

## Project Setup and API Documentation

Maven Dependencies that were used.

```
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.36</version>
    <scope>provided</scope>
  </dependency>
</dependencies>
```

```
59  <dependency>
60    <groupId>org.springframework.boot</groupId>
61    <artifactId>spring-boot-starter-validation</artifactId>
62  </dependency>
63 </dependencies>
64
65 <build>
66   <plugins>
67     <plugin>
68       <groupId>org.springframework.boot</groupId>
69       <artifactId>spring-boot-maven-plugin</artifactId>
70     </plugin>
71   </plugins>
72 </build>
73
74 </project>
```

MySQL Workbench started on localhost:3306

Created database with name inventory management

```
689 -> ;
691 +-----+
772 | Database |
866 +-----+
922 | im       |
409 | information_schema |
568 | inventorymanagement |
L [0 | mysql    |
und | performance_schema |
: 8. | sys      |
und +-----+
und 6 rows in set (0.03 sec)
e: Undefined/unknown
e: undefined/unknown

mysql> |
```

Application.properties Configuration

```
inventorymanagementsystemApplication.java pom.xml (inventory-management-system)
1 spring.application.name=Inventory-Management-System
2 spring.datasource.url=jdbc:mysql://localhost:3306/inventorymanagement
3 spring.datasource.username=root123
4 spring.datasource.password=root123
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
Run InventoryManagementSystemApplication x
2025-03-16T20:35:01.866-04:00 INFO 19940 --- [Inventory-Management-System] [main] org.hibernate.Version : HHH0000412: Hibernate ORM core v
2025-03-16T20:35:01.922-04:00 INFO 19940 --- [Inventory-Management-System] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache d
2025-03-16T20:35:02.409-04:00 INFO 19940 --- [Inventory-Management-System] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignor
2025-03-16T20:35:02.568-04:00 INFO 19940 --- [Inventory-Management-System] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0.41
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-03-16T20:35:02.994-04:00 INFO 19940 --- [Inventory-Management-System] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000489: No JTA platform avai
2025-03-16T20:35:03.000-04:00 INFO 19940 --- [Inventory-Management-System] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFa
2025-03-16T20:35:03.057-04:00 WARN 19940 --- [Inventory-Management-System] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enab
2025-03-16T20:35:03.624-04:00 INFO 19940 --- [Inventory-Management-System] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (ht
Inventory-Management-System > src > main > resources > @ application.properties 5:61 LF ISO-8859-1 4
```

## Controller Class mappings

```
@RestController no usages
@RequestMapping("/items")
@Validated
public class ItemController {

    @Autowired 7 usages
    public ItemService itemService;

    @GetMapping("/getItem/{id}") no usages
    public Optional<Item> getItemById(@PathVariable int id) {
        return itemService.getItembyID(id);
    }

    @PostMapping("/addItem") no usages
    public Item saveitem(@Valid @RequestBody Item item) {
        return itemService.saveItem(item);
    }

    @PutMapping("/updateItemQuantity") no usages
    public Item updateitem(@Valid @RequestBody Item item) {
        return itemService.updateItemQuantity(item);
    }
}
```

```

@PutMapping("/updateItemById/{id}") no usages
public Item updateWholeItem(@PathVariable int id, @Valid @RequestBody Item item) {
    return itemService.updateItemEntity(id,item);
}

@DeleteMapping("/deleteById/{id}") no usages
public void deleteItembyid(@PathVariable int id) {
    itemService.deleteItem(id);
}

@DeleteMapping("/deleteAllItems") no usages
public void deleteallitems() {
    itemService.deleteAll();
}

