

G2G Backend Task

Description

Spring Boot project with postgresSQL. It is an api for fetching, sorting, filtering, modifying and deleting items. It also has a pagination feature.

The api is tested using Postman and the screenshots are attached.

GET api

1. localhost:8080/host/v1/items – lists all items

GET localhost:8080/host/v1/items

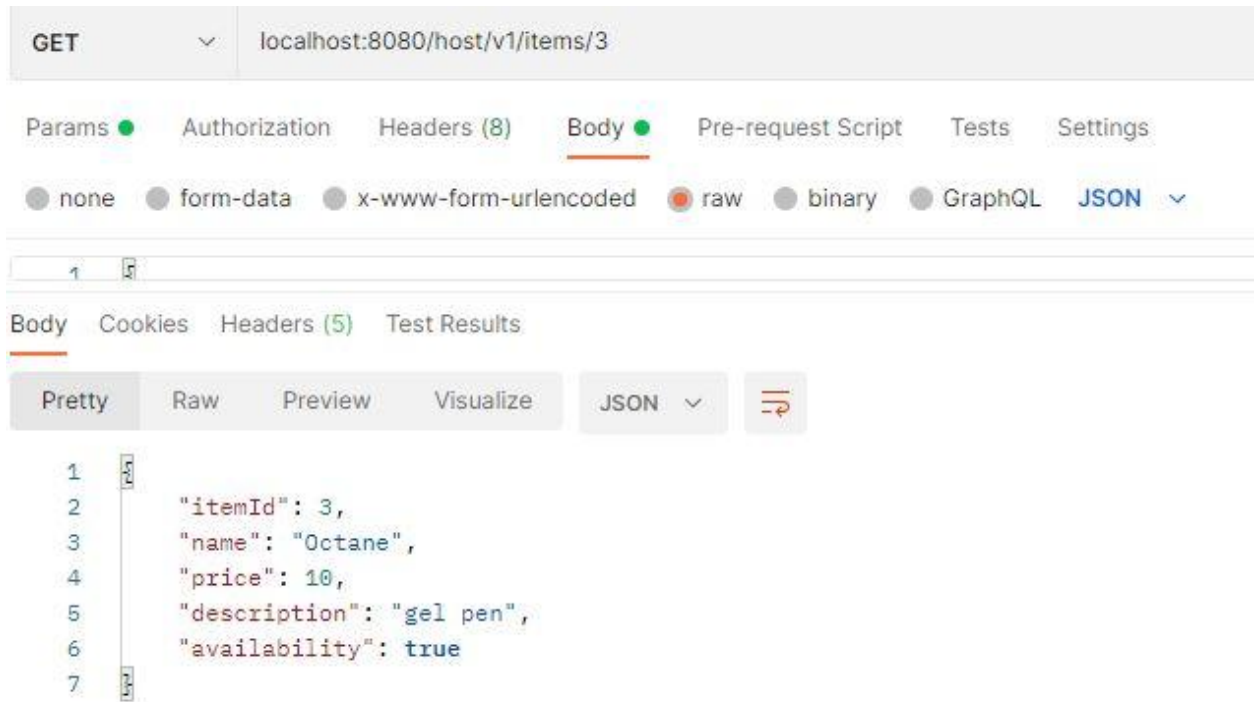
Params Authorization Headers (8) Body Pre-request Script Tests Set

