JOHN ABBOTT
CEGEP/COLLEGE

Spring Boot RESTful API

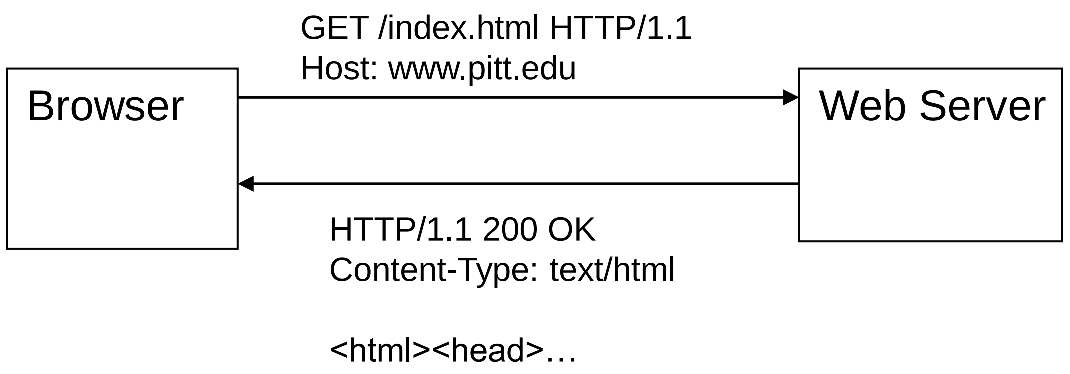
Elie Mambou, Ph.D.

(420-JD5-AB) Programming III

Summer 2022

Hypertext Transfer Protocol (HTTP)

- A communications protocol
- Allows retrieving inter-linked text documents (hypertext)
  - World Wide Web.
- HTTP Verbs
  - HEAD
  - **GET**
  - **POST**
  - PUT
  - DELETE
  - TRACE
  - OPTIONS
  - CONNECT



```
GET /index.html HTTP/1.1
Host: www.pitt.edu
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<html><head>…
```

Representational State Transfer (REST)

- A style of software architecture for distributed hypermedia systems such as the World Wide Web.

- Introduced in the doctoral dissertation of Roy Fielding
  - One of the principal authors of the HTTP specification.

- A collection of network architecture principles which outline how resources are defined and addressed

REST and HTTP

- The motivation for REST was to capture the characteristics of the Web which made the Web successful.
  - URI Addressable resources
  - HTTP Protocol
  - Make a Request – Receive Response – Display Response
- Exploits the use of the HTTP protocol beyond HTTP POST and HTTP GET
  - HTTP PUT, HTTP DELETE

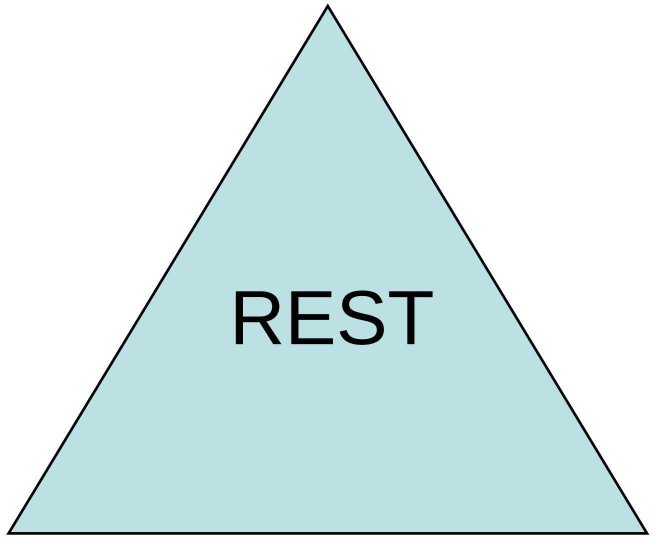
REST - not a Standard

- REST is not a standard
  - JSR 311: JAX-RS: The Java^{TM} API for RESTful Web Services

- But it uses several standards:
  - HTTP
  - URL
  - XML/HTML/GIF/JPEG/etc (Resource Representations)
  - text/xml, text/html, image/gif, image/jpeg, etc (Resource Types, MIME Types)

Main Concepts

Nouns (Resources)
unconstrained
i.e., http://example.com/employees/12345



Verbs
constrained
i.e., GET

Representations
constrained
i.e., XML

Resources

- The key abstraction of information in REST is a resource.
- A resource is a conceptual mapping to a set of entities
  - Any information that can be named can be a resource: a document or image, a temporal service (e.g. "today's weather in Los Angeles"), a collection of other resources, a non-virtual object (e.g. a person), and so on
- Represented with a global identifier (URI in HTTP)
  - http://www.boeing.com/aircraft/747

Naming Resources

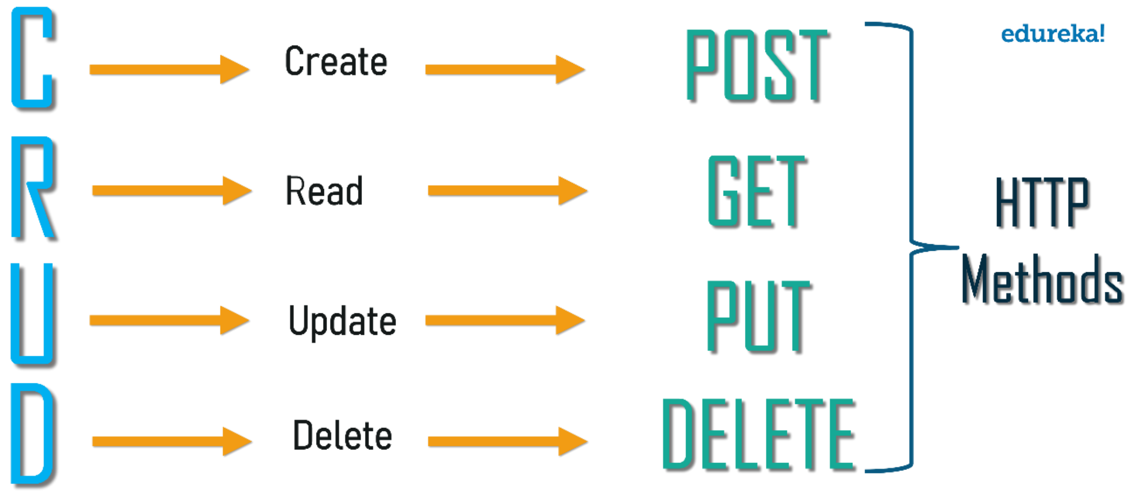
- REST uses URI to identify resources
  - http://localhost/books/
  - http://localhost/books/ISBN-0011
  - http://localhost/books/ISBN-0011/authors

