CMS-GEN

Release 1

cms-generator

Apr 11, 2021

CMS Generator group information

Current version is 1.

This page intends to:

- Provides Generator related information.
- Supports CMS user in term of GEN related technicality.
- Provides a platform for FAQ.

Hightlight

- Monte Carlo generator tutorial

1.1 Monte Carlo Generator Tutorial

There are also many other demo projects, give them a try!

We have created for you a selection of fun projects, that can show you how to create application from the *blog* to the applications related to data science. Please feel free to add your open source example project by making Pull Request.

- Shortify - *URL shortener* with *Redis* storage.
- Moderator - UI and API for classification of offensive and toxic comments using *Kaggle* data and *scikit-learn*.
- Moderator bot - Slack bot for moderating offensive and toxic comments using provided model from *Moderator AI*
- Motortwit - *Twitter* clone with *MongoDB* storage.
- Imagetagger - Example how to deploy deep learning model with *aiohttp*.
- Chat - Simple *chat* using websockets.
- Polls - Simple *polls* application with PostgreSQL storage.
- Blog - The *blog* application with *PostgreSQL* storage and *Redis* session store.
- Graphql - The simple real-time chat that based on the *GraphQL* api and *Apollo client*.

1.2 FAQ

1.2.1 Are there plans for an @app.route decorator like in Flask?

As of aiohttp 2.3, `RouteTableDef` provides an API similar to Flask's `@app.route`. See aiohttp-web-alternative-routes-definition.

Unlike Flask's `@app.route`, `RouteTableDef` does not require an `app` in the module namespace (which often leads to circular imports).

Instead, a `RouteTableDef` is decoupled from an application instance:

```
routes = web.RouteTableDef()

@routes.get('/get')
async def handle_get(request):
    ...


@routes.post('/post')
async def handle_post(request):
    ...

app.router.add_routes(routes)
```

1.2.2 Does aiohttp have a concept like Flask's "blueprint" or Django's "app"?

If you're writing a large application, you may want to consider using nested applications, which are similar to Flask's "blueprints" or Django's "apps".

See: aiohttp-web-nested-applications.

1.2.3 How do I create a route that matches urls with a given prefix?

You can do something like the following:

```
app.router.add_route('*', '/path/to/{tail:.+}', sink_handler)
```

The first argument, `*`, matches any HTTP method (*GET, POST, OPTIONS*, etc). The second argument matches URLS with the desired prefix. The third argument is the handler function.

1.2.4 Where do I put my database connection so handlers can access it?

`aiohttp.web.Application` object supports the `dict` interface and provides a place to store your database connections or any other resource you want to share between handlers.

```
async def go(request):
    db = request.app['db']
    cursor = await db.cursor()
    await cursor.execute('SELECT 42')
    # ...
    return web.Response(status=200, text='ok')


async def init_app(loop):
    app = Application(loop=loop)
    db = await create_connection(user='user', password='123')
    app['db'] = db
    app.router.add_get('/', go)
    return app
```