Routing

`route()` decorator is used to bind a `function` to a `URL`

Example:

`@app.r out e('/')`

By default a route only answers to GET requests, but you can provide the `methods` argument.

`@app.r out e(' /lo gin', method s=[ 'GET', 'POST'])`

flask-restful

⚡ With Flask-Restful you can create RESTful API with your Flask app

Create an Flask App

`app = Flask( __n ame__)`

Then create the Api object passing the App object

`api = Api(app)`

Then you can create `Resource`s and add them to the API

`class NewsFi nde r(R eso urce): pass`

`api.ad d_r eso uce (Ne wsF inder, '/', '/news')`

You can implement each HTTP verb with functions named like the verb, but in lowercase

Example:

`def get(self): pass`

`def put(self, id): pass`

To parse arguments passed by the url use

`parser = reqpar se.R eq ues tPa rser()`

You can pass `parse_ arg s(s tri ct= True)` to throw an error if arguments that were not defined by you has been passed

Add the arguments with `parser.ad d_a rgu men ts( 'li mit', typ e=i nt, hel p='He lp Text', requir ed= True)`

You can specify the location to look for this argument with `add_ar gum ent s('l imi t', locati on= 'he aders')`

Example locations: form, args, headers, session, cookies, files

Then inside the function you can `args = parser.pa rse _args()` to get the parsed args

This variable `args` will become a dictionary with the values, ccess via `args[ 'li mit']`

Imports

`from flask_ restful import Api, Resource, reqparse`

Flask_jwt

`from flask import Flask`
`from flask_ restful import Api, Resource`
`from flask_jwt import JWT, jwt_re quired, curren -`
`t_i dentity`
`app = Flask( __n ame__)`

Flask_jwt (cont)

`> app.config['SECRET_KEY'] = 'my-secret'`
`api = Api(app, prefix='/api/v1')`
`USER_DATA = {`
`"amicheletti": "coracaopeludo"`
`}`
`class User(object):`
`def __init__(self, id):`
`self.id = id`
`def __str__(self):`
`return "User (id={})".format(self.id)`
`def verify(username, password):`
`if not (username and password):`
`return False`
`if (USER_DATA.get(username) == password):`
`return User(id=123)`
`def identity(payload):`
`user_id = payload['identity']`
`return { "uid": user_id }`
`jwt = JWT(app, verify, identity)`
`class UltimateQuestion(Resource):`
`@jwt_required()`
`def get(self):`
`return { "meaningoflife" : 42, "who_asked" : dict(current_identity) }`
`api.add_resource(UltimateQuestion, '/', '/life')`
`if __name__ == "__main__":`
`app.run(debug=True)`

You must have an `authen tic ati on_ han dler()` which takes 2 arguments and a `identi ty_ han dler()` which takes 1 argument

Authentication handler must return an Object that has an id attribute

Identity handler return what is going to be send to `'identity'` key of the JSON

To get the token, curl POST to the `/auth` like this:

`curl -H " Con ten t-type: applic ati on/ jso n" -X`
` POST -d '{"u ser nam e":"a mic hel ett i","p ass -`
`wor d":"c ora cao pel udo "}' http:/ /12 7.0.0.1 :`
`5 000 /auth`

By amicheletti

cheatography.com/amicheletti/

URL Building

When routing some function to a URL, you can use function `url_for()` to generate the URL to that function.

Example, if you have something like

```
@app.r out e(' /us er/ <us ern ame >') def profil e(u ser
name): pass
```
 you use `url_fo r(' pro file', userna me= " And e")` to get the URL for that route.

That way you can avoid having to change the hardcoded URL everywhere in the code.