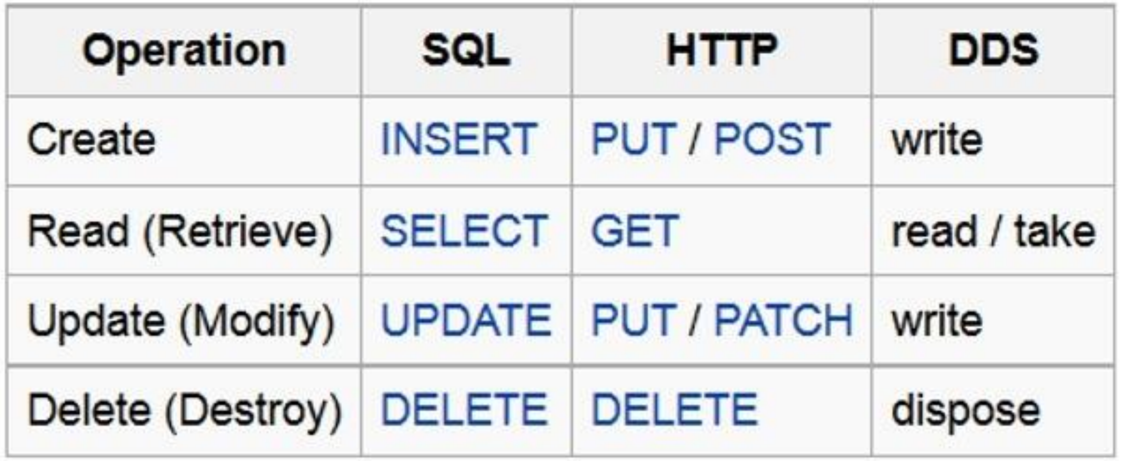
  - http://localhost/classes
  - http://localhost/classes/cs2650
  - http://localhost/classes/cs2650/students
- As you traverse the path from more generic to more specific, you are navigating the data

Verbs

- Represent the actions to be performed on resources

- HTTP GET
- HTTP POST
- HTTP PUT
- HTTP DELETE





Operation	SQL	HTTP	DDS
Create	INSERT	PUT / POST	write
Read (Retrieve)	SELECT	GET	read / take
Update (Modify)	UPDATE	PUT / PATCH	write
Delete (Destroy)	DELETE	DELETE	dispose

HTTP GET

- How clients ask for the information they seek.

- Issuing a GET request transfers the data from the server to the client in some representation

- GET http://localhost/books
  - Retrieve all books

- GET http://localhost/books/ISBN-0011021
  - Retrieve book identified with ISBN-0011021

- GET http://localhost/books/ISBN-0011021/authors
  - Retrieve authors for book identified with ISBN-0011021

HTTP PUT, HTTP POST

- HTTP POST creates a resource
- HTTP PUT updates a resource

- POST http://localhost/books/
  - Content: {title, authors[], …}
  - Creates a new book with given properties

- PUT http://localhost/books/isbn-111
  - Content: {isbn, title, authors[], …}
  - Updates book identified by isbn-111 with submitted properties

HTTP DELETE

- Removes the resource identified by the URI

- DELETE http://localhost/books/ISBN-0011
    - Delete book identified by ISBN-0011

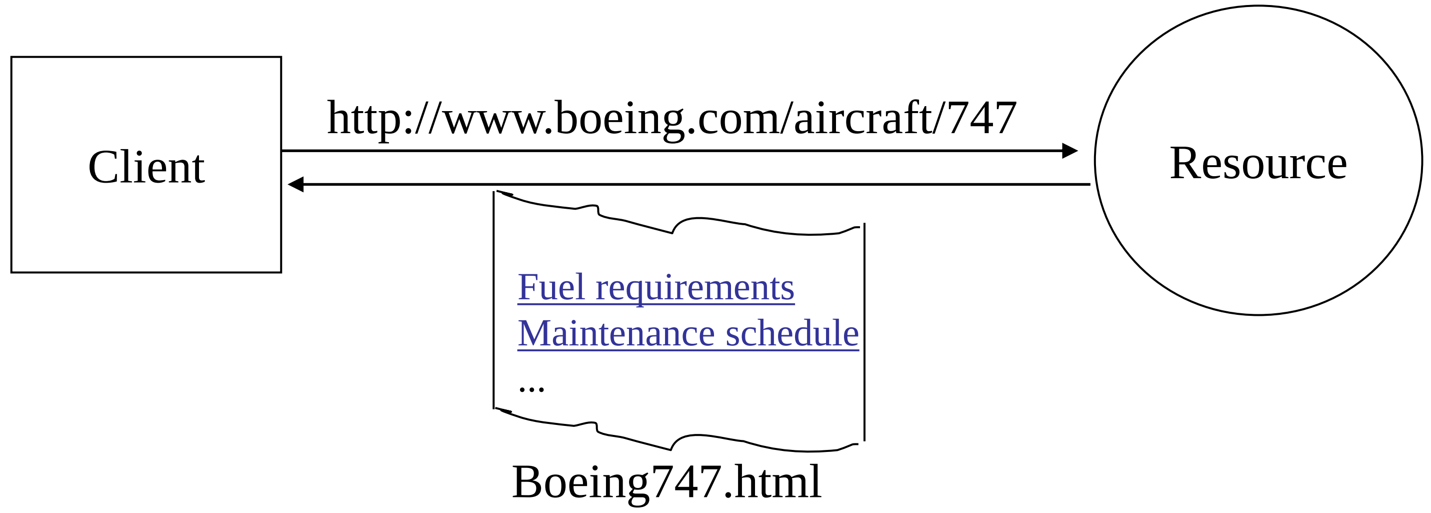
Representations

- How data is represented or returned to the client for presentation.
- Two main formats:
  - JavaScript Object Notation (JSON)
  - XML
- It is common to have multiple representations of the same data

