Εργαστηριακή άσκηση 5: HTTP services (Part 2)

Spring Boot REST Example Continue

1 Configure project

1.1 Edit the pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.4.0</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>gr.upatras</groupId>
	<artifactId>rest.example</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>rest.example</name>
	<description>Demo project for Spring Boot</description>
	<properties>
		<java.version>11</java.version>
		<swagger.version>3.0.0</swagger.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>

		<!-- swagger -->
		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-boot-starter</artifactId>
			<version>${swagger.version}</version>
		</dependency>
		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger2</artifactId>
			<version>${swagger.version}</version>
		</dependency>

		<dependency>
			<groupId>io.springfox</groupId>
			<artifactId>springfox-swagger-ui</artifactId>
			<version>${swagger.version}</version>
		</dependency>


	</dependencies>

	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
```

```
                        <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
        </plugins>
    </build>
    <repositories>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <releases>
                <enabled>false</enabled>
            </releases>
        </repository>
    </repositories>
    <pluginRepositories>
        <pluginRepository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
        <pluginRepository>
            <id>spring-snapshots</id>
            <name>Spring Snapshots</name>
            <url>https://repo.spring.io/snapshot</url>
            <releases>
                <enabled>false</enabled>
            </releases>
        </pluginRepository>
    </pluginRepositories>

</project>
```

1.2 Create the following classes:

1.2.1 Category

```
package gr.upatras.rest.example;

import java.util.ArrayList;
import java.util.List;

/**
 * @author ctranoris
 *
 */
public class Category {

    /**
     *
     */
    private int id;

    /**
     * name of category
     */
```

```
	private String name;


	/**
	 * list of products in category
	 */
	private List<Product> products = new ArrayList<>();


	/**
	 *
	 * constructire
	 * @param id
	 * @param name
	 */
	public Category(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}


	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public List<Product> getProducts() {
		return products;
	}

	public void setProducts(List<Product> products) {
		this.products = products;
	}



}
```

1.2.2 CategoryController

```
package gr.upatras.rest.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
```

```
import io.swagger.annotations.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@RestController
public class CategoryController {
	@Autowired
	private ICategoryService categoryService;


	private static final Logger log = LoggerFactory.getLogger( CategoryController.class);

	@ApiOperation(value = "Retrieves all Categorys",  notes = "This operation retrieves all Category entities. ",
response = Category.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Category.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/category/" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.GET)
	public List<Category> getCategory() {
		// finds all the categorys
		List<Category> categorys = categoryService.findAll();
		// returns the category list
		return categorys;
	}

	@ApiOperation(value = "Retrieves a Category by ID",  notes = "This operation retrieves a Category entity. ", response
= Category.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Category.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/category/{id}" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.GET)
	public Category getCategoryById( @ApiParam(value = "Identifier of the Category", required = true) @PathVariable("id")
int id) {

		log.info( String.format( "Will return category with id %s" , id) );
		Category category = categoryService.findById(id);
		return category;
	}

	@ApiOperation(value = "Deletes a Category by ID",  notes = "This operation retrieves a Category entity. ", response =
Category.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Category.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/category/{id}" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.DELETE)
	public ResponseEntity<Void> deletetById(@ApiParam(value = "Identifier of the Category", required = true)
@PathVariable("id") int id) {

		try {
```

```
                        log.info( String.format( "Will delete object with id %s" , id) );
                        return new ResponseEntity<Void>( categoryService.deleteCategory(id), HttpStatus.OK);
                } catch (Exception e) {
                        log.error("Couldn't serialize response for content type application/json", e);
                        return new ResponseEntity<Void>(HttpStatus.INTERNAL_SERVER_ERROR);
                }
        }


        @ApiOperation(value = "Creates a Category", notes = "This operation creates a Category entity.", response =
Category.class)
        @ApiResponses(value = { @ApiResponse(code = 201, message = "Created", response = Category.class),
                        @ApiResponse(code = 400, message = "Bad Request", response = Error.class),
                        @ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
                        @ApiResponse(code = 403, message = "Forbidden", response = Error.class),
                        @ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
                        @ApiResponse(code = 409, message = "Conflict", response = Error.class),
                        @ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
        @RequestMapping(value = "/category", produces = { "application/json;charset=utf-8" }, consumes = {
"application/json;charset=utf-8" }, method = RequestMethod.POST)
        public ResponseEntity<Category> createCategory(@ApiParam(value = "The Category to be created", required = true)
@RequestBody Category p) {


                log.info(   "Will add a new category"  );
                Category category = categoryService.addCategory(p);
                return new ResponseEntity<Category>( category, HttpStatus.OK);
        }