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
3      "itemId": 2,  
4      "name": "Pentonic",  
5      "price": 15,  
6      "description": "ball pen",  
7      "availability": true  
8    },  
9    {  
10     "itemId": 4,  
11     "name": "Oreo",  
12     "price": 25,  
13     "description": "cream biscuit",  
14     "availability": true  
15   },  
16   {  
17     "itemId": 5,  
18     "name": "Fevicol",  
19     "price": 15,  
20     "description": "adhesive",  
21     "availability": true  
22   },  
23   {  
24     "itemId": 6,  
25     "name": "Closeup",  
26     "price": 10,
```

2. localhost:8080/host/v1/items/id/{id} – gets item by id



3. localhost:8080/host/v1/items?limit = x & offset = y – pagination with limit x and offset y

Param : limit field and offset field (Integer)

GET localhost:8080/host/v1/items?limit=2&offset=1

Params ● Authorization Headers (8) Body ● Pre-request Script

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1  [
2    {
3      "itemId": 3,
4      "name": "Octane",
5      "price": 10,
6      "description": "gel pen",
7      "availability": true
8    },
9    {
10     "itemId": 4,
11     "name": "Oreo",
12     "price": 25,
13     "description": "cream biscuit",
14     "availability": true
15   }
16 ]
```

4. localhost:8080/host/v1/items?sortBy = x & sortByDirection = y — sorts field x in direction y (ASC or DESC)

Param : sortBy field and sortByDirection field (strings, sortByDirection can only take ASC or DESC values)

GET localhost:8080/host/v1/items?sortBy=price&sortDirection=DESC

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

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1  [
2    {
3      "itemId": 1,
4      "name": "Bourbon biscuit",
5      "price": 30,
6      "description": "chocolaty biscuit",
7      "availability": false
8    },
9    {
10     "itemId": 4,
11     "name": "Oreo",
12     "price": 25,
13     "description": "cream biscuit",
14     "availability": true
15   },
16   {
17     "itemId": 2,
18     "name": "Pentonic",
19     "price": 15,
20     "description": "ball pen",
21     "availability": true
22   },
23   {
24     "itemId": 5,
```

5. localhost:8080/host/v1/items?nameLike = x — filter name entries by x

Param : nameLike (string)

GET localhost:8080/host/v1/items?nameLike=Oreo

Params ● Authorization Headers (8) Body ● Pre-request Script

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 [
2   {
3     "itemId": 4,
4     "name": "Oreo",
5     "price": 25,
6     "description": "cream biscuit",
7     "availability": true
8   }
9 ]
```

6. localhost:8080/host/v1/items?descriptionLike = x — filter description entries by x

Param : descriptionLike (string)

GET localhost:8080/host/v1/items?descriptionLike=toothpaste

Params ● Authorization Headers (8) Body ● Pre-request Script

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▾

```
1 [
2   {
3     "itemId": 6,
4     "name": "Closeup",
5     "price": 10,
6     "description": "toothpaste",
7     "availability": true
8   }
9 ]
```

POST api

1. localhost:8080/host/v1/items — posts a new payload

Param : payload

Payload format :

```
{  
    "name" : String,  
    "description" : String,  
    "price": long,  
    "availability" : boolean  
}
```

POST localhost:8080/host/v1/items

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

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

```
1 {
2   ... "name": "Bourbon biscuit",
3   ... "description": "chocolaty biscuit",
4   ... "price": 30.0,
5   ... "availability": false
6 }
7
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "itemId": 1,
3   "name": "Bourbon biscuit",
4   "price": 30,
5   "description": "chocolaty biscuit",
6   "availability": false
7 }
```

PATCH api

1. localhost:8080/host/v1/items/id/{id} — modifies specific field according to id

Param : field to modify (String - name, price, description; long - price; availability - boolean)

PATCH localhost:8080/host/v1/items/id/1

Params Authorization Headers (8) Body Pre-request S

none form-data x-www-form-urlencoded raw bin

```
1 {
2   "price": 25.0
3 }
4
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "itemId": 1,
3   "name": "Bourbon biscuit",
4   "price": 25,
5   "description": "chocolaty biscuit",
6   "availability": false
7 }
```

Delete api

1. localhost:8080/host/v1/items/id/{id} — deletes entry according to id

DELETE localhost:8080/host/v1/items/id/3

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

Body Cookies Headers (4) Test Results Status: 200 OK

Pretty Raw Preview Visualize Text

```
1
```

Features

1. All apis work and return 200 OK for all responses.
2. Data persists in DB using JPA.
3. Meaningfully named variables