Representations

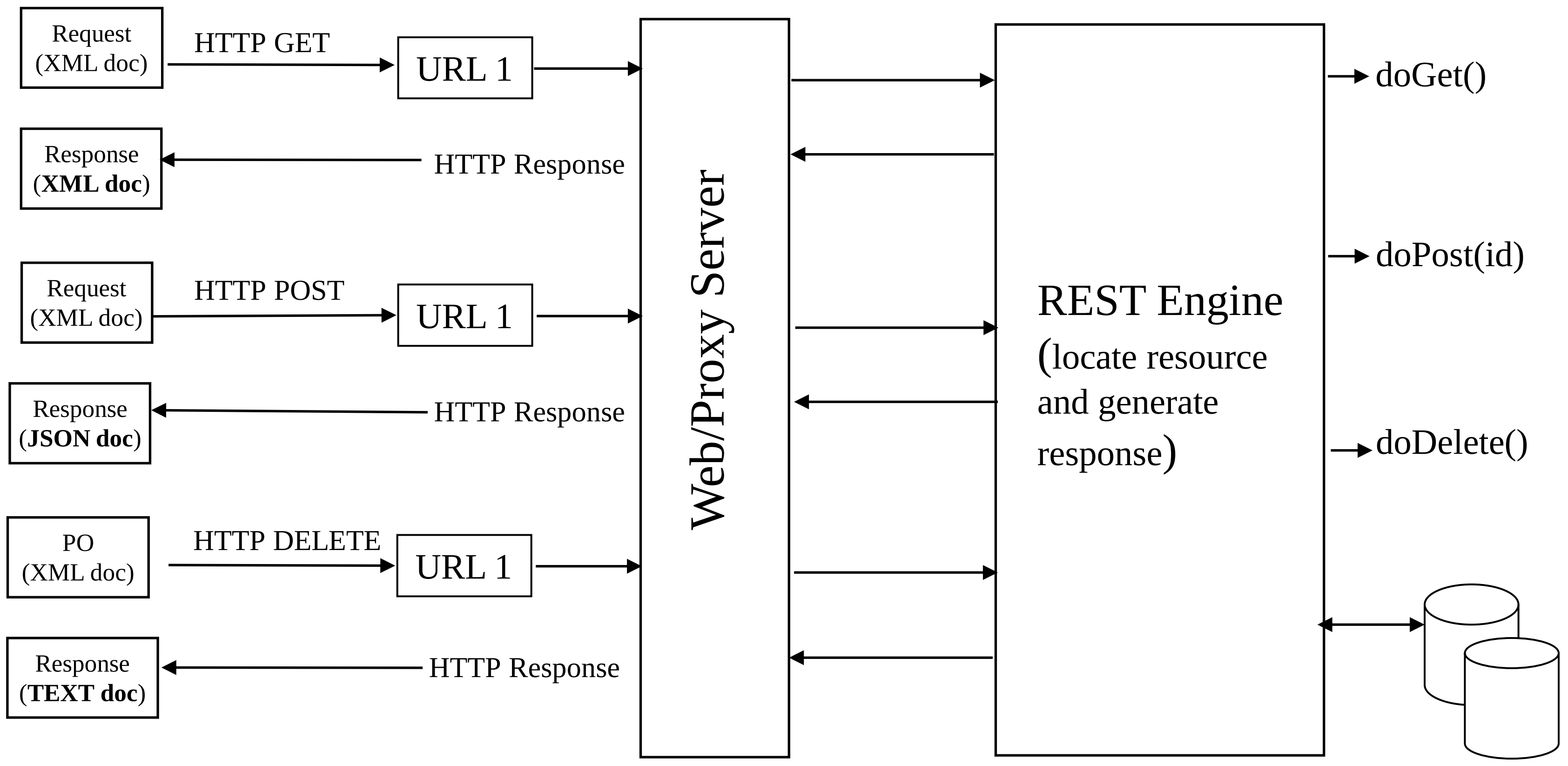
- XML
  - <COURSE>
    - <ID>CS2650</ID>
    - <NAME>Distributed Multimedia Software</NAME>
  - </COURSE>
- JSON
  - {course
    - {id: CS2650}
    - {name: Distributed Multimedia Sofware}
  - }

Why is it called "Representational State Transfer"?



The Client references a Web resource using a URL. A **representation** of the resource is returned (in this case as an HTML document).
The representation (e.g., Boeing747.html) places the client application in a **state**. The result of the client traversing a hyperlink in Boeing747.html is another resource accessed. The new representation places the client application into yet another state. Thus, the client application changes (**transfer**s) state with each resource representation --> Representation State Transfer!

Architecture Style



REST and the Web

- The Web is an example of a REST system!
- All of those Web services that you have been using all these many years - book ordering services, search services, online dictionary services, etc - are REST-based Web services.
- Alas, you have been using REST, building REST services and you didn't even know it.

REST Implementations

- Restlet
  - http://www.restlet.org/
- Project Zero
  - http://www.projectzero.org
- GlassFish Jersey
  - https://jersey.dev.java.net/
- JBoss RESTeasy
  - http://www.jboss.org/resteasy/

HTTP Status Code

•**200 OK:** This code indicates that the request is successful, and the response content is returned to the client as appropriate.

•**201 Created:** This code indicates that the request is successful, and a new resource is created.

•**400 Bad Request:** This code indicates that the server failed to process the request because of the malformed syntax in the request. The client can try again after correcting the request.

•**401 Unauthorized:** This code indicates that authentication is required for the resource. The client can try again with appropriate authentication.

•**403 Forbidden:** This code indicates that the server is refusing to respond to the request even if the request is valid. The reason will be listed in the body content if the request is not a HEAD method.

•**404 Not Found:** This code indicates that the requested resource is not found at the location specified in the request.

