RESTful Shop-Demo API Guide

Malte Basse

Table of Contents

Introduction

This document describes all RESTful microservice of this shop demo project.

Overview

HTTP verbs

RESTful notes tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs.

Verb	Usage
`GET`	Used to retrieve a resource
`POST`	Used to create a new resource
`PATCH`	Used to update an existing resource, including partial updates
`DELETE`	Used to delete an existing resource

RESTful notes tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP status codes.

Status code	Usage
`200 OK`	The request completed successfully
`201 Created`	A new resource has been created successfully. The resource's URI is available from the response's `Location` header
`204 No Content`	An update to an existing resource has been applied successfully
`400 Bad Request`	The request was malformed. The response body will include an error providing further information
`404 Not Found`	The requested resource did not exist

Hypermedia

RESTful Notes uses hypermedia and resources include links to other resources in their responses. Responses are in Hypertext Application from resource to resource. Language (HAL) format. Links can be found beneath the `_links` key. Users of the API should not create URIs themselves, instead they should use the above-described links to navigate

Product REST Service

The product rest service is used to create, modify and list products.

Listing product

The `GET` request returns all products.

Path	Type	Description
`content[].productUuid`	`String`	The unique identifier of the product
`content[].name`	`String`	Name of the product
`content[].shortDescription`	`String`	Short product description
`content.[].longDescription`	`String`	Long product description
`content[].links`	`Array`	Link to the next product
`totalPages`	`Number`	Number of total pages.
`totalElements`	`Number`	Total amount of elements.
`last`	`Boolean`	Indicates whether the current page is the first one.
`size`	`Number`	Size of the page.
`number`	`Number`	Number of the current page.
`sort`	`Null`	Sorting parameters for the page.
`first`	`Boolean`	Indicates whether the current page is the first one.
`numberOfElements`	`Number`	Number of elements currently on this page.

Example request

```
$ curl 'http://localhost:8080/products/0/3' -i -H 'Accept: application/json'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 828

{
  "content" : [ {
    "productUuid" : "aff1fb8f-f8da-41e6-a67c-27850555f898",
    "name" : "Product1",
    "shortDescription" : "product1 short description",
    "longDescription" : "product1 long description",
    "links" : [ ]
  }, {
    "productUuid" : "dbdb788c-eb8a-4c98-9a02-9c35ff717181",
    "name" : "Product3",
    "shortDescription" : "product3 short description",
    "longDescription" : "product3 long description",
    "links" : [ ]
  }, {
    "productUuid" : "d7c24759-8ac5-4f03-b1f0-e6545ec964c6",
    "name" : "name updated",
    "shortDescription" : "short description updated",
    "longDescription" : "long description updated",
    "links" : [ ]
  } ],
  "last" : false,
  "totalPages" : 2,
  "totalElements" : 4,
  "size" : 3,
  "number" : 0,
  "first" : true,
  "sort" : null,
  "numberOfElements" : 3
}
```

Returning product

The `GET` request returns a product identified by the unique identifier.

Path	Type	Description
`productUuid`	`String`	The unique identifier of the product
`name`	`String`	Name of the product
`shortDescription`	`String`	Short product description
`longDescription`	`String`	Long product description
`_links.next.href`	`String`	Link to the next product

Example request

```
$ curl 'http://localhost:8080/product/aff1fb8f-f8da-41e6-a67c-27850555f898' -i -H
'Accept: application/hal+json'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json;charset=UTF-8
Content-Length: 284

{
  "productUuid" : "aff1fb8f-f8da-41e6-a67c-27850555f898",
  "name" : "Product1",
  "shortDescription" : "product1 short description",
  "longDescription" : "product1 long description",
  "_links" : {
    "next" : {
      "href" : "http://localhost:8080/product/testuuid"
    }
  }
}
```

Saving/Updating product

The `POST` request saves or updates a product.

Path	Type	Description
`productUuid`	`Null`	The unique identifier of the product
`name`	`String`	Name of the product
`shortDescription`	`String`	Short product description
`longDescription`	`String`	Long product description
`links`	`Array`	Not used, only for the response!

Example request

```
$ curl 'http://localhost:8080/product' -i -X POST -H 'Content-Type: application/json'
-d '{
  "productUuid" : null,
  "name" : "Product2",
  "shortDescription" : "product2 short description",
  "longDescription" : "product2 long description",
  "links" : [ ]
}'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/hal+json;charset=UTF-8
Content-Length: 187

{
  "productUuid" : "bfd644ce-5cb2-47ae-9f90-1639750c62ce",
  "name" : "Product2",
  "shortDescription" : "product2 short description",
  "longDescription" : "product2 long description"
}
```

Deleting product

The `DELETE` request deletes a product identified by the unique identifier.

Example request

```
$ curl 'http://localhost:8080/product/dbdb788c-eb8a-4c98-9a02-9c35ff717181' -i -X
DELETE -H 'Accept: application/json'
```

Example response

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: 1

1
```