Coding Assessment:

The goal is to implement a RESTful inventory tracking web service. Clients can request information about the entire inventory, or a single item. Resource identifiers are assigned as follows:

- "/inventory" refers to the entire inventory
- "/inventory/Apples" refers to the item named "Apples" in the inventory

The data accepted and generated by the web service is encoded using JSON.

The web service listens for requests on port 8081. So, for example, a **GET** request to the URL

http://localhost:8081/inventory

should return the initial inventory as a JSON array, which is

```
[
 {
  "name": "Apples",
  "quantity": 3
 },
 {
  "name": "Oranges",
  "quantity": 7
 },
 {
  "name": "Pomegranates",
  "quantity": 55
 }
]
```

The starting point code includes a handler for the HTTP **GET** method which allows the client to access the entire inventory, or a single item. So, for example, the URL

http://localhost:8081/inventory/Apples

should return a single item as a JSON object:

```
{
 "name": "Apples",
 "quantity": 3
}
```

Service Methods to implement:

Handle HTTP **PUT** requests as follows:

- If resource identifier is "/inventory", replaces the entire inventory with the inventory encoded in the JSON document found in the body of the request
- If the resource identifier is "/inventory/*itemname*", replaces the item with the given item name, replacing it with the item that is encoded in the JSON document found in the body of the request

Handle HTTP **POST** requests as follows:

- If the resource identifier is "/inventory", then a single item is ready from the JSON document encoded in the body of the request, and the item is *added* to the inventory.

Handle **DELETE** requests as follows:

- If the resource identifier is "/inventory", all items are deleted from the inventory
- If the resource identifier is "/inventory/*itemname*", the named item is deleted

Note that delete requests have no body.

Testing

You can use the **curl** program to test your implementation. (You can't use a web browser for testing because a web browser cannot generate a **PUT** request.)

Let's say that you have a text file called **replaceApples.txt** with the following JSON data:

```
{
  "name": "Apples",
  "quantity": 16
}
```

You can send this as the body of a **PUT** request using the following **curl** command:

```
curl -X PUT -d @replaceApples.txt http://localhost:8081/inventory/Apples
```

Note that you must run the command from the directory containing **replaceApples.txt**.

Note that using the **-X POST** option will send a **POST** request.