File Uploads

To handle file uploads with Flask, the HTML form must be set with `enctyp e="m ult ipa rt/ - for m-d ata "`

Then you can use it from a dictionary in `reques ts.f iles`

Example:

```
f = reques t.f ile s[' the _file'] f.save ('/ var /ww w/u plo ads /up loa -
ded _fi le.t xt')
```

Redirects and Errors

`redire ct( 'url')` Pass a URL to this function to redirect a user

`abort(401)` This will abort the request early with an error code

To customize the error page use `@app.e rro rha ndl er(404)`, but don't forget to pass the error code. Example:

```
return render _te mpl ate ('p age _no t_f oun d.h t
ml'), 404
```

virtualenv

`virtualenv my_project`	Create environment named my_project
`-p /usr/b in/ pyt hon3.5`	Pass this argument to define Python to be used
`source my_pro jec t/b in/ act i vate`	Start using the environment
`deactivate`	To leave your environment
`pip freeze > requir eme nts.txt`	Freeze your requirements to a file
`pip install -r requir eme nts.t xt`	Install using the requirements file

Blueprint

Blueprints are objects similar to the Flask application object, but are not actual application. They can record operations and endpoints routing and deliver resources, and then they are registered to the application (can be registered multiple times) under a specific URL.

Create a blueprint:

```
feed_b lue print = Bluepr int ('f eed', __name__)
```

Use blueprint like an Flask app object:

```
@feed_ blu epr int.ro ute ('\')
```

Register the blueprint to the real application

```
app.re gis ter _bl uep rin t(f eed _bl uep rint, url_
i x=' /feed')
```

Blueprint root folder

```
feed_b lue pri nt.r oo t_path
```

To build url for Blueprints, put the name used in the object creation befo function name:

```
url_fo r(' fee d.i ndex')
```

Also you can use the error handler just like the Flask object

```
@feed_ blu epr int.er ror han dle r(404)
```

JWT

JWT stands for JSON Web Token, that are used to securely transmit JSON information between two parties or authenticate

They consist in three parts: Header, Payload and Signature. These three parts are JSON object which are then `Base64URL` encoded and included to

the token `header.pa ylo ad.s ig nature`

- Header

In Header, you generally have two information:

the type of the token and the algorithm used

```
{
" alg " : " HS2 56",
 " typ " : " JWT "
}
```

- Payload

In Payload you have "claims" about an Entity (the user for example) and other metadata.

Example:

```
{
" id": " 123 456 789 0",
  " nam e": "John Doe",
 " adm in": true
```

By **amicheletti**

cheatography.com/amicheletti/

Published 11th July, 2017.
Last updated 18th July, 2017.
Page 2 of 3.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

JWT (cont)

```
}
```

There are Reserved Claims (predefined), Public Claims (defined by users at IANA JSON Web Token Registry) and Private Claims (custom claims agreed by both parties)

- Signature

To generate the signature, take the encoded header and payload, a secret and encode all that with the algorithm used.

Example: `HMACSH A256( base64 Url Enc ode (he ader) + "." + base64 Url Enc ode (p a yload), secret)`

- Usage

Now when the user wants to access a protected route or resource, the user agent must send the JWT typically in the *Authorization header*, using the *Bearer* schema, like this:

```
Author iza tion: Bearer <to ken>
```

Variable Rules

`<us ern ame>`	default for `<st rin g:>`
`<st rin g:>`	accepts any text without slash
`<in t:>`	accepts integers
`<fl oat :>`	floating point values
`<pa th:>`	like `<st rin g:>` but accept slashes
`<an y:>`	matches one of the items provided
`<uu id:>`	accepts UUID strings

Add variable parts to a URL. You can also specify a converter to the variable.

Request Object

The request object is available when routing passing `method` argument.

`reques t.m ethod` is the HTTP method (POST, GET...)

`reques t.fòrm` Use this to access the form data passed

`reques t.a rgs.ge t(' key', '')` Use this to access parameters passed by url `?key=value`

```
from flask import request
```

Logging

```
app.logger.debug('A value for debugging')
app.lo gge r.w arn ing('A warning occurred (%d
apples)', 42)
app.lo gge r.e rro r('An error occurred')
```