Flask-Journey Documentation

Release 0.1.1

Robert Wikman

Mar 11, 2018

Contents

The two core components of Journey, namely `route` and `BlueprintBundle`, are not dependent of each other. However, there might be non-breaking code implemented in the future that enables the components integrate, either in a unidirectional or bidirectional way. This, and the fact that they operate in the same field, was the reason for adding them both to this extension.

CHAPTER 1

Installation

Use pip to install the extension:

```
$ pip install flask-journey
```

CHAPTER 2

Usage

Flask-Journey is managed through a `Journey` instance.

This component and procedure is only necessary if you plan on using the `journey.BlueprintBundle` for managing blueprints.

Anyway - if you do want to use it and you're utilizing the application factory pattern, then you probably want to set up `Journey` with init_app():

```
from flask import Flask
from flask_journey import Journey

from .bundles import bundle1, bundle2

app = Flask(__name__)
journey = Journey()
journey.attach_bundle(bundle1)
journey.attach_bundle(bundle2)
journey.init_app(app)
```

You may also set it up directly, passing a list of bundles to the `Journey` constructor:

```
app = Flask(__name__)
journey = Journey(app, bundles=[bundle1, bundle2])
```

CHAPTER 3

Examples

3.1 Bundling blueprints

There are various benefits of using the Journey BlueprintBundle, and in many cases just one BlueprintBundle is enough.

- It can be used to easily segregate your blueprint registration code from the other parts of your application.
- It helps you group blueprints logically.
- It allows you to utilize the Journey API (currently only to list routes)

```
# file: api/bundles.py

from flask_journey import BlueprintBundle

from .users import bp as users
from .groups import bp as groups
from .companies import bp as companies
from .danger import bp as danger

v1 = BlueprintBundle(path='/api/v1', description="API v1, stable")
v1.attach_bp(users, description='Users CRUD')
v1.attach_bp(groups)
v1.attach_bp(companies, description='Companies API')

v2 = BlueprintBundle(path='/api/v2', description="API v2, beta")
v2.attach_bp(users, description='Users CRUD')
v2.attach_bp(groups)
v2.attach_bp(companies, description='Companies API')
v2.attach_bp(danger, description='Dangerous testing API, not for production use')
```

3.2 Importing bundles

Importing and registering bundles (and along with their blueprints) is easy as pie:

```
# file: api/__init__.py

from flask import Flask
from .bundles import v1, v2

app = Flask(__name__)
journey = Journey()
journey.attach_bundle(v1)
journey.attach_bundle(v2)
journey.init_app(app)
```

3.3 Using the `Journey.route` decorator

The `route` component, as mentioned previously, is not dependent of the Journey blueprint manager. However, functions decorated with `flask_journey.route` can of course, just as `flask.Blueprint.route`, be added to your app with the help of Journey.

Regular marshmallow type schemas:

```
# file: api/users/schemas.py

from marshmallow import Schema, fields, validate

class QuerySchema(Schema):
    first_name = fields.String(required=False)
    last_name = fields.String(required=False)


class UserSchema(Schema):
    id = fields.Integer(required=True)
    first_name = fields.String(required=True)
    last_name = fields.String(required=True)
    user_name = fields.String(required=True)
```

The `flask_journey.route` enables easy (de)serialization and validation with the help of the Marshmallow library.

```
# api/users/views.py

from flask import Blueprint
from flask_journey import route
from db import create_user, get_user

from .schema import UserSchema

bp = Blueprint('users', __name__)

@route(bp, '/', methods=['GET'], query_schema=QuerySchema(strict=True), marshal_
↪with=UserSchema(many=True))
def get_many(__query=None):
    return get_users(**__query['data'])
```

```
@route(bp, '/', methods=['POST'], body_schema=UserSchema(strict=True), marshal_
↪with=UserSchema())
def create(__body=None):
    return create_user(**__body['data'])
```

3.4 Real examples

Full and usable examples can be found here

3.5 Route decorator

`flask_journey.utils.route`(*bp*, **args*, ***kwargs*)

Journey route decorator

Enables simple serialization, deserialization and validation of Flask routes with the help of Marshmallow.

If a schema (body_schema and/or query_schema) was passed to the decorator, the corresponding :class‘marshmallow.Schema‘ object gets passed to the decorated function:

__query - kwarg if *query_schema* was passed __body - kwarg if *body_schema* was passed

Parameters

- **bp** – `flask.Blueprint` object
- **args** – args to pass along to *Blueprint.route*
- **kwargs** –
  - **strict_slashes** Enable / disable strict slashes (default False)
  - **body_schema** Deserialize JSON body with this schema
  - **query_schema** Deserialize Query string with this schema
  - **marshal_with** Serialize the output with this schema

3.6 Journey API

class `flask_journey.Journey`(*app=None*, *bundles=None*)

Central controller class. Registers bundles and exposes properties for listing routes.

Parameters **app** – App to pass directly to Journey

attach_bundle(*bundle*)

Attaches a bundle object

Parameters **bundle** – *`flask_journey.BlueprintBundle`* object

Raises

- IncompatibleBundle if the bundle is not of type *BlueprintBundle*

static get_blueprint_routes(*app*, *base_path*)

Returns detailed information about registered blueprint routes matching the *BlueprintBundle* path

Parameters

- **app** – App instance to obtain rules from
- **base_path** – Base path to return detailed route info for

Returns List of route detail dicts

static get_child_path(*bp*)

Strips leading slashes from url_prefix, if it has been set, and returns. If not, return *Blueprint.name*.

Parameters **bp** – `flask.Blueprint` object

Returns blueprint name

init_app(*app*)

Initializes Journey extension

Parameters **app** – App passed from constructor or directly to init_app

routes_detailed

Returns a detailed list bundles and its blueprints and routes

Returns List of blueprint routes

routes_simple

Returns simple info about registered blueprints

Returns Tuple containing endpoint, path and allowed methods for each route

3.7 BlueprintBundle API

class `flask_journey`.**BlueprintBundle**(*path='/'*, *description=''*)

Creates a BlueprintBundle at the path specified

Parameters **path** – blueprint base path

attach_bp(*bp*, *description=''*)

Attaches a flask.Blueprint to the bundle

Parameters

- **bp** – `flask.Blueprint` object
- **description** – Optional description string

static sanitize_path(*path*)

Performs sanitation of the route path after validating

Parameters **path** – path to sanitize

Returns sanitized path

3.8 Exceptions

Python Module Index

f

Index