        @ApiOperation(value = "Updates partially a Category", nickname = "patchServiceTestSpecification", notes = "This
operation updates partially a Category entity.", response = Category.class )
        @ApiResponses(value = { @ApiResponse(code = 200, message = "Updated", response = Category.class),
                        @ApiResponse(code = 400, message = "Bad Request", response = Error.class),
                        @ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
                        @ApiResponse(code = 403, message = "Forbidden", response = Error.class),
                        @ApiResponse(code = 404, message = "Not Found", response = Error.class),
                        @ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
                        @ApiResponse(code = 409, message = "Conflict", response = Error.class),
                        @ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
        @RequestMapping(value = "/category/{id}", produces = {
                        "application/json;charset=utf-8" }, consumes = {
                                        "application/json;charset=utf-8" }, method = RequestMethod.PATCH)
        ResponseEntity<Category> patchCategory(
                        @ApiParam(value = "The Category to be updated", required = true)  @RequestBody Category body,
                        @ApiParam(value = "Identifier of the Category", required = true) @PathVariable("id") String id) {


                Category category = categoryService.editCategory(body);
                return new ResponseEntity<Category>( category, HttpStatus.OK);
        }

}
```

1.2.3 ICategoryService

```
package gr.upatras.rest.example;

import java.util.List;

/**
 * @author ctranoris
 *
 */
public interface ICategoryService {
        /**
         * @return all categories
         */
```

```
        List<Category> findAll();

        /**
         * @param id
         * @return a {@link Category}
         */
        Category findById(int id);

        /**
         * @param c
         * @return the Category added
         */
        Category addCategory(Category c);

        /**
         * @param c
         * @return the edited {@link Category}
         */
        Category editCategory(Category c);

        /**
         * @param id of Category
         */
        Void deleteCategory(int id);

}
```

1.2.4 CategoryService

```
package gr.upatras.rest.example;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 * @author ctranoris
 *
 */

@Service
public class CategoryService implements ICategoryService {

        // creating an object of ArrayList
        List<Category> categories = new ArrayList<Category>();

        @Autowired
        private IProductService productService;

        int ix = 10;


        public CategoryService() {
                super();

//              categories.add( new Category(1, "TV"));
//              categories.add( new Category(2, "Electronics"));
//              categories.add( new Category(3, "Home & Kitchen"));
        }

        @Override
        public List<Category> findAll() {
                return categories;
        }
```

```
	@Override
	public Category findById(int id) {
		for (Category c : categories) {
			if (c.getId() == id) {
				return c;
			}
		}
		return null;
	}

	@Override
	public Category addCategory(Category catToAdd) {
		ix = ix +1; //increase product index
		Category c = new Category(ix, catToAdd.getName());
		for (Product p : catToAdd.getProducts()) {
			Product productToAdd = productService.findById(p.getId());
			if ( productToAdd != null ) {
				c.getProducts().add(productToAdd);
			}

		}
		categories.add( c );
		return c;
	}

	@Override
	public Category editCategory(Category catToAdd) {
		Category editCat = findById( catToAdd.getId() );
		editCat.getProducts().clear();
		if ( editCat != null ) {
			editCat.setName( catToAdd.getName() );
			for (Product p : catToAdd.getProducts()) {
				Product productToAdd = productService.findById(p.getId());
				if ( productToAdd != null ) {
					editCat.getProducts().add(productToAdd);
				}

			}

			return editCat;
		}
		return null;
	}

	@Override
	public Void deleteCategory(int id) {
		for (Category p : categories) {
			if (p.getId() == id) {
				categories.remove(p);
				break;
			}
		}
		return null;

	}

}
```

1.2.5 IProductService

```
package gr.upatras.rest.example;

import java.util.List;

/**
 * @author ctranoris
 *
```

```
 */
public interface IProductService {
	/**
	 * @return all products
	 */
	List<Product> findAll();

	/**
	 * @param id
	 * @return a {@link Product}
	 */
	Product findById(int id);

	/**
	 * @param p
	 * @return the @Product added
	 */
	Product addProduct(Product p);

	/**
	 * @param p
	 * @return the edited {@link Product}
	 */
	Product editProduct(Product p);


