/v3/api-docs Explore

Inventory Management API

3.0.0 OAS3

/v3/api-docs

API documentation for Inventory Management System

Servers

http://localhost:8080 - Generated server url

Items Operations related to inventory items

PUT `/items/sell/{id}` Sell an item

This API allows you to sell an item by reducing its quantity by 1. If the item is sold out (quantity = 0), its status is updated to `NotAvailable` . The `sellItem` method in the `ItemService` performs the update. If the item is unavailable (quantity is 0), a `CustomExceptions.ItemNotAvailableException` is thrown.

Parameters

Cancel

Name	Description
id * required `integer($int64)` *(path)*	1

Execute | Clear

Responses

Curl

```
curl -X 'PUT' \
  'http://localhost:8080/items/sell/1' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/items/sell/1
```

Server response

Code	Details
200	**Response body**

```
{
  "id": 1,
  "name": "T-Shirt",
  "quantity": 95,
  "price": 19.99,
  "soldCount": 20,
  "status": "Available",
  "category": "Clothing"
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,01 Apr 2025 05:39:29 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

Code	Description	Links
	/	

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "name": "string",
  "quantity": 0,
  "price": 0,
  "soldCount": 0,
  "status": "Available",
  "category": "Clothing"
}
```

POST `/items/bulkAdd` Bulk add items

This API allows you to add multiple items to the inventory in a single request. The `addBulkItems` method in the `ItemService` is used to insert a list of items into the inventory. If any item fails to be added, a `CustomExceptions.ItemMissingFieldException` is thrown, and the transaction is rolled back.

Parameters

Try it out

No parameters

Request body required

application/json

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

Responses

Code	Description	Links
200	OK Media type */* Controls `Accept` header. **Example Value** Schema	*No links*

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

POST `/items/addItem` Add a new item

This API adds a new item to the inventory. The `addItem` method in the `ItemService` is used to save the new item. If the item has missing fields, a `CustomExceptions.ItemMissingFieldException` is thrown. The added item is returned with its generated ID.

Parameters

Cancel Reset

No parameters

Request body required

application/json

```
{
  "name": "Gaming Monitor",
  "quantity": 40,
  "price": 299.99,
  "soldCount": 10,
  "status": "Available",
  "category": "Electronics"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/items/addItem' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Gaming Monitor",
  "quantity": 40,
  "price": 299.99,
  "soldCount": 10,
  "status": "Available",
  "category": "Electronics"
}
'
```

Request URL

```
http://localhost:8080/items/addItem
```

Server response

Code **Details**

200

Response body

```
{
  "id": 37,
  "name": "Gaming Monitor",
  "quantity": 40,
  "price": 299.99,
  "soldCount": 10,
  "status": "Available",
  "category": "Electronics"
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,01 Apr 2025 05:40:00 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code **Description** **Links**

200

OK

No links

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "name": "string",
  "quantity": 0,
  "price": 0,
  "soldCount": 0,
  "status": "Available",
  "category": "Clothing"
}
```

PATCH `/items/updateQuantity/{id}` Update item quantity

This API updates the quantity of a specific item identified by its `ID`. The `updateItemQuantity` method in the `ItemService` is responsible for modifying the item's quantity and updating its availability status based on the new quantity. If the item doesn't exist, a `CustomExceptions.ItemNotFoundException` is thrown. If there is an issue during the update process, a `CustomExceptions.ItemMissingFieldException` is thrown.

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(path)*	id
quantity * required `integer($int32)` *(query)*	quantity

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "name": "string",
  "quantity": 0,
  "price": 0,
  "soldCount": 0,
```

Code	Description	Links

```
    "status": "Available",
    "category": "Clothing"
  }
```

GET **/items/status** Get items by status

This API filters items based on their availability status (e.g., `Available` or `NotAvailable`). The `getItemsByStatus` method in the `ItemService` is used to retrieve items with a particular status. If no items match the status, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Try it out

Name	Description
status * required `string` *(query)*	*Available values* : Available, NotAvailable Available

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
```

Code	Description	Links

```
      "quantity": 0,
      "price": 0,
      "soldCount": 0,
      "status": "Available",
      "category": "Clothing"
    }
  ]
```

GET **/items/soldCount** Get items sorted by sold count

This API retrieves a list of items sorted by the number of units sold. The `getItemsBySoldCount` method in the `ItemService` is responsible for fetching items sorted by the sold count. If no items are found, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/items/soldCount' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/items/soldCount
```

Server response

Code	Details
200	**Response body**

Code **Details**