•**500 Internal Server Error:** This code indicates a generic error message, and it tells that an unexpected error occurred on the server and that the request cannot be fulfilled.

DEMO OF RESTful API WEB SERVICES

- MAKE A NEW SPRING BOOT PROJECT NAMED **RESTfulAPI**

- ADD THE DEPENDENCY **SPRING WEB**

Product.java class under com.data.model

```java
package com.data.model;

public class Product {
    private String id;
    private String name;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

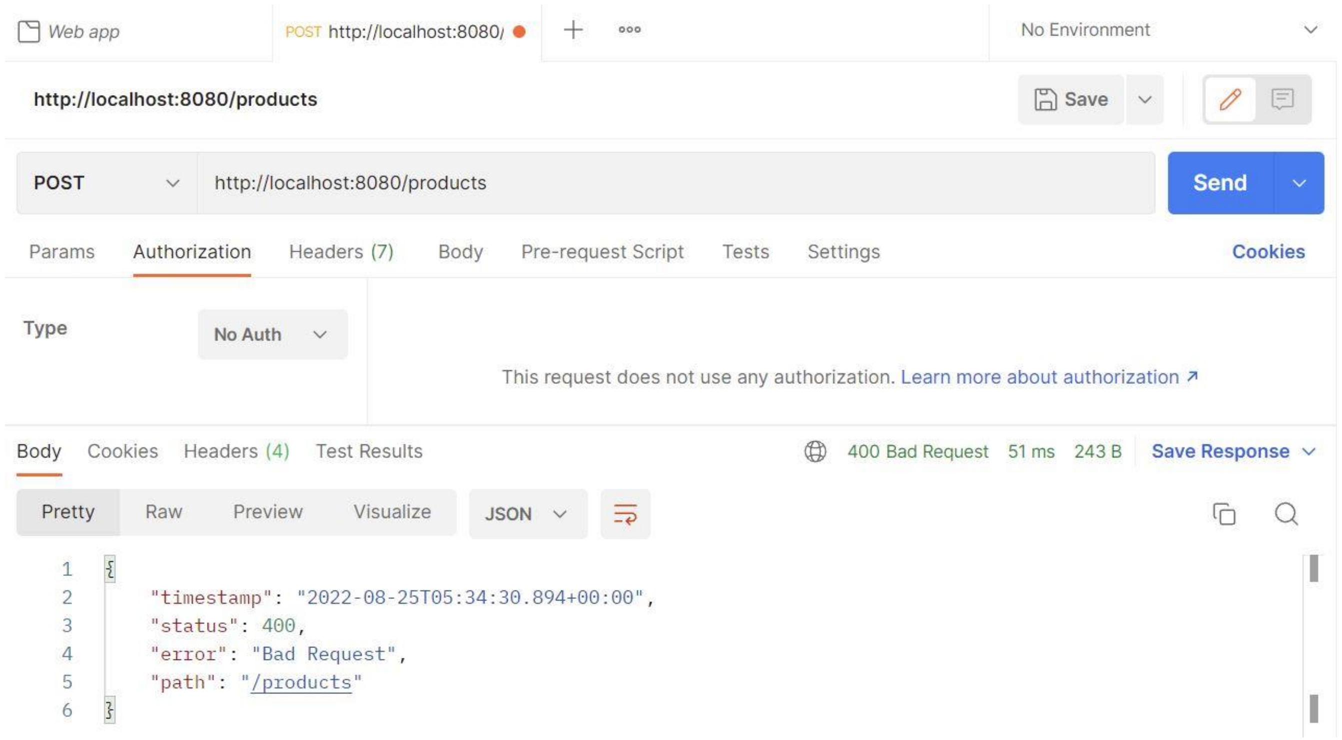
ProductServiceController.java class

```java
package com.data.controller;

import java.util.HashMap;
import java.util.Map;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.data.model.Product;
@RestController
public class ProductServiceController {
   private static Map<String, Product> productRepo = new HashMap<>();
   static {
      Product honey = new Product();
      honey.setId("1");
      honey.setName("Honey");
      productRepo.put(honey.getId(), honey);

      Product almond = new Product();
      almond.setId("2");
      almond.setName("Almond");
      productRepo.put(almond.getId(), almond);
    }
```