	/**
	 * @param id of product
	 */
	Void deleteProduct(int id);

}
```

1.2.6 Product

```
package gr.upatras.rest.example;

/**
 * @author ctranoris
 *
 */
public class Product {

	private int id;
	private String pname;
	private String batchno;
	private double price;
	private int noofproduct;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getPname() {
		return pname;
	}

	public void setPname(String pname) {
		this.pname = pname;
	}

	public String getBatchno() {
		return batchno;
	}

	public void setBatchno(String batchno) {
		this.batchno = batchno;
```

```
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public int getNoofproduct() {
		return noofproduct;
	}

	public void setNoofproduct(int noofproduct) {
		this.noofproduct = noofproduct;
	}

	public Product(int id, String pname, String batchno, double price, int noofproduct) {
		super();
		this.id = id;
		this.pname = pname;
		this.batchno = batchno;
		this.price = price;
		this.noofproduct = noofproduct;
	}


}
```

1.2.7 ProductController

```
package gr.upatras.rest.example;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import io.swagger.annotations.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@RestController
public class ProductController {
	@Autowired
	private IProductService productService;


	private static final Logger log = LoggerFactory.getLogger( ProductController.class);

	@ApiOperation(value = "Retrieves all Products",  notes = "This operation retrieves all Product entities. ", response
= Product.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Product.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
```

```
	@RequestMapping(value = "/product/" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.GET)
	public List<Product> getProduct() {
		// finds all the products
		List<Product> products = productService.findAll();
		// returns the product list
		return products;
	}

	@ApiOperation(value = "Retrieves a Product by ID",  notes = "This operation retrieves a Product entity. ", response =
Product.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Product.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/product/{id}" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.GET)
	public Product getProductById( @ApiParam(value = "Identifier of the Category", required = true) @PathVariable("id")
int id) {

		log.info( String.format( "Will return product with id %s" , id) );
		Product product = productService.findById(id);
		return product;
	}

	@ApiOperation(value = "Deletes a Product by ID",  notes = "This operation retrieves a Product entity. ", response =
Product.class)
	@ApiResponses(value = { @ApiResponse(code = 200, message = "Success", response = Product.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 404, message = "Not Found", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/product/{id}" , produces = {	"application/json;charset=utf-8" }, method =
RequestMethod.DELETE)
	public ResponseEntity<Void> deletetById( @ApiParam(value = "Identifier of the Category", required = true)
@PathVariable("id") int id) {

		try {

			log.info( String.format( "Will delete object with id %s" , id) );
			return new ResponseEntity<Void>( productService.deleteProduct(id), HttpStatus.OK);
		} catch (Exception e) {
			log.error("Couldn't serialize response for content type application/json", e);
			return new ResponseEntity<Void>(HttpStatus.INTERNAL_SERVER_ERROR);
		}
	}


	@ApiOperation(value = "Creates a Product", notes = "This operation creates a Product entity.", response =
Product.class)
	@ApiResponses(value = { @ApiResponse(code = 201, message = "Created", response = Product.class),
			@ApiResponse(code = 400, message = "Bad Request", response = Error.class),
			@ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
			@ApiResponse(code = 403, message = "Forbidden", response = Error.class),
			@ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
			@ApiResponse(code = 409, message = "Conflict", response = Error.class),
			@ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
	@RequestMapping(value = "/product", produces = { "application/json;charset=utf-8" }, consumes = {
"application/json;charset=utf-8" }, method = RequestMethod.POST)
	public ResponseEntity<Product> createProduct(@ApiParam(value = "The Product to be created", required = true)
@RequestBody Product p) {
```

```
                log.info(   "Will add a new product"  );
                Product product = productService.addProduct(p);
                return new ResponseEntity<Product>( product, HttpStatus.OK);
        }