```

## API Endpoints

### POST url /addItem

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

- URL:** `http://localhost:8080/items/addItem`
- Method:** `POST`
- Body (JSON):**

```

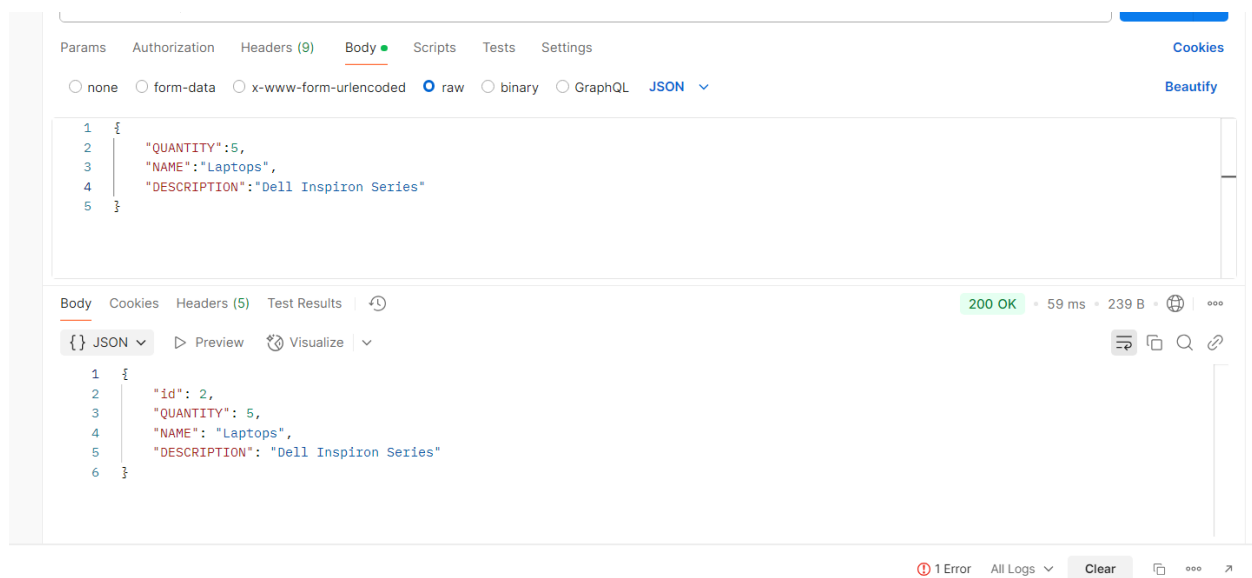
1 {
2   "QUANTITY":2,
3   "NAME":"tomatoes",
4   "DESCRIPTION":"Fresh Delicious Tomatoes"
5 }

```
- Response:** `200 OK` (421 ms, 244 B)
- Response Body (JSON):**

```

1 {
2   "id": 1,
3   "QUANTITY": 2,
4   "NAME": "tomatoes",
5   "DESCRIPTION": "Fresh Delicious Tomatoes"
6 }

```
- Console:** Shows the response type as `application/json` and the status as `chunked`. The timestamp is `025 04:25:28 GMT+`.
- Postbot:** A button labeled "Postbot" is visible at the bottom right.



Data Loaded in MYSQL From POST call

```
mysql> select * from item
-> ;
+-----+-----+-----+-----+
| id | quantity | description          | name    |
+-----+-----+-----+-----+
| 1 | 2 | Fresh Delicious Tomatoes | tomatoes |
| 2 | 5 | Dell Inspiron Series    | Laptops  |
+-----+-----+-----+-----+
2 rows in set (0.01 sec)

mysql>
```

## GET API endpoint – to get all the items /getItems

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

- URL:** `http://localhost:8080/items/getItems`
- Method:** `GET`
- Body:** A JSON object representing a single item:

```
1 {
2   "QUANTITY": 5,
3   "NAME": "Laptops",
4   "DESCRIPTION": "Dell Inspiron Series"
5 }
```
- Response:** A `200 OK` status with a response time of `334 ms` and a size of `322 B`. The response body is a JSON array of two items:

```
1 [
2   {
3     "id": 1,
4     "QUANTITY": 2,
5     "NAME": "tomatoes",
6     "DESCRIPTION": "Fresh Delicious Tomatoes"
7   },
8   {
9     "id": 2,
10    "QUANTITY": 5,
11    "NAME": "Laptops",
12    "DESCRIPTION": "Dell Inspiron Series"
13  }
14 ]
```

## DELETE API endpoint – Delete by ID

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

- URL:** `http://localhost:8080/items/deleteById/1`
- Method:** `DELETE`
- Body:** A JSON object representing a single item (identical to the one in the first screenshot):

```
1 {
2   "QUANTITY": 5,
3   "NAME": "Laptops",
4   "DESCRIPTION": "Dell Inspiron Series"
5 }
```
- Response:** A `200 OK` status with a response time of `156 ms` and a size of `123 B`. The response body is empty, as indicated by the `Raw` view showing only a newline character.

PUT API endpoint – update only the Item Quantity here ... /updateItemQuantity

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

- URL:** `http://localhost:8080/items/updateItemQuantity`
- Method:** `PUT`
- Body (raw JSON):**

```
1 {
2   "id": 2,
3   "QUANTITY": 15,
4   "NAME": "Laptops1",
5   "DESCRIPTION": "Dell Inspiron Series"
6 }
```
- Response:** `200 OK` (93 ms, 240 B)
- Response Body (JSON):**

```
1 {
2   "id": 2,
3   "QUANTITY": 15,
4   "NAME": "Laptops",
5   "DESCRIPTION": "Dell Inspiron Series"
6 }
```

Response Returned, only the quantity is updated

This screenshot is identical to the one above, showing the same PUT request and response. The response body shows that the quantity has been updated from 10 to 15, while the name and description remain unchanged.