```java
@RequestMapping(value = "/products/{id}", method =
RequestMethod.DELETE)
   public ResponseEntity<Object> delete(@PathVariable("id") String id)
{
      productRepo.remove(id);
      return new ResponseEntity<>("Product is deleted successsfully",
HttpStatus.OK);
   }
   @RequestMapping(value = "/products/{id}", method =
RequestMethod.PUT)
   public ResponseEntity<Object> updateProduct(@PathVariable("id")
String id, @RequestBody Product product) {
      productRepo.remove(id);
      product.setId(id);
      productRepo.put(id, product);
      return new ResponseEntity<>("Product is updated successsfully",
HttpStatus.OK);
   }
   @RequestMapping(value = "/products", method = RequestMethod.POST)
   public ResponseEntity<Object> createProduct(@RequestBody Product
product) {
      productRepo.put(product.getId(), product);
      return new ResponseEntity<>("Product is created successfully",
HttpStatus.CREATED);
   }
   @RequestMapping(value = "/products")
   public ResponseEntity<Object> getProduct() {
      return new ResponseEntity<>(productRepo.values(),
HttpStatus.OK);
   }
}
```

COMPILE AND RUN

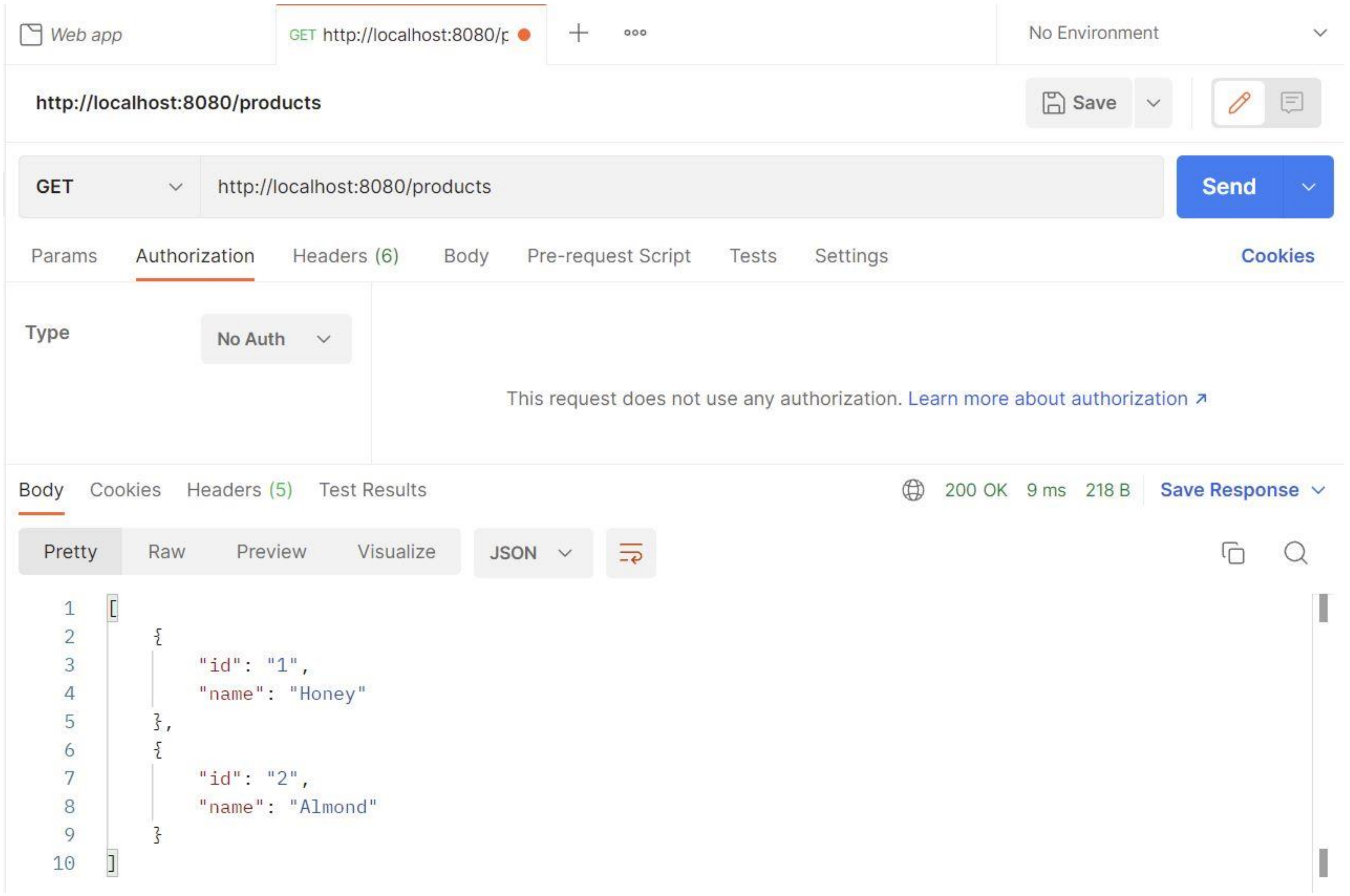
- INSTALL **POSTMAN DESKTOP**
- MAKE A NEW WORKSPACE AND FORK IT TO A REPO
- ERROR 400 WHEN TRYING TO POST NOTHING



This request does not use any authorization. Learn more about authorization ↗

```
{
    "timestamp": "2022-08-25T05:34:30.894+00:00",
    "status": 400,
    "error": "Bad Request",
    "path": "/products"
}
```

GET from http://localhost:8080/products



http://localhost:8080/products

GET http://localhost:8080/products

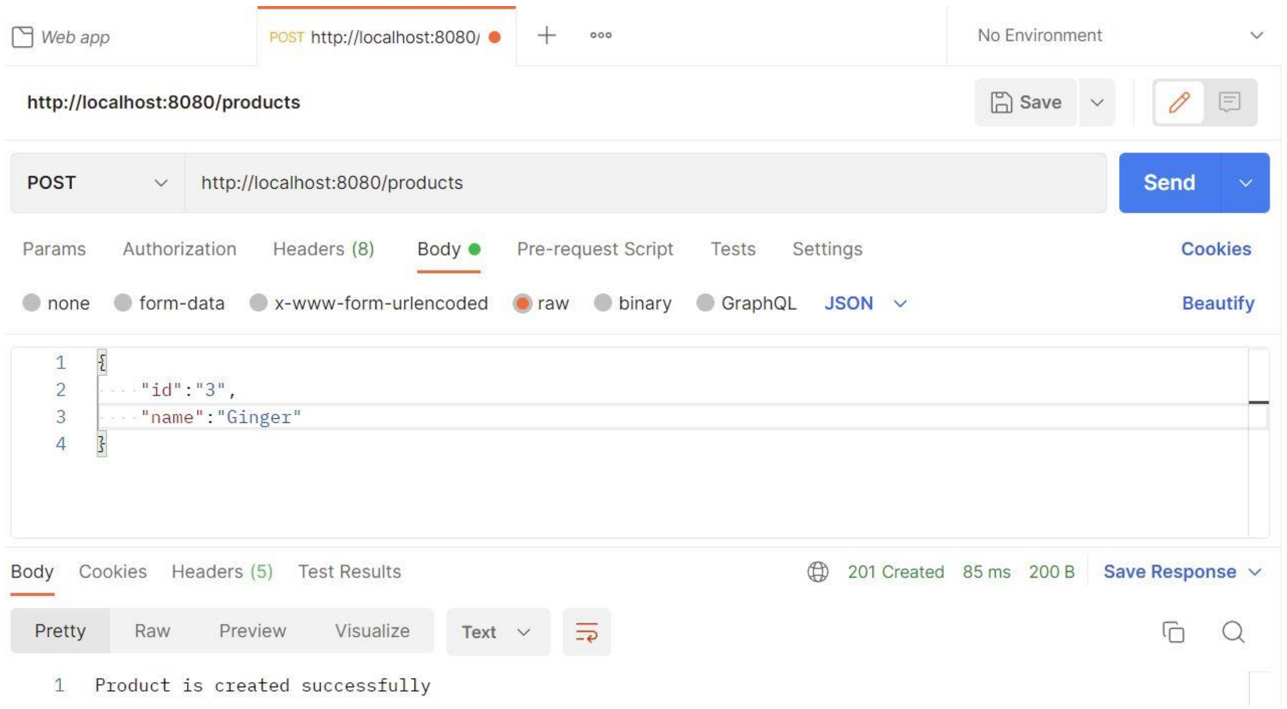
Type: No Auth

This request does not use any authorization. Learn more about authorization ↗

200 OK 9 ms 218 B

