0

Your task is to finish implementation of a FooBar REST service and test your result using a REST client. The FooBar service converts numbers into FooBar values.

FooBar values are mostly numbers but for multiples of 3 the value is “Foo” and for the multiples of 5 the value is “Bar”. For numbers which are multiples of both 3 and 5 the value is “FooBar”.

So

```
1 => 1
2 => 2
3 => Foo
4 => 4
5 => Bar
6 => Foo
7 => 7
…
14 => 14
15 => FooBar
```

and so on

1.

Some of the work and wiring is already done but what's missing is the implementation of the heart of the service which produces FooBar values. This logic is to be contained in `com.my.app.FooBarServiceImpl` class. Take a look at the documentation of `com.my.app.FooBarService` to understand the methods which need to be implemented.

In `com.my.app.FooBarController` method implementations are empty. You will have to call `com.my.app.FooBarService` from them and return the appropriate values.

Fill in the implementations of the following verbs

1.1

GET numbers

```
GET /foobars?start=5&end=9 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

["Bar","Foo","7","8","Foo"]
```

1.2

GET number

```
GET /foobars/7 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"7"}
```

1.3

PUT new value. It replaces the value of a number.

For example, GET original value

```
GET /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"Bar"}
```

Update it with `Hello`

```
PUT /foobars/5 HTTP/1.1
Host: localhost:8080
Content-Type: application/json

{"value":"Hello"}

HTTP/1.1 201 Created
Location: http://localhost:8080/foobars/5
Content-Length: 0
```

Now you see the updated value

```
GET /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"Hello"}
```

2.

Implement POST and DELETE with the following semantics

2.1

DELETE resets the number value to its original value.

Take the number 5 updated with the PUT above

```
GET /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"Hello"}
```

Calling DELETE resets it back to `Bar`

```
DELETE /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 204 No Content
```

```
GET /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"Bar"}
```

2.2

POST appends the string value to the existing number value.

As above, the original value of 5 is `Foo`. After this update

```
POST /foobars/5 HTTP/1.1
Host: localhost:8080
Content-Type: application/json

{"value":" and Foo"}

HTTP/1.1 201 Created
```

```
Location: http://localhost:8080/foobars/5
Content-Length: 0
```

the new value is

```
GET /foobars/5 HTTP/1.1
Host: localhost:8080

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked

{"value":"Bar and Foo"}
```

Good luck!

You will access your service using a REST client using the Chrome browser. Please ask for help in using the client.

If you have any questions please don't hesitate to ask.