

Web API Design with SpringBoot Week 2 Coding Assignment

Points possible: 70

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

Follow the tutorial in the homework section to build an API. Paste screenshots of your code and your postman requests and responses to show the API works. Push your project to GitHub and paste the link below.



Screenshots of Code:


(code uploaded to github)








Screenshots of Running Application:

1) POST CUSTOMER

http://localhost:8080/customers



POST  http://localhost:8080/customers 

Params Authorization Headers (8) **Body**  Pre-request Script Tests Settings

 none  form-data  x-www-form-urlencoded  **raw**  binary  GraphQL **JSON** 

```
1 {  
2   ... "firstName": "Joebob",  
3   ... "lastName": "Johnson",  
4   ... "level": "GOLD"
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize **JSON**  

```
1 {  
2   "id": 1,  
3   "firstName": "Joebob",  
4   "lastName": "Johnson",  
5   "address": null,  
6   "level": "GOLD",  
7   "orders": null  
8 }
```

2) POST CUSTOMER 2

Overview

POST http://localhost:8080/customers ●

GET http://localhost:8080/customers ●

+

...

http://localhost:8080/customers

POST ●

▼

http://localhost:8080/customers ●

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

```
1 {
2   ... "firstName": "Adam",
3   ... "lastName": "Adder",
4   ... "level": "PLATINUM",
5   ... "address": {
6     ... "street": "123 Main Street",
7     ... "city": "Kalamazoo",
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

↺

```
1 {
2   "id": 2,
3   "firstName": "Adam",
4   "lastName": "Adder",
5   "address": {
6     "id": 1,
```

3) PUT CUSTOMER 1

http://localhost:8080/customers/1

PUT

http://localhost:8080/customers/1

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

"firstName": "Gene",

"lastName": "Hackman",

"level": "GOLD",

"address": {

"street": "123 Main Street",

"city": "Kalamazoo",

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

"id": 1,

"firstName": "Gene",

"lastName": "Hackman",

"address": {

"id": 2,

"street": "123 Main Street"

4) GET CUSTOMER 2

<http://localhost:8080/customers/2>

The screenshot shows the REST Client interface with a GET request to `http://localhost:8080/customers/2`. The request body is a JSON object: `{ "firstName": "Gene", "lastName": "Hackman", "level": "GOLD", "address": { "street": "123 Main Street", "city": "Kalamazoo" } }`. The response body is a JSON array: `[{ "id": 2, "firstName": "Adam", "lastName": "Adder", "address": { "id": 1, "street": "123 Main Street", "city": "Kalamazoo" } }, { "id": 1, "firstName": "Gene", "lastName": "Hackman", "level": "GOLD", "address": { "id": 2, "street": "123 Main Street", "city": "Kalamazoo" } }]`.

5) DELETE CUSTOMER 1

http://localhost:8080/customers/1

DELETE ● ▼

http://localhost:8080/customers/1 ●

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

1

2

3

4

5

6

7

→

→

→

→

→

→

→

"firstName": "Gene",

"lastName": "Hackman",

"level": "GOLD",

"address": {

→ "street": "123 Main Street",

→ "city": "Kalamazoo",

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

Text ▼

≡ ↺

1

Successfully deleted customer with id: 1 ●

6) POST PRODUCTS

http://localhost:8080/products

POST

http://localhost:8080/products

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   "name": "Widget",
3   "description": "A widget",
4   "price": 19.95
5 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 1,
3   "name": "Widget",
4   "description": "A widget",
5   "price": 19.95
6 }
```

7) POST PRODUCT 2

POST

http://localhost:8080/products

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

```
1 {
2   "name": "Doohickey",
3   "description": "A doohickey",
4   "price": 39.95
5 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw


Preview


Visualize

JSON

```
1 {
2   "id": 2,
3   "name": "Doohickey",
4   "description": "A doohickey",
5   "price": 39.95
6 }
```


8) POST PRODUCT 3


POST 

http://localhost:8080/products 

Params

Authorization

Headers (8)