Postbot

## MYSQL Update data

```
mysql> select * from item;
+----+-----+-----+-----+
| id | quantity | description          | name    |
+----+-----+-----+-----+
| 2  |      15 | Dell Inspiron Series | Laptops |
+----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

PUT mapping Update, update the entire Record or data - /updateItemById/{id}

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

- URL:** `http://localhost:8080/items/updateItemById/2`
- Method:** PUT
- Body (JSON):**

```
{
  "id": 2,
  "QUANTITY": 15,
  "NAME": "Laptops",
  "DESCRIPTION": "HP Gaming Series"
}
```
- Response:** 200 OK, 31 ms, 240 B
- Response Body (JSON):**

```
{
  "id": 2,
  "QUANTITY": 15,
  "NAME": "Laptops",
  "DESCRIPTION": "Dell Inspiron Series"
}
```

At the bottom of the interface, there is a button labeled "Do that".



HTTP `http://localhost:8080/items/udateltemById/2`

PUT `http://localhost:8080/items/udateltemById/2`

Params Authorization Headers (9) **Body** Scripts Tests Settings

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

```
1 {
2   "id":2,
3   "QUANTITY":15,
4   "NAME":"Laptops",
5   "DESCRIPTION":"HP Gaming Series"
6 }
```

Body Cookies Headers (5) Test Results ↻

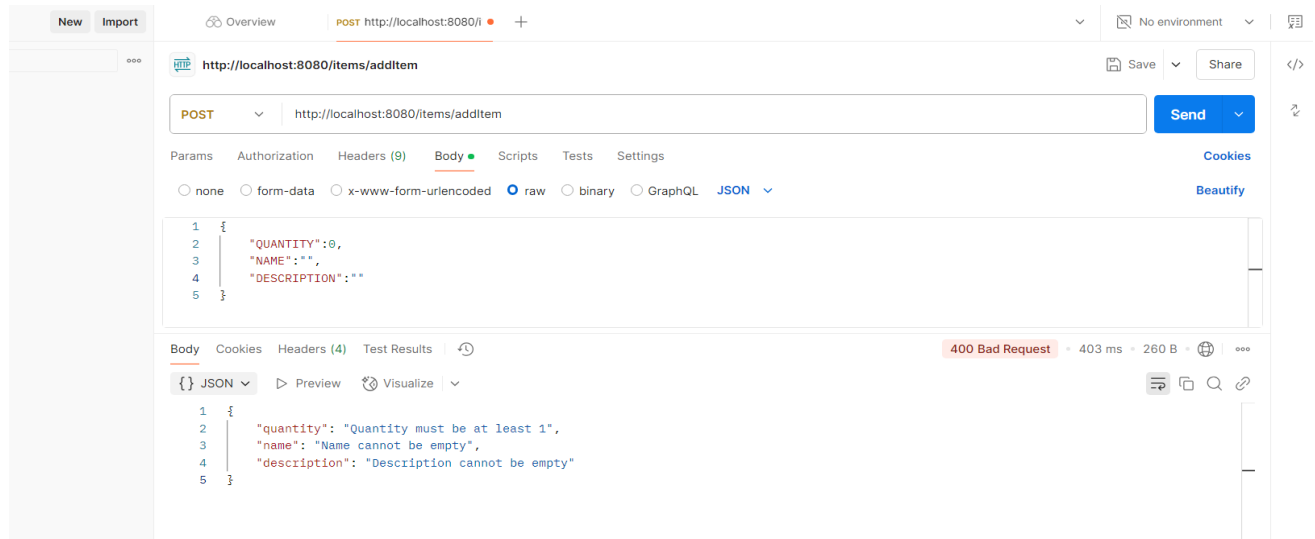
{ JSON ▾ ▶ Preview 📄 Visualize ▾

```
1 {
2   "id": 2,
3   "QUANTITY": 15,
4   "NAME": "Laptops",
5   "DESCRIPTION": "HP Gaming Series"
6 }
```

```
mysql> select * from item;
+----+-----+-----+-----+
| id | quantity | description      | name   |
+----+-----+-----+-----+
| 2  |      15 | HP Gaming Series | Laptops |
+----+-----+-----+-----+
1 row in set (0.01 sec)

mysql>
```

## Validating the Errors for Empty POST request



Number of characters can be at most 50