        @ApiOperation(value = "Updates partially a Product", nickname = "patchProduct", notes = "This operation updates
partially a Product entity.", response = Product.class )
        @ApiResponses(value = { @ApiResponse(code = 200, message = "Updated", response = Product.class),
                        @ApiResponse(code = 400, message = "Bad Request", response = Error.class),
                        @ApiResponse(code = 401, message = "Unauthorized", response = Error.class),
                        @ApiResponse(code = 403, message = "Forbidden", response = Error.class),
                        @ApiResponse(code = 404, message = "Not Found", response = Error.class),
                        @ApiResponse(code = 405, message = "Method Not allowed", response = Error.class),
                        @ApiResponse(code = 409, message = "Conflict", response = Error.class),
                        @ApiResponse(code = 500, message = "Internal Server Error", response = Error.class) })
        @RequestMapping(value = "/product/{id}", produces = {
                        "application/json;charset=utf-8" }, consumes = {
                                        "application/json;charset=utf-8" }, method = RequestMethod.PATCH)
        ResponseEntity<Product> patchProduct(
                        @ApiParam(value = "The Product to be updated", required = true)  @RequestBody Product body,
                        @ApiParam(value = "Identifier of the Product", required = true) @PathVariable("id") String id) {


                Product product = productService.editProduct(body);
                return new ResponseEntity<Product>( product, HttpStatus.OK);
        }

}
```

1.2.8 ProductService

```
package gr.upatras.rest.example;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;

/**
 * @author ctranoris
 *
 */

@Service
public class ProductService implements IProductService {

        // creating an object of ArrayList
        List<Product> products = new ArrayList<Product>();

        int ix = 1000;
        /**
         *  adding products to the List
         */
        public ProductService() {
                super();

                products.add(new Product(100, "Mobile", "CLK98123", 9000.00, 6));
                products.add(new Product(101, "Smart TV", "LGST09167", 60000.00, 3));
                products.add(new Product(102, "Washing Machine", "38753BK9", 9000.00, 7));
                products.add(new Product(103, "Laptop", "LHP29OCP", 24000.00, 1));
                products.add(new Product(104, "Air Conditioner", "ACLG66721", 30000.00, 5));
                products.add(new Product(105, "Refrigerator ", "12WP9087", 10000.00, 4));
        }

        /**
```

```
         * returns a list of product
         */
        @Override
        public List<Product> findAll() {
                return products;
        }

        @Override
        public Product findById(int id) {
                for (Product p : products) {
                        if (p.getId() == id) {
                                return p;
                        }
                }
                return null;
        }

        @Override
        public Product addProduct(Product p) {
                ix = ix +1; //increase product index
                p.setId( ix  );
                products.add( p );
                return p;
        }

        @Override
        public Product editProduct(Product p) {
                Product editProd = findById( p.getId() );
                if ( editProd != null ) {
                        editProd.setPname( p.getPname() );
                        editProd.setPrice( p.getPrice() );

                        return editProd;
                }
                return null;
        }

        @Override
        public Void deleteProduct(int id) {
                for (Product p : products) {
                        if (p.getId() == id) {
                                products.remove(p);
                                break;
                        }
                }
                return null;

