## Restful API Design – Online Store

## Base URL

The base URL of the API is **http://localhost/online-store/**

## Access Control

The Online Store can only be accessed by only one user, the administrator, as of now. The Administrator can add, edit, search or delete products and other resources in the system.

## Authentication

Authentication shall happen using HTTP Basic Authentication . The user will have to provide his username and password as authorization credentials:

$ curl -u username:password http://localhost/online-store/v1/products

**HTTP Verbs**

Standard HTTP verbs are used to indicate different types of requests:

* GET - To retrieve a resource or a collection of resources
* POST - To create a resource
* PUT - To set/modify a resource
* DELETE - To delete a resource

## Resources in the System

There is one resources in the system.

## Products: This resource type is for managing products in the store.

## Requests

All POST and PUT requests are JSON encoded and must have content type of ‘application/json’.

## Responses

All response bodies are encoded in JSON.

### HTTP Response Status Codes

HTTP response status codes shall be used to indicate the result of a request.

Success codes:

* 200 OK - Request succeeded. Response included.
* 201 Created - Resource created. URL to the new resource in Location header.
* 204 No Content - Request succeeded, but no response body

Error codes:

* 401 Unauthorized- No authentication credentials provided or authentication failed
* 404 Not Found- Resource not found
* 415 Unsupported Media Type - POST/PUT request occurred without a application/json content type

**Query Parameters**

## Filtering Fields in the Response Object

Responses from the API can limit fields returned to only the fields required. For that, a **‘fields’** query parameter with a comma separated list of field names can be passed.

For example:

GET http://localhost/online-store/products?**fields=product\_id,name**

## Pagination

Pagination can be managed using the **‘per\_page’** and **‘page’** query parameters.

GET http://localhost/online-store/products?**per\_page=10&page=2**

## Sorting

The API offers sorting, using the **‘sort’** query parameter. The value of ‘sort’ should be a comma separated list of fields to sort by. A descending sort can be specified by prepending a **‘-’** to a field name.

For example :- To get recently added products, sorted in descending order of added date:

GET http://localhost/online-store/products?**sort=-created**

## Products

Following are the attributes of the products in the online store:

| **Field** | **Type** | **Notes** |
| --- | --- | --- |
| Id | Integer | Automatically Generated ID of the Product. An auto incremented integer stored. |
| name | varchar(50) | Name of the product. |
| description | text | Description of the Product. |
| sku | varchar(50) | Available quantity of the product in Store. 0 if the item is not available. |
| salePrice | float | Sale Price of product. |
| regularPrice | float | Regular Price of product |
| created | datetime | Timestamp of the time when the product was created. |
| modified | datetime | Timestamp of the time when the product was last updated. |

### Listing products(Searching enabled using query parameters)

GET http://localhost/online-store/products

Sample Request to display all Products

GET http://localhost/online-store/products

200 OK

Content-Type: application/json

[

{

"id": "1",

"name": "testting\_product",

"description": "test desc",

"sku": "testsku",

"salePrice": "10",

"regularPrice": "20",

"created": "2016-06-07 12:38:00",

"modified": "2016-06-08 05:14:15"

},

{

"id": "2",

"name": "testting\_product1",

"description": "test desc1",

"sku": "testsku1",

"salePrice": "25000",

"regularPrice": "300",

"created": "2016-06-07 12:40:00",

"modified": "0000-00-00 00:00:00"

},

{

"id": "3",

"name": "Test product 3",

"description": "Test product 3 desc",

"sku": "test\_pr\_3",

"salePrice": "22",

"regularPrice": "25",

"created": "2016-06-08 13:27:00",

"modified": "0000-00-00 00:00:00"

}

]

Sample Request for Products which have Category ID=8

GET http://localhost/online-store/**products?sku=testsku&fields=id,name**

200 OK

Content-Type: application/json

[

{

"id": "1",

"name": "testting\_product"

}

]

### Getting details of a single product

GET http://localhost/online-store/products/product\_id

A Sample Request:

GET http://localhost/online-store/products/1?fields=id,name,description

200 OK

Content-Type: application/json

[

{

"id": "1",

"name": "testting\_product",

"description": "test desc"

}

]

### Updating a product

### No attribute is compulsory for updating the product. The attributes included in the JSON request object will get updated. The rest shall remain the same

PUT http://localhost/online-store/products/4

Sample Request:

PUT /api/v1/products

Content-Type: application/json

[

{

"id": "4",

"name": "Test product 41",

"description": "Test product 4 desc",

"sku": "test\_pr\_4",

"salePrice": "323",

"regularPrice": "353"

}

]

200 OK

Content-Type: application/json

[

{

"id": "4",

"name": "Test product 41",

"description": "Test product 4 desc",

"sku": "test\_pr\_4",

"salePrice": "323",

"regularPrice": "353",

"created": "2016-06-07 17:09:23",

"modified": "2016-06-08 17:49:02"

}

]

### Deleting a product

DELETE /api/v1/products/5

Content-Type: application/json

{

"deleted": true

}

### Creating a new product

POST http://localhost/online-store/products

Sample Request:

POST /api/v1/products

Content-Type: application/json

[

{

"id": "6",

"name": "Test product 89",

"description": "Test product 89 desc",

"sku": "test\_pr\_89",

"salePrice": "32",

"regularPrice": "35",

"created": "2016-06-08 17:54:28",

"modified": "0000-00-00 00:00:00"

}

]

201 Created

Content-Type: application/json

[

{

"id": "6",

"name": "Test product 89",

"description": "Test product 89 desc",

"sku": "test\_pr\_89",

"salePrice": "32",

"regularPrice": "35",

"created": "2016-06-08 17:54:28",

"modified": "0000-00-00 00:00:00"

}

]