Minh Nguyen

Assignment 3 Part 2: http://assignment3minh.appspot.com

The URL structure in my API has 2 main routes. One for Client, and one for Candy. Attaching /client or /candy to the URL above will call a get request to display a list of client and a list of candy, respectively. The response is a JSON response. When Posting data, the URL above also uses /client and /candy, depending on which data you'd like to send. Using cURL calls, data will be sent via the URL and routed to a post method whose definition is in the class Client_action or Candy_action, depending on if it's a /client post or a /candy post.

Furthermore, attaching /client/<username> to the URL will create a JSON response for that specific client, if it is created. There is a get method routed to Client_action get method in which it responds to any keyword arguments supplied. Thus calling /client/clientname will result in data for that specific client. The same is designed for /candy/candyname URL.

The PUT method is embedded in a /client/clientname/candyname fashion. This URL is routed to a PUT method in the class Client_Candy. Using cURL, candy is added to clients via the URL order above. See test suite for more clarification.

The delete can be called using cURL with /client/clientname for client deletion, or /candy/candyname for candy. With a delete call using cURL, the URL is routed to a delete method in both the Client_action or Candy_action class depending on the call.

This program is not a truly RESTful api in that resources are not Cacheable. It is indeed stateless, and focused on client-server interaction. I've not made a front-end for this yet, so the client has no UI, but the server has a backend that can communicate just fine using cURL calls. As far as uniform interfaces go, all the data are in JSON format, but the messages are not all that self-descriptive. According to HATEOAS, most of the state is handled via hyperlinks. Although resources do not give information on how to access other resources.

I've changed a few things from my schema in assignment 3 part 1. Client model now does not require a last_name. Quantity model has been remove completely for ease of programming as it was

unnecessary.  Client and Candy model have a named id which is the username and name, respectively. This is to ease the access of each of their keys via strings instead of the random id given by the GAE's data store.

If I were to do this API differently again, I would spend more time developing test cases to handle null inputs.  Empty values right now can be inputted into the server and it will still run fine.  Of course this is a major issue, but with my time constraint I was unable to fix this.