```
[
  {
    "id": 4,
    "name": "Headphones",
    "quantity": 200,
    "price": 59.99,
    "soldCount": 150,
    "status": "Available",
    "category": "Electronics"
  },
  {
    "id": 12,
    "name": "Coffee Maker",
    "quantity": 200,
    "price": 89.99,
    "soldCount": 150,
    "status": "Available",
    "category": "Electronics"
  },
  {
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,01 Apr 2025 05:40:17 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
```

Code	Description	Links

```
        "price": 0,
        "soldCount": 0,
        "status": "Available",
        "category": "Clothing"
      }
    ]
```

GET /items/priceRange Get items by price range

This API filters items by a specified price range. The `getItemsByPriceRange` method in the `ItemService` retrieves items whose prices are between `minPrice` and `maxPrice`. If no items match the criteria, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Cancel

Name	Description
minPrice * required `number($double)` *(query)*	30
maxPrice * required `number($double)` *(query)*	50

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/items/priceRange?minPrice=30&maxPrice=50' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/items/priceRange?minPrice=30&maxPrice=50
```

Server response

Code **Details**

200

Response body

```
      "status": "Available",
      "category": "Clothing"
    },
    {
      "id": 21,
      "name": "Tennis Racket",
      "quantity": 15,
      "price": 50,
      "soldCount": 5,
      "status": "NotAvailable",
      "category": "Clothing"
    },
    {
      "id": 35,
      "name": "Backpack",
      "quantity": 150,
      "price": 40,
      "soldCount": 90,
      "status": "Available",
      "category": "Clothing"
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue,01 Apr 2025 05:40:29 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Code	Description	Links

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

GET `/items/name/{name}` Get item by name

This API retrieves a list of items that match the provided name. The `getItemByName` method in the `ItemService` is used to search for items by name. If no matching items are found, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Try it out

Name	Description
name * required `string` *(path)*	name

Responses

Code	Description	Links
200	OK	*No links*

Code	Description	Links

Media type

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

GET `/items/lowCount` Get items with low stock

This API retrieves items with a low stock quantity (typically less than 10). The `getLowCountItems` method in the `ItemService` filters out items that are nearing depletion. If no items have low stock, a `CustomExceptions.EmptyLowCountException` is thrown.

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	OK	*No links*

Media type

Code	Description	Links

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

GET `/items/id/{id}` Get item by ID

This API retrieves an item from the inventory using its unique `ID`. The `getItemById` method in the `ItemService` is used to fetch the item from the repository. If the item is not found, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "name": "string",
  "quantity": 0,
  "price": 0,
  "soldCount": 0,
  "status": "Available",
  "category": "Clothing"
}
```

GET `/items/category` Get items by category

This API retrieves items from the inventory based on their category (e.g., Clothing, Electronics). The `getItemsByCategory` method in the `ItemService` is used to filter items by their category. If no items match the category, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Try it out

Name	Description
category * required `string` *(query)*	*Available values* : Clothing, Electronics, Furniture Clothing

Responses

Code	Description	Links
200	OK Media type */* Controls Accept header. **Example Value** Schema	*No links*

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

GET **`/items/all`** Get all items

This API retrieves a list of all items in the inventory. The **`getAllItems`** method in the **`ItemService`** is used to fetch all items from the database. If the inventory is empty, a **`CustomExceptions.ItemNotFoundException`** is thrown, indicating that no items are found.

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

DELETE `/items/delete/{id}` Delete item by ID

This API deletes an item from the inventory identified by its `ID` . The `deleteItem` method in the `ItemService` deletes the item from the database. If the item is not found, a `CustomExceptions.ItemNotFoundException` is thrown.

Parameters

Try it out

Name	Description
id * required `integer($int64)` *(path)*	id

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
{
  "id": 0,
  "name": "string",
  "quantity": 0,
  "price": 0,
  "soldCount": 0,
  "status": "Available",
  "category": "Clothing"
}
```

DELETE `/items/bulkDelete` Bulk delete items

This API allows you to delete multiple items from the inventory in a single request. The `deleteBulkItems` method in the `ItemService` deletes a list of items based on their `ID` . If any item is not found or a deletion fails, a `CustomExceptions.ItemNotFoundException` is thrown, and all deletions are rolled back.

Parameters

Try it out

No parameters

Request body required

application/json

Example Value Schema

```
[
  0
]
```

Responses

Code	Description	Links
200	OK	*No links*

Media type

/

Controls `Accept` header.

Example Value Schema

```
[
  {
    "id": 0,
    "name": "string",
    "quantity": 0,
    "price": 0,
    "soldCount": 0,
    "status": "Available",
    "category": "Clothing"
  }
]
```

Schemas

```
Item  {
   id                 [...]
   name               [...]
   quantity           [...]
   price              [...]
   soldCount          [...]
   status             [...]
   category           [...]
}
```