Body 

Pre-request Script

Test

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ Gr

```
1 {  
2   "name": "Whatchamacallit",  
3   "description": "A whachamacallit",  
4   "price": 79.95  
5 }
```

Body

Cookies

Headers (4)


Test Results


Pretty

Raw

Preview

Visualize

JSON 



```
1 {  
2   "id": 3,  
3   "name": "Whatchamacallit",  
4   "description": "A whachamacallit",  
5   "price": 79.95  
6 }
```

9) GET ALL PRODUCTS

GET

http://localhost:8080/products

ParamsAuthorizationHeaders (8)BodyPre-request Script


BodyCookiesHeaders (4)Test Results

PrettyRawPreviewVisualize


JSON

```
1  [
2    {
3      "id": 1,
4      "name": "Widget",
5      "description": "A widget",
6      "price": 19.95
7    },
8    {
9      "id": 2,
10     "name": "Doohickey",
11     "description": "A doohickey",
12     "price": 39.95
13   },
14   {
15     "id": 3,
16     "name": "Whatchamacallit",
17     "description": "A whatchamacallit"
```

10) PUT PRODUCT 2

PUT 


▼

http://localhost:8080/products/2 

Params


Authorization


Headers (8)


Body 


Pre-request Script


Tests


 none

 form-data

 x-www-form-urlencoded

 raw

 binary

 GraphQL

```
1  {}
2  →  "name": "Doohickey",
3  →  "description": "A doohickey",
4  →  "price": 34.96
5  }
```

Body

Cookies

Headers (4)

Test Results


Pretty

Raw

Preview

Visualize

JSON ▼



```
1  {}
2  →  "id": 2,
3  →  "name": "Doohickey",
4  →  "description": "A doohickey",
5  →  "price": 34.96
6  }
```

11) DELETE PRODUCT 2

The screenshot shows a REST client interface with the following components:

- Method and URL:** DELETE http://localhost:8080/products/3
- Tabs:** Params, Authorization, Headers (8), Body , Pre-request Script
- Body Type:** ☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary
- Body Content:**

```
1 {  
2   "name": "Doohickey",  
3   "description": "A doohickey",  
4   "price": 34.96  
5 }
```
- Response Tabs:** Body, Cookies, Headers (4), Test Results
- Response Format:** Pretty, Raw, Preview, Visualize, Text
- Response Content:**

```
1 successfully deleted product with id: 3 
```

(I went back and added the “S”)

12) POST ORDER

POST

http://localhost:8080/customers/2/orders

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

1

```
[{"id": 1, "year": 2021, "month": "MARCH", "era": "CE"}]
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

```
{
```

2

```
  "id": 1,
```

3

```
  "ordered": {
```

4

```
    "year": 2021,
```

5

```
    "month": "MARCH",
```

6

```
    "era": "CE",
```

13) POST ORDER 2

POST

http://localhost:8080/customers/2/orders

Params

Authorization

Headers (8)

Body

Pre-request Script

Test Results

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

1

[{"id": 2, "year": 2021, "month": "MARCH"}]

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"id": 2,

3

"ordered": {

4

"year": 2021,

5

"month": "MARCH",

6

"day": "05"

14) PUT ORDER CANCELLED

PUT

http://localhost:8080/customers/2/orders/1

Params

Authorization

Headers (8)

Body

Pre-request Script

T

none

form-data

x-www-form-urlencoded

raw

binary

```
1 {
2   "status": "CANCELLED"
3 }
```

Body

Cookies

Headers (4)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
30 {,
31   "delivered": null,
32   "invoiceAmount": 15.959999999999999,
33   "status": "CANCELLED",
34   "products": [
35     {
36       "id": 1,
```

15) PUT ORDER DELIVERED

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/customers/2/orders/2
- Body:**

```
{
  "status": "DELIVERED"
}
```
- Response:**

```
{
  "invoiceAmount": 15.959999999999999,
  "status": "DELIVERED",
  "products": [
    {

```

URL to GitHub Repository:

<https://github.com/wlindstrom55/SpringBoot-wk-2-assignments>