Lab – Creating Your Own API

For this lab you will be creating a basic API that will support the creation and retrieval of to-do list resources. This lab is very basic and does not cover security, advanced filtering, or some of the other topics we discussed in class today. However, it should give you a good starting point on which to build.

Requirements

1. Create an endpoint at ‘todo/api/v1/tasks’ that retrieves and returns all of the to-do list items stored in the views.py variable ‘tasks’.
   1. The return object should be a JSON object
   2. *(In real life, you would be retrieving these items from a database, but we are only focusing on creating the API endpoint logic)*
2. Create an endpoint at ‘todo/api/v1/tasks/<int:task\_id>’ that searches the ‘tasks’ variable in views.py and returns the task that matches the task id given in the endpoint.
   1. The return object should be a JSON object
   2. Route the user to the provided 404 error handler function if the resource can’t be located (i.e., the given task id does not correspond to any tasks in the ‘tasks’ variable in views.py)
   3. *Hint: You can use the task id given in the endpoint as an argument for the endpoint handler function*
3. Create an endpoint at ‘todo/api/v1/tasks’ that creates a new to-do item and appends it to the ‘tasks’ variable in views.py.
   1. Assume the incoming request data is encoded in JSON
      1. If the incoming POST request data is not in JSON format, route the user to the provided 400 error handler function
      2. *Hint: Use* [*Flask’s built-in request.json method*](http://flask.pocoo.org/docs/0.10/api/#incoming-request-data) *to retrieve the incoming POST request data appropriately*
   2. Create a new to-do item with the following fields
      1. ‘id’ – increment this by 1 more than the last item in the ‘tasks’ variable in views.py
      2. ‘title’ – use the incoming POST request data for the title. If no title is given in the POST request data, route the user to the provided 400 error handler function
      3. ‘description’ – use the incoming POST request data for the description. Default to an empty string if no description provided
      4. ‘done’ – set to False (a newly created task is not done yet, right?)
   3. Return the newly created to-do item as a JSON response to the user
   4. Include the appropriate HTTP [status code](https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html) for a newly created resource [along with your JSON response](http://flask.pocoo.org/docs/0.10/quickstart/#about-responses)
4. Test your API endpoints by running the Flask API server (the one you’ve been working on thus far) as well as running the API Test Server, which is in the apiTest directory of the provided .zip file. This API Test Server represents what might be a third-party application (consumer) of your API, although it’s very basic and is just for testing purposes.
   1. In your browser, navigate to <http://localhost:5001>
   2. Each of the 5 links at the top of the page correspond to a different test. There is one test for each of the 3 API endpoints you’ve created, and 2 tests that correspond to the handlers for getting one task and creating a new task.
   3. Click on each link at the top of the page to test your API – you should see the resulting JSON response data that your API returns.
   4. The tests associated with these links are hard-coded in the views.py file of the testAPI directory. Feel free to play around with or add to these tests as you expand your API.