of the file to return
>
> - **fileobj** (*file*) – the file-like object to save
>
> - **base** (*str*) – base the base name of the GridFS collections to use
>
> - **content_type** (*str*) – the MIME content-type of the file. If None, the content-type is guessed from the filename using guess_type()
>
> - **kwargs** – extra attributes to be stored in the file's document, passed directly to gridfs.GridFS.put()

class flask_pymongo.**BSONObjectIdConverter**(*map*)

> A simple converter for the RESTful URL routing system of Flask.

```python
@app.route("/<ObjectId:task_id>")
def show_task(task_id):
    task = mongo.db.tasks.find_one_or_404(task_id)
    return render_template("task.html", task=task)
```

Valid object ID strings are converted into ObjectId objects; invalid strings result in a 404 error. The converter is automatically registered by the initialization of *PyMongo* with keyword ObjectId.

Configuration

You can configure Flask-PyMongo either by passing a MongoDB URI to the *PyMongo* constructor, or assigning it to the `MONGO_URI` Flask configuration variable

You may also pass additional keyword arguments to the `PyMongo` constructor. These are passed directly through to the underlying `MongoClient` object.

Note: By default, Flask-PyMongo sets the `connect` keyword argument to `False`, to prevent PyMongo from connecting immediately. PyMongo itself is not fork-safe, and delaying connection until the app is actually used is necessary to avoid issues. If you wish to change this default behavior, pass `connect=True` as a keyword argument to `PyMongo`.

You can create multiple `PyMongo` instances, to connect to multiple databases or database servers:

```python
app = Flask(__name__)

# connect to MongoDB with the defaults
mongo1 = PyMongo(app, uri="mongodb://localhost:27017/databaseOne")

# connect to another MongoDB database on the same host
mongo2 = PyMongo(app, uri="mongodb://localhost:27017/databaseTwo")

# connect to another MongoDB server altogether
mongo3 = PyMongo(app, uri="mongodb://another.host:27017/databaseThree")
```

Each instance is independent of the others and shares no state.

API

5.1 Classes

class `flask_pymongo.`**`PyMongo`**(*app=None*, *uri=None*, **args*, ***kwargs*)

Manages MongoDB connections for your Flask app.

PyMongo objects provide access to the MongoDB server via the *db* and *cx* attributes. You must either pass the `Flask` app to the constructor, or call *init_app()*.

PyMongo accepts a MongoDB URI via the `MONGO_URI` Flask configuration variable, or as an argument to the constructor or `init_app`. See *init_app()* for more detail.

`cx`

The `MongoClient` connected to the MongoDB server.

`db`

The `Database` if the URI used named a database, and `None` otherwise.

`init_app`(*app*, *uri=None*, **args*, ***kwargs*)

Initialize this *PyMongo* for use.

Configure a `MongoClient` in the following scenarios:

1. If `uri` is not `None`, pass the `uri` and any positional or keyword arguments to `MongoClient`

2. If `uri` is `None`, and a Flask config variable named `MONGO_URI` exists, use that as the `uri` as above.

The caller is responsible for ensuring that additional positional and keyword arguments result in a valid call.

Changed in version 2.2: The `uri` is no longer required to contain a database name. If it does not, then the *db* attribute will be `None`.

Changed in version 2.0: Flask-PyMongo no longer accepts many of the configuration variables it did in previous versions. You must now use a MongoDB URI to configure Flask-PyMongo.

`save_file`(*filename*, *fileobj*, *base='fs'*, *content_type=None*, ***kwargs*)

Save a file-like object to GridFS using the given filename.

```python
@app.route("/uploads/<path:filename>", methods=["POST"])
def save_upload(filename):
    mongo.save_file(filename, request.files["file"])
    return redirect(url_for("get_upload", filename=filename))
```

Parameters

- **filename** (*str*) – the filename of the file to return

- **fileobj** (*file*) – the file-like object to save

- **base** (*str*) – base the base name of the GridFS collections to use

- **content_type** (*str*) – the MIME content-type of the file. If `None`, the content-type is guessed from the filename using `guess_type()`

- **kwargs** – extra attributes to be stored in the file's document, passed directly to `gridfs.GridFS.put()`

send_file (*filename*, *base='fs'*, *version=-1*, *cache_for=31536000*)
> Respond with a file from GridFS.

> Returns an instance of the `response_class` containing the named file, and implement conditional GET semantics (using `make_conditional()`).

```python
@app.route("/uploads/<path:filename>")
def get_upload(filename):
    return mongo.send_file(filename)
```

Parameters

- **filename** (*str*) – the filename of the file to return

- **base** (*str*) – the base name of the GridFS collections to use

- **version** (*bool*) – if positive, return the Nth revision of the file identified by filename; if negative, return the Nth most recent revision. If no such version exists, return with HTTP status 404.

- **cache_for** (*int*) – number of seconds that browsers should be instructed to cache responses

5.2 Wrappers

Flask-PyMongo wraps PyMongo's `MongoClient`, `Database`, and `Collection` classes, and overrides their attribute and item accessors. Wrapping the PyMongo classes in this way lets Flask-PyMongo add methods to `Collection` while allowing user code to use MongoDB-style dotted expressions.

```python
>>> type(mongo.cx)
<type 'flask_pymongo.wrappers.MongoClient'>
>>> type(mongo.db)
<type 'flask_pymongo.wrappers.Database'>
>>> type(mongo.db.some_collection)
<type 'flask_pymongo.wrappers.Collection'>
```

class flask_pymongo.wrappers.**Collection**(*...*)
> Sub-class of PyMongo `Collection` with helpers.

find_one_or_404(**args*, ***kwargs*)
>
> Find a single document or raise a 404.
>
> This is like `find_one()`, but rather than returning `None`, cause a 404 Not Found HTTP status on the request.

```python
@app.route("/user/<username>")
def user_profile(username):
    user = mongo.db.users.find_one_or_404({"_id": username})
    return render_template("user.html",
        user=user)
```

5.3 History and Contributors

Changes:

- 2.3.0: April 24, 2019

  - Update version compatibility matrix in tests, drop official support for PyMongo less than 3.3.x.

- 2.2.0: November 1, 2018

  - #117 Allow URIs without database name.

- 2.1.0: August 6, 2018

  - #114 Accept keyword arguments to *save_file()* (Andrew C. Hawkins).

- 2.0.1: July 17, 2018

  - #113 Make the `app` argument to `PyMongo` optional (yarobob).

- 2.0.0: July 2, 2018

  This release is not compatible with Flask-PyMongo 0.5.x or any earlier version. You can see an explanation of the reasoning and changes in issue #110.

  - Only support configuration via URI.

  - Don't connect to MongoDB by default.

  - Clarify version support of Python, Flask, PyMongo, and MongoDB.

  - Readability improvement to `README.md` (MinJae Kwon).

- 0.5.2: May 19, 2018

  - #102 Return 404, not 400, when given an invalid input to *BSONObjectIdConverter* (Abraham Toriz Cruz).

- 0.5.1: May 24, 2017

  - #93 Supply a default `MONGO_AUTH_MECHANISM` (Mark Unsworth).

- 0.5.0: May 21, 2017

  This will be the last 0.x series release. The next non-bugfix release will be Flask-PyMongo 2.0, which will introduce backwards breaking changes, and will be the foundation for improvements and changes going forward. Flask-PyMongo 2.0 will no longer support Python 2.6, but will support Python 2.7 and Python 3.3+.

  - #44, #51 Redirect `/` to `/HomePage` in the wiki example (David Awad)

  - #76 Build on more modern Python versions (Robson Roberto Souza Peixoto)

- #79, #84, #85 Don't use `flask.ext` import paths any more (ratson, juliascript)

  - #40, #83, #86 Fix options parsing from `MONGO_URI` (jobou)

  - #72, #80 Support `MONGO_SERVER_SELECTION_TIMEOUT_MS` (Henrik Blidh)

  - #34, #64, #88 Support from `MONGO_AUTH_SOURCE` and `MONGO_AUTH_MECHANISM` (Craig Davis)

  - #74, #77, #78 Fixed `maxPoolSize` in PyMongo 3.0+ (Henrik Blidh)

  - #82 Fix "another user is already authenticated" error message.

  - #54 Authenticate against "admin" database if no `MONGO_DBNAME` is provided.

- 0.4.1: January 25, 2016

  - Add the connect keyword: #67.

- 0.4.0: October 19, 2015

  - Flask-Pymongo is now compatible with pymongo 3.0+: #63.

- 0.3.1: April 9, 2015

  - Flask-PyMongo is now tested against Python 2.6, 2.7, 3.3, and 3.4.

  - Flask-PyMongo installation now no longer depends on nose.

  - #58 Update requirements for PyMongo 3.x (Emmanuel Valette).

  - #43 Ensure error is raised when URI database name is parsed as 'None' (Ben Jeffrey).

  - #50 Fix a bug in read preference handling (Kevin Funk).

  - #46 Cannot use multiple replicaset instances which run on different ports (Mark Unsworth).

  - #30 Configuation Error with MONGO_READ_PREFERENCE (Mark Unsworth).

- 0.3.0: July 4, 2013

  - This is a minor version bump which introduces backwards breaking changes! Please read these change notes carefully.

  - Removed read preference constants from Flask-PyMongo; to set a read preference, use the string name or import contants directly from `pymongo.read_preferences.ReadPreference`.

  - #22 (partial) Add support for `MONGO_SOCKET_TIMEOUT_MS` and `MONGO_CONNECT_TIMEOUT_MS` options (ultrabug).

  - #27 (partial) Make Flask-PyMongo compatible with Python 3 (Vizzy).

- 0.2.1: December 22, 2012

  - #19 Added `MONGO_DOCUMENT_CLASS` config option (jeverling).

- 0.2.0: December 15, 2012

  - This is a minor version bump which may introduce backwards breaking changes! Please read these change notes carefully.

  - #17 Now using PyMongo 2.4's `MongoClient` and `MongoReplicaSetClient` objects instead of `Connection` and `ReplicaSetConnection` classes (tang0th).

  - #17 Now requiring at least PyMongo version 2.4 (tang0th).

  - #17 The wrapper class `flask_pymongo.wrappers.Connection` is renamed to `flask_pymongo.wrappers.MongoClient` (tang0th).

- – #17 The wrapper class `flask_pymongo.wrappers.ReplicaSetConnection` is renamed to `flask_pymongo.wrappers.MongoReplicaSetClient` (tang0th).

- – #18 `MONGO_AUTO_START_REQUEST` now defaults to `False` when connecting using a URI.

- 0.1.4: December 15, 2012

  - – #15 Added support for `MONGO_MAX_POOL_SIZE` (Fabrice Aneche)

- 0.1.3: September 22, 2012

  - – Added support for configuration from MongoDB URI.

- 0.1.2: June 18, 2012

  - – Updated wiki example application

  - – #14 Added examples and docs to PyPI package.

- 0.1.1: May 26, 2012

  - – Added support for PyMongo 2.2's "auto start request" feature, by way of the `MONGO_AUTO_START_REQUEST` configuration flag.

  - – #13 Added BSONObjectIdConverter (Christoph Herr)

  - – #12 Corrected documentation typo (Thor Adam)

- 0.1: December 21, 2011

  - – Initial Release

Contributors:

- jeverling

- tang0th

- Fabrice Aneche

- Thor Adam

- Christoph Herr

- Mark Unsworth

- Kevin Funk

- Ben Jeffrey

- Emmanuel Valette

- David Awad

- Robson Roberto Souza Peixoto

- juliascript

- Henrik Blidh

- jobou

- Craig Davis

- ratson

- Abraham Toriz Cruz

- MinJae Kwon

- yarobob

- Andrew C. Hawkins