```
[
    {
        "id": "1",
        "name": "Honey"
    },
    {
        "id": "2",
        "name": "Almond"
    }
]
```

POST to http://localhost:8080/products



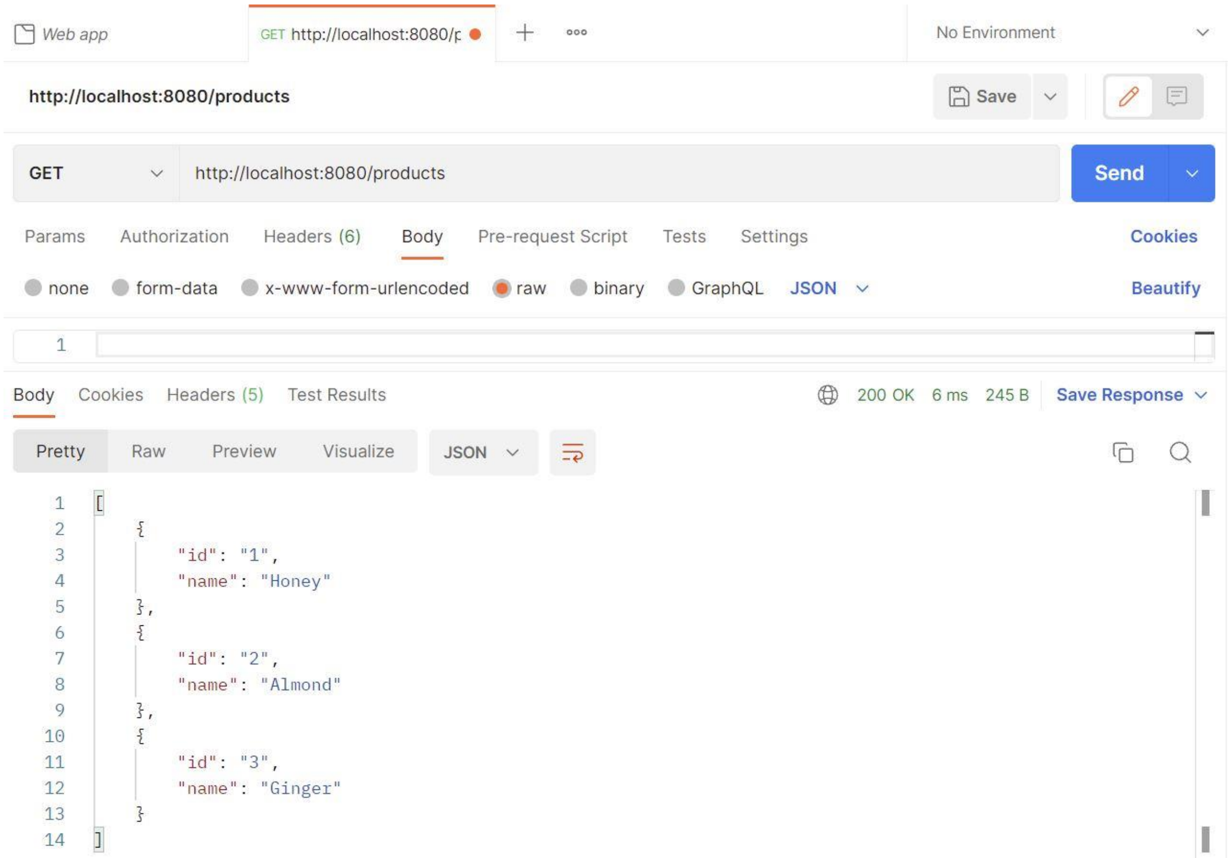
http://localhost:8080/products

POST http://localhost:8080/products

```
{
    "id":"3",
    "name":"Ginger"
}
```

201 Created 85 ms 200 B

```
Product is created successfully
```



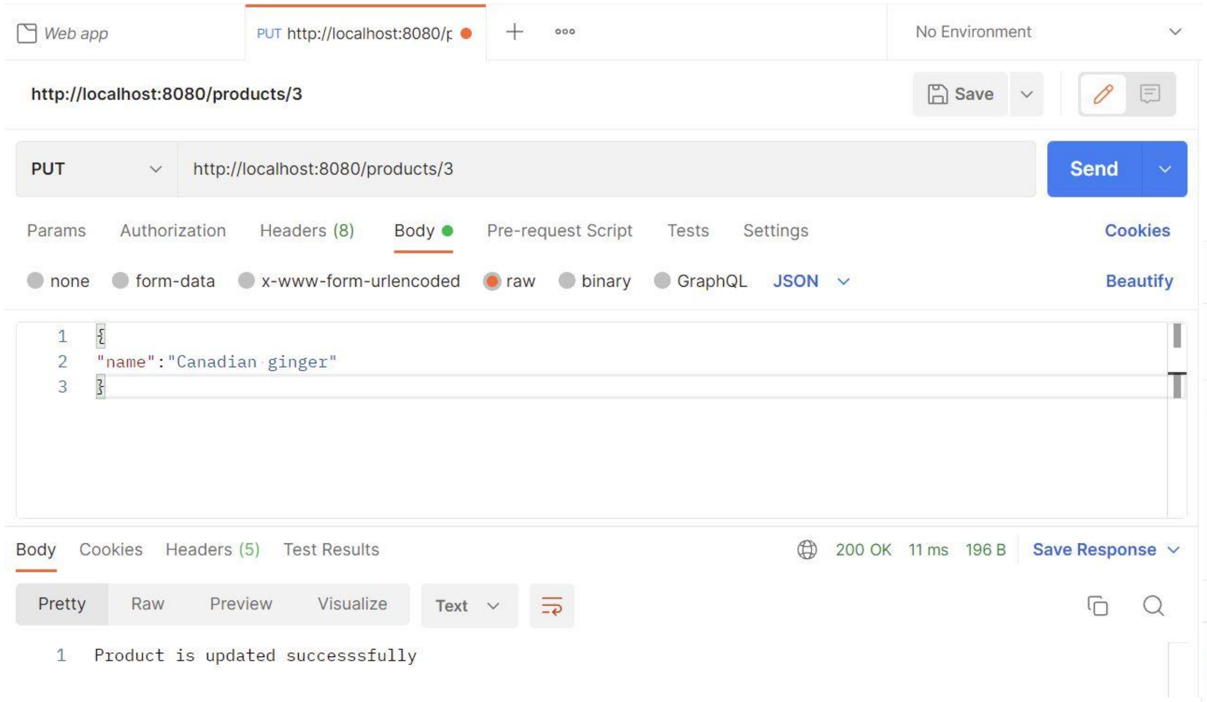
http://localhost:8080/products

GET http://localhost:8080/products

200 OK 6 ms 245 B

```
[
    {
        "id": "1",
        "name": "Honey"
    },
    {
        "id": "2",
        "name": "Almond"
    },
    {
        "id": "3",
        "name": "Ginger"
    }
]
```

PUT(update) at http://localhost:8080/products/3

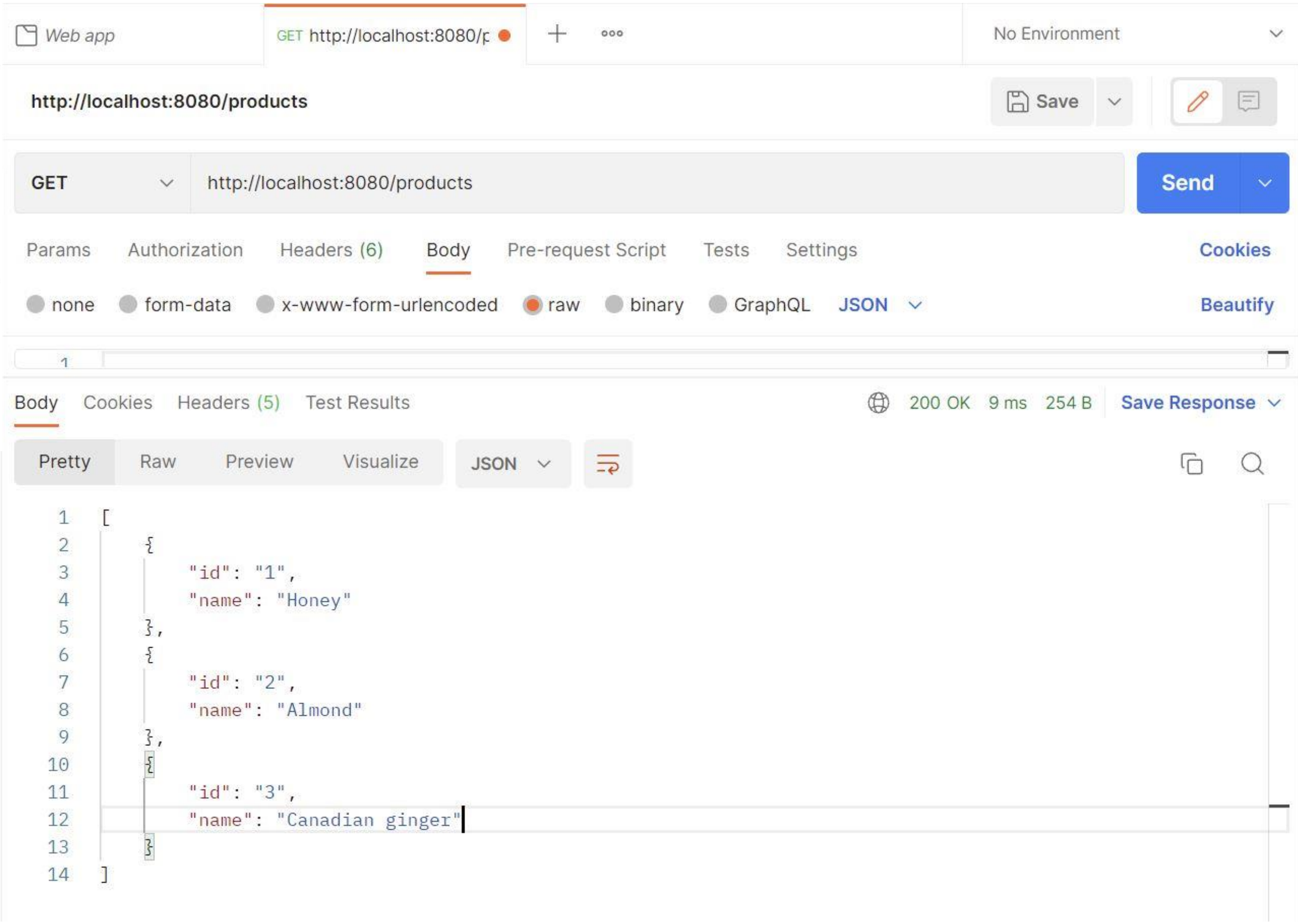


http://localhost:8080/products/3

PUT http://localhost:8080/products/3

```
{
"name":"Canadian ginger"
}
```

```
Product is updated successsfully
```

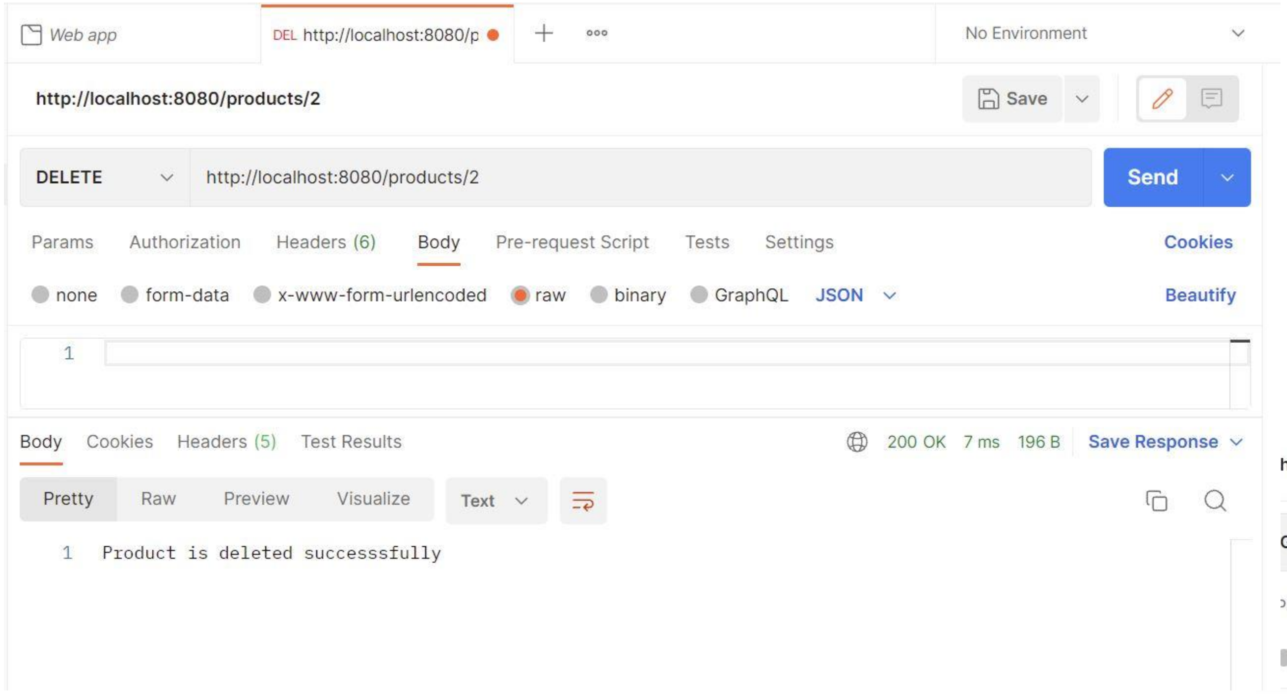


http://localhost:8080/products

GET http://localhost:8080/products

```
[
    {
        "id": "1",
        "name": "Honey"
    },
    {
        "id": "2",
        "name": "Almond"
    },
    {
        "id": "3",
        "name": "Canadian ginger"
    }
]
```

DELETE http://localhost:8080/products/2

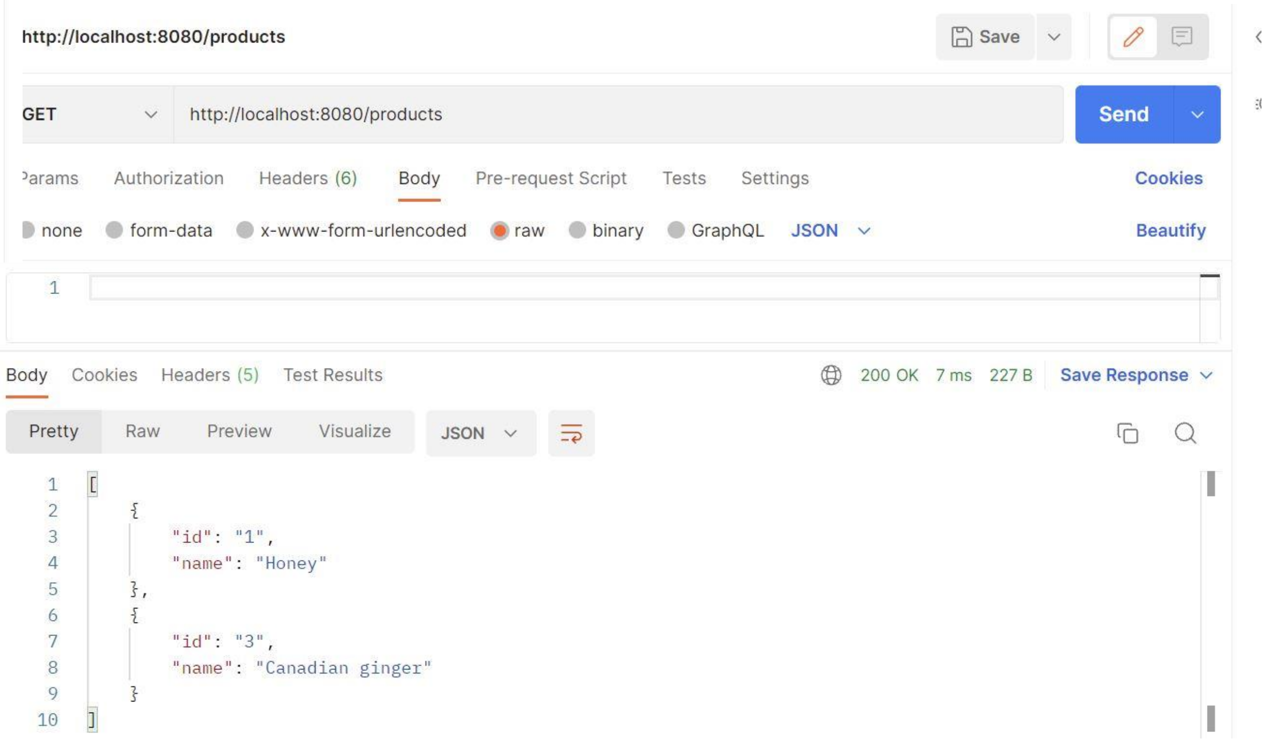


http://localhost:8080/products/2

DELETE http://localhost:8080/products/2

200 OK 7 ms 196 B

```
Product is deleted successsfully
```



http://localhost:8080/products

GET http://localhost:8080/products

200 OK 7 ms 227 B

```
[
    {
        "id": "1",
        "name": "Honey"
    },
    {
        "id": "3",
        "name": "Canadian ginger"
    }
]
```

Q & A

JOHN ABBOTT
CEGEP/COLLEGE