aiohttp example

TodoMVC

While exploring RESTful APIs, we will use a simple "todo" application as an example. This example is compatible with TodoMVC (http://todomvc.com/). TodoMVC is "a project which offers the same Todo application implemented using MV* concepts in most of the popular JavaScript MV* frameworks of today".

You can try your API using any of the TodoMVC clients (e.g. Todo-Backend's client (http://www.todobackend.com/client/)). You can also compare different frameworks with the Todo-Backend (http://www.todobackend.com/) project as they offer "a shared example to showcase backend tech stacks" as they provides a similar RESTful API.

Same-origin policy

In order to test the API from an external webpage, you should enable CORS (https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) on your server. With aiohttp, this is achieved by installing the aiohttp-cors (https://github.com/aio-libs/aiohttp-cors) plugin (`pip3 install aiohttp-cors`).

Here is a minimal example to create an app with CORS enabled:

```python
from aiohttp import web
import aiohttp_cors

async def handle(request):
    name = request.match_info.get('name', "Anonymous")
    text = "Hello, " + name
    return web.Response(text=text)

app = web.Application()
# Configure default CORS settings.
cors = aiohttp_cors.setup(app, defaults={
    "*": aiohttp_cors.ResourceOptions(
            allow_credentials=True,
            expose_headers="*",
            allow_headers="*",
            allow_methods="*",
        )
})
cors.add(app.router.add_get('/{name}', handle))
```

Running the example

Let's grab the todo example from [GitHub (https://github.com/DurandA/todobackend-aiohttp-min/blob/master/aiotodo.py)](https://github.com/DurandA/todobackend-aiohttp-min/blob/master/aiotodo.py) (slightly modified version of [from alec.thoughts import * (http://justanr.github.io/getting-start-with-aiohttpweb-a-todo-tutorial)](http://justanr.github.io/getting-start-with-aiohttpweb-a-todo-tutorial) or copy the following code to a `aiotodo.py` file and run it with `python3 aiotodo.py`.

```
import logging
from aiohttp import web
import aiohttp_cors

TODOS = {
    0: {'title': 'build an API', 'order': 1, 'completed': False},
    1: {'title': '?????', 'order': 2, 'completed': False},
    2: {'title': 'profit!', 'order': 3, 'completed': False}
}

def get_all_todos(request):
    return web.json_response([
        {'id': key, **todo} for key, todo in TODOS.items()
    ])

def remove_all_todos(request):
    TODOS.clear()
    return web.Response(status=204)

def get_one_todo(request):
    id = int(request.match_info['id'])

    if id not in TODOS:
        return web.json_response({'error': 'Todo not found'}, status=404)

    return web.json_response({'id': id, **TODOS[id]})

async def create_todo(request):
    data = await request.json()

    if 'title' not in data:
        return web.json_response({'error': '"title" is a required field'})
    title = data['title']
    if not isinstance(title, str) or not len(title):
        return web.json_response({'error': '"title" must be a string with at
least one character'})

    data['completed'] = bool(data.get('completed', False))
    new_id = max(TODOS.keys(), default=0) + 1
    data['url'] =
str(request.url.join(request.app.router['one_todo'].url_for(id=str(new_id))))

    TODOS[new_id] = data

    return web.Response(
        headers={'Location': data['url']},
        status=303
    )

async def update_todo(request):
    id = int(request.match_info['id'])

    if id not in TODOS:
        return web.json_response({'error': 'Todo not found'}, status=404)

    data = await request.json()
    TODOS[id].update(data)

    return web.json_response(TODOS[id])

def remove_todo(request):
    id = int(request.match_info['id'])

    if id not in TODOS:
        return web.json_response({'error': 'Todo not found'})

    del TODOS[id]

    return web.Response(status=204)

app = web.Application()

# Configure default CORS settings.
cors = aiohttp_cors.setup(app, defaults={
    "*": aiohttp_cors.ResourceOptions(
            allow_credentials=True,
            expose_headers="*",
            allow_headers="*",
            allow_methods="*",
        )
})

cors.add(app.router.add_get('/todos/', get_all_todos, name='all_todos'))
cors.add(app.router.add_delete('/todos/', remove_all_todos,
name='remove_todos'))
cors.add(app.router.add_post('/todos/', create_todo, name='create_todo'))
cors.add(app.router.add_get('/todos/{id:\d+}', get_one_todo, name='one_todo'))
cors.add(app.router.add_patch('/todos/{id:\d+}', update_todo,
name='update_todo'))
cors.add(app.router.add_delete('/todos/{id:\d+}', remove_todo,
name='remove_todo'))

logging.basicConfig(level=logging.DEBUG)
web.run_app(app, port=8080)
```

Test your API on your browser with the TodoMVC client http://www.todobackend.com/client/. Point to `127.0.0.1:8080/todos/` (`-P 8080` parameter starts the server on port 8080). Use you browser built-in network debugger to see how the application client and backend are interacting. Alternatively, use an HTTP debugger like Fiddler (http://www.telerik.com/fiddler) or any network analyzer that can parse HTTP requests. Mastering one of these tools is necessary to complete your project without headaches.

You can also query the *todo* API directly with HTTPie:

description	**verb**	**path**	**command**
List all todos	GET	/todos/	http 127.0.0.1:8080/todos/
Create a todo	POST	/todos/	http POST 127.0.0.1:8080/todos/ title=blah
Fetch a todo	GET	/todos/{todo_id}	http 127.0.0.1:8080/todos/1
Update a todo	PATCH	/todos/{todo_id}	http PATCH 127.0.0.1:8080/todos/2 title="changed title" completed=true
Delete a todo	DELETE	/todos/{todo_id}	http DELETE 127.0.0.1:8080/todos/3

Note that this is a minimal example. Real applications split their codebase into modules (https://docs.python.org/3/tutorial/modules.html).

Dissecting the example

The code is based on Getting start with aiohttp.web: A todo tutorial (http://justanr.github.io/getting-start-with-aiohttpweb-a-todo-tutorial) which is a step-by-step guide to create a Todo application. Once you're familiar with this simple code, you can switch to a cleaner, view-based and modular version of https://github.com/DurandA/todobackend-aiohttp (todobackend-aiohttp) which shows how you can organize your project.