Vend-O-Matic

OVERVIEW

You are responsible for designing a service that will support a beverage vending machine that is tested via HTTP before being placed into a production environment (we believe it's the lobby of a Goodyear Tire franchise - *very* classy).

Please provide detailed instructions so that we can set up and run your project. We run primarily on OSX.

Finally, it is recommended (but not required) that you try to use as few dependencies where possible*. As always, if you have *any* questions, please do not hesitate to ask!

** We don't want you to re-implement HTTP, nor string-concat JSON*

Vending Constraints

1. The machine only accepts US quarters - you physically cannot put anything else in, and you can only put one coin in at a time.
2. Purchase price of an item is two US quarters.
3. Machine only holds five of each of the three beverages available to purchase in its inventory.
4. Machine will accept more than the purchase price of coins, but will only dispense a single beverage per transaction.
5. Upon transaction completion, any unused quarters must be dispensed back to the customer.
6. All test interactions will be performed with a single content type of "application/json".

SPECIFICATIONS

Verb	URI	Request Body	Response Code	Response Headers	Response Body

PUT	/	{
		“coin”: 1
		}
			204	X-Coins: $[# of coins accepted]	

DELETE	/		204	X-Coins: $[# of coins to be returned]	
GET	/inventory		200		Array of remaining item quantities, (an array of integers)
GET	/inventory/:id		200		Remaining item quantity (an integer)

PUT	/inventory/:id		200	X-Coins: $[# of coins to be returned]
				X-Inventory-Remaining: $[item quantity]
					{
					“quantity”: $[number of items vended]
					}

[1]PUT	/inventory/:id		404	X-Coins: $[# of coins accepted]	
[2]PUT	/inventory/:id		403	X-Coins: $[0|1]	

Note: $[*message*] denotes a dynamic value.

1. We return the following header and response code if that item is out of stock.
2. We use this if an attempt to purchase is made, but the number of coins are insufficient.