UTILIZACION DE CURL

GET:
with JSON:
```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" http://hostname/resource
```

with XML:
```
curl -H "Accept: application/xml" -H "Content-Type: application/xml" -X GET http://hostname/resourc
```

POST:

For posting data:

```
curl --data "param1=value1&param2=value2" http://hostname/resource
```

For file upload:

```
curl --form "fileupload=@filename.txt" http://hostname/resource
```

RESTful HTTP Post:

```
curl -X POST -d @filename http://hostname/resource
```

For logging into a site (auth):

```
curl -d "username=admin&password=admin&submit=Login" --dump-header headers http://localhost/Login
curl -L -b headers http://localhost/
```

POST application/json

curl -d '{"key1":"value1", "key2":"value2"}' -H "Content-Type: application/json" -X POST http://localhost:3000/data

with a data file

curl -d "@data.json" -X POST http://localhost:3000/data

Creating Data with POST

Improve this doc

Once the application has started up you can create a `Product` instance using your preferred HTTP client. In the following examples we will be using the curl.

To submit a `POST` request use the following in a terminal window:

```
$ curl -i -H "Content-Type:application/json" -X POST localhost:8080/products -d '{"name":"Orange","price":2.0}'
```

reating Data with POST

Improve this doc

Once the application has started up you can create a `Product` instance using your preferred HTTP client. In the following examples we will be using the curl.

To submit a `POST` request use the following in a terminal window:

```
$ curl -i -H "Content-Type:application/json" -X POST localhost:8080/products -d '{"name":"Orange","price":2.0}'
```

As you can see an HTTP 201 status code is returned.

eading Data with GET

Improve this doc

You can read all of the `Product` instances back using a `GET` request:

```
$ curl -i  localhost:8080/products
```

Or read only a single instance by id:

```
$ curl -i  localhost:8080/products/1
```

Which will return:

```
HTTP/1.1 200
X-Application-Context: application:development
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Wed, 23 Nov 2016 08:50:58 GMT

{"id":1,"name":"Orange","price":"$2.0"}
```

4.3 Updating Data with PUT

Improve this doc

To update data you can use a PUT request with the id and in the URI and data you want to change:

```
$ curl -i -H "Content-Type:application/json" -X PUT localhost:8080/products/1 -d '{"price":3.0}'
```

In this case the resulting output will be:

```
HTTP/1.1 200
```

```
X-Application-Context: application:development
Location: http://localhost:8080/products/1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Wed, 23 Nov 2016 08:52:14 GMT

{"id":1,"name":"Orange","price":"$3.0"}
```

If you were to attempt update the data with an invalid value:

```
$ curl -i -H "Content-Type:application/json" -X PUT localhost:8080/products/1 -d
'{"price":-2.0}'
```

Then an error response will be received:

```
HTTP/1.1 422
X-Application-Context: application:development
Location: http://localhost:8080/products/1
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Wed, 23 Nov 2016 08:54:25 GMT

{"message":"Property [price] of class [class hibernate.example.Product] with value
[-2] does not fall within the valid range from [0] to [1,000]","path":"","_links":{"self":
{"href":"http://localhost:8080/products/1"}}}
```

eleting Data with DELETE

Improve this doc

To delete a `Product` simply send a `DELETE` request:

```
$ curl -i -X DELETE localhost:8080/products/1
```

If deleting the instance was successful the output will be:

```
HTTP/1.1 204
X-Application-Context: application:development
Content-Type: application/json;charset=UTF-8
Date: Wed, 23 Nov 2016 08:57:27 GMT
```