        }
}
```

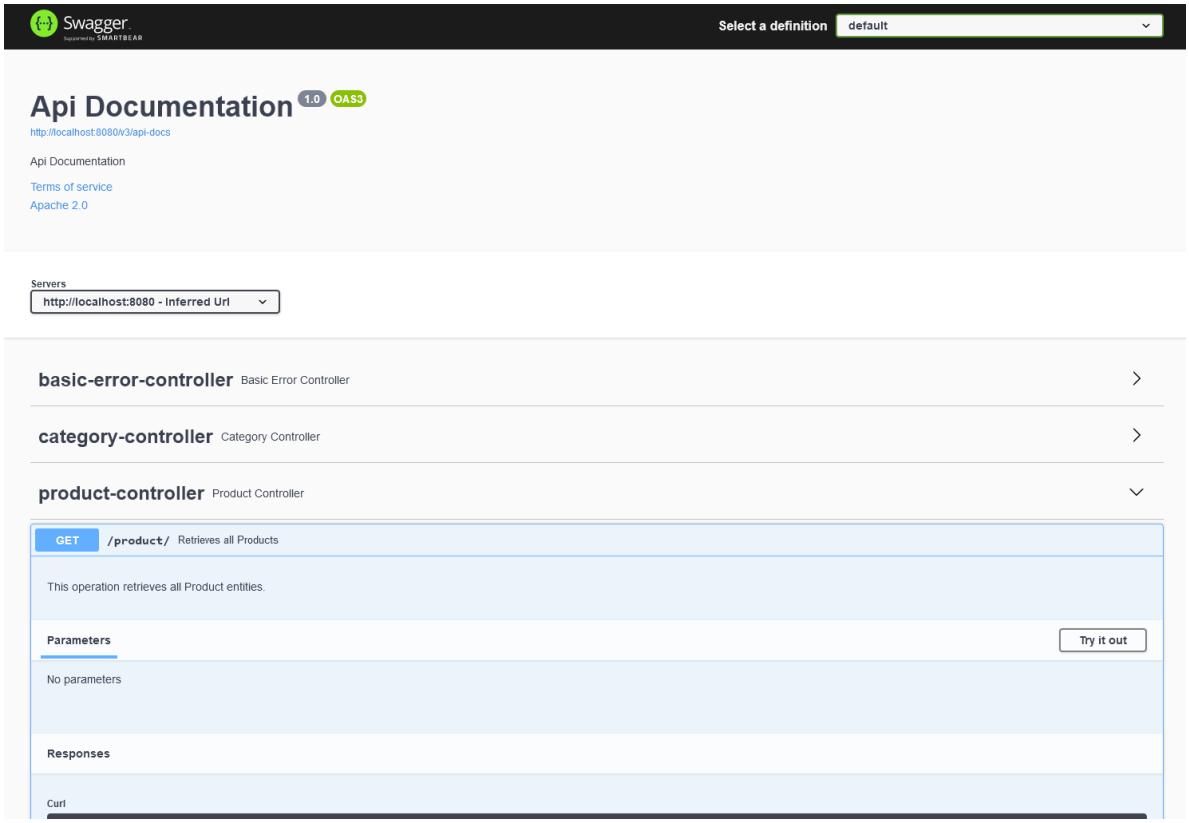
1.3 Run application

2 Monitor swagger and play with the REST API

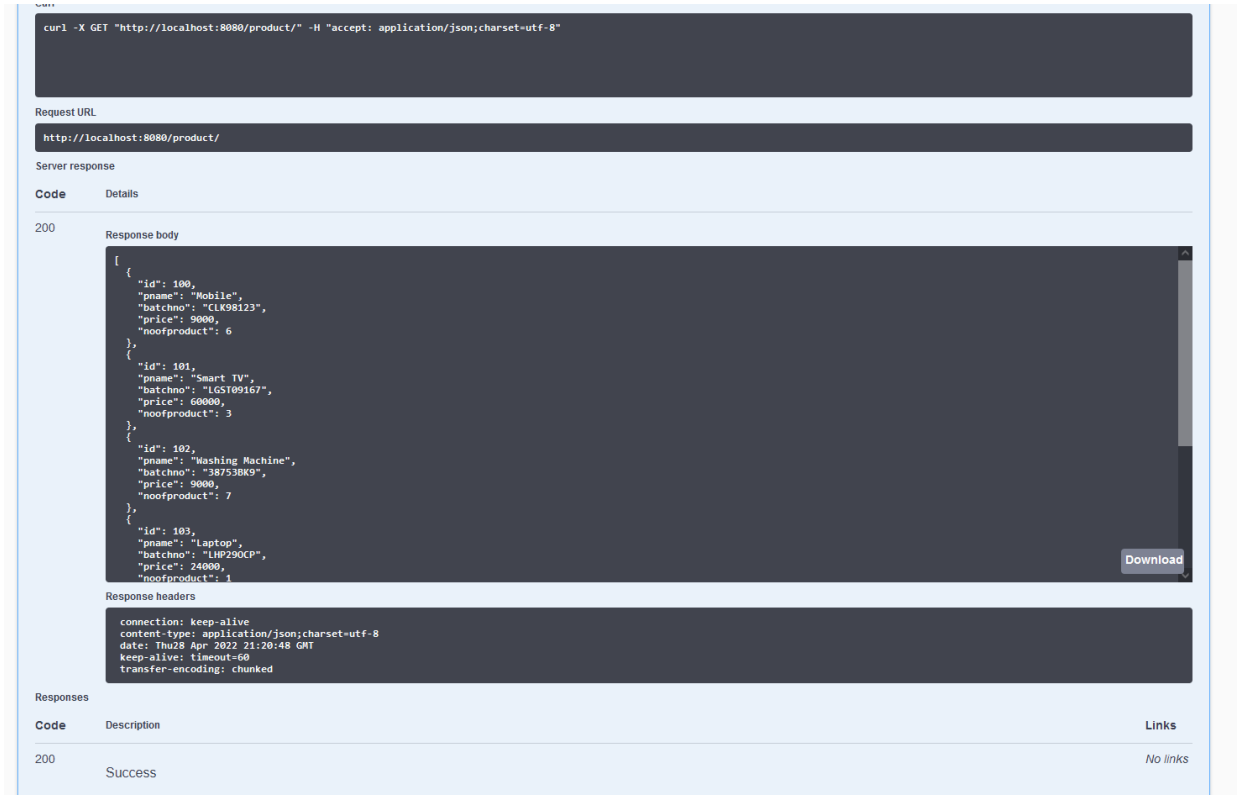
Go to:

http://localhost:8080/swagger-ui/

open product-controller, click Try it out click Execute



See the result:



2.1 Try to get Product with id.

Try to get Product with id 100.

2.2 Insert a new product

Insert with POST the following product:

```
{
  "batchno": "ABC987",
  "noofproduct": 3,
  "pname": "MobilePhoneX",
  "price": 456
}
```

See the result in Response body:

```
{
  "id": 1001,
  "pname": "MobilePhoneX",
  "batchno": "ABC987",
  "price": 456,
  "noofproduct": 3
}
```

Try to get Product with id 1001.

2.3 Check categories

- GET categories are empty

- Insert a new category and enter products

```
{
  "name": "Electronics",
  "products": [
    {
      "id": 100
    },

    {
```

```
      "id": 101
    }
  ]
}
```

Observe the output of Response body

- Edit category with PATCH

```
{
  "id": 11,
  "name": "Electronics",
  "products": [
    {
      "id": 102
    },

    {
      "id": 101
    }
  ]
}
```

Observe the output of Response body

- Insert more categories, add products into categories and GET categories

3 Interact with Command line

3.1 Linux curl command

```
curl -X GET "http://localhost:8080/product/" -H  "accept: application/json;charset=utf-8"
```

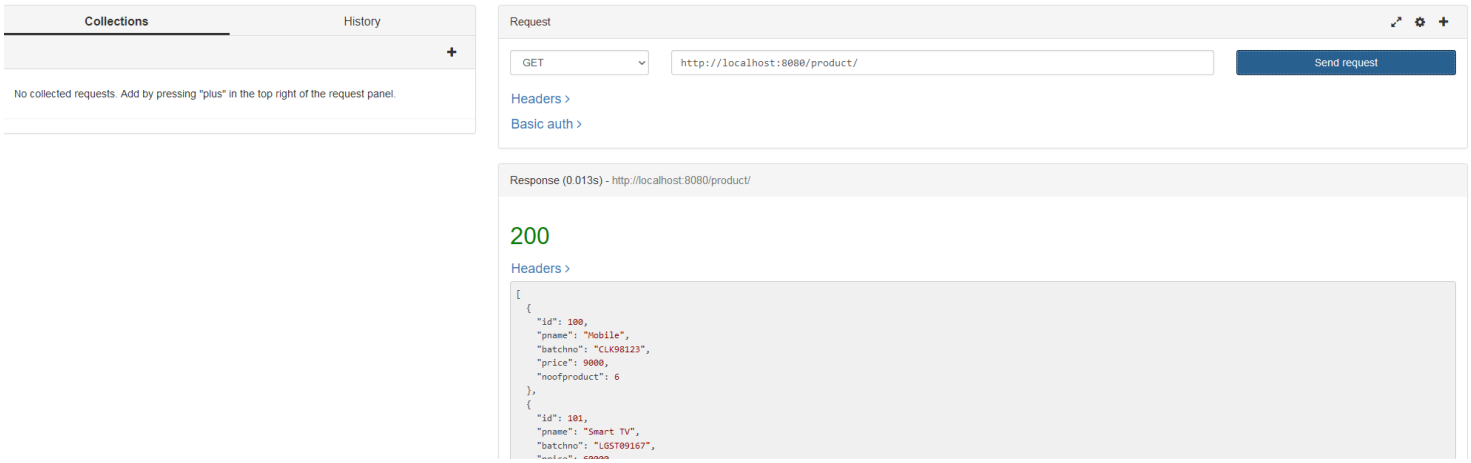
```
curl -X POST "http://localhost:8080/product" -H  "accept: application/json;charset=utf-8" -
H  "Content-Type: application/json" -d
"{\"batchno\":\"ABC987\",\"noofproduct\":3,\"pname\":\"MobilePhoneX\",\"price\":456}"
```

4 Interact with another tool

4.1 RESTED extension (Firefox)

GET example:

</> RESTED



POST: