웹프레임워크2(N)

-과제1

학번: 1771324

이름: 김수연

제출일: 2021.04.25

1) 설정 파일:

<dao-context.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xmlns:tx="http://www.springframework.org/schema/tx"
	xmlns:jpa="http://www.springframework.org/schema/data/jpa"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context	http://www.springframework.org/schema/context/spring-
context-4.3.xsd
		http://www.springframework.org/schema/data/jpa http://www.springframework.org/schema/data/jpa/spring-
jpa.xsd
		http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">



	<bean id="dataSource"
		class="org.apache.commons.dbcp2.BasicDataSource"
		destroy-method="close">
		<property name="driverClassName"
			value="${jdbc.driverClassName}" />
		<property name="url" value="${jdbc.url}" />
		<property name="username" value="${jdbc.username}" />
		<property name="password" value="${jdbc.password}" />
	</bean>

	<context:annotation-config></context:annotation-config>

	<context:property-placeholder
		location="/WEB-INF/props/jdbc.properties" />



	<bean id="sessionFactory"
		class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
		<property name="dataSource" ref="dataSource"></property>
		<property name="packagesToScan">
			<list>
				<value>kr.ac.hansung.entity</value>
			</list>
		</property>

		<property name="hibernateProperties">
			<props>
				<prop key="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</prop>
				<prop key="hibernate.hbm2ddl.auto">create</prop>
				<prop key="hibernate.show_sql">true</prop>
				<prop key="hibernate.format_sql">false</prop>
			</props>
		</property>

	</bean>

	<tx:annotation-driven
		transaction-manager="transactionManager" />

	<bean id="transactionManager"
		class="org.springframework.orm.hibernate5.HibernateTransactionManager">
		<property name="sessionFactory" ref="sessionFactory"></property>
	</bean>

	<context:component-scan base-package="kr.ac.hansung.dao">
	</context:component-scan>

</beans>
```

<service-context.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
		http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-
context-4.3.xsd">

	<context:annotation-config></context:annotation-config>

	<context:component-scan base-package="kr.ac.hansung.service">
	</context:component-scan>
```

```
</beans>
```

<servlet-context.xml>

```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd
		http://www.springframework.org/schema/beans	https://www.springframework.org/schema/beans/spring-
beans.xsd
		http://www.springframework.org/schema/context	https://www.springframework.org/schema/context/spring-
context.xsd">

	<!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->

	<!-- Enables the Spring MVC @Controller programming model -->
	<annotation-driven />

	<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the
${webappRoot}/resources directory -->
	<resources mapping="/resources/**" location="/resources/" />

	<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
	<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
		<beans:property name="prefix" value="/WEB-INF/views/" />
		<beans:property name="suffix" value=".jsp" />
	</beans:bean>

	<context:component-scan base-package="kr.ac.hansung.controller" />

</beans:beans>
```

2) 컨트롤러 파일:

<ProductController.java>

```
@RestController
@RequestMapping(path = "/api/products")
public class ProductController {


	@Autowired
	private ProductService productService;

	@RequestMapping(method = RequestMethod.GET)
	public ResponseEntity<?> retrieveAllProducts() {

		// Getting all products in application...
		final List<Product> products = productService.getAllProducts();

		if (products.isEmpty()) {
			return new ResponseEntity<>(HttpStatus.NO_CONTENT);
		}

		return new ResponseEntity<List<Product>>(products, HttpStatus.OK);
	}

	@RequestMapping(path = "/{id}", method = RequestMethod.GET)
	public ResponseEntity<Product> retrieveProduct(@PathVariable Long id) {
		Product product = productService.getProductById(id);
		if (product == null) {
			throw new NotFoundException(id);
		}
		return new ResponseEntity<Product>(product, HttpStatus.OK);

	}

	// DTO(Data Transfer Object) : 계층간 데이터 교환을 위한 객체, 여기서는 클라이언트(Postman)에서 오는 데이터를 수신할 목적으로 사용
    // Product와 ProductDto와의 차이를 비교해서 살펴보기 바람

    @RequestMapping(method = RequestMethod.POST)
	public ResponseEntity<Product> createProduct(@RequestBody @Valid ProductDto request) {

		Product product = productService.createProduct(request.getName(), request.getPrice());

		return new ResponseEntity<Product>(product, HttpStatus.CREATED);
	}

	@RequestMapping(path = "/{id}", method = RequestMethod.PUT)
	public ResponseEntity<Product> updateProduct(@PathVariable Long id, @RequestBody @Valid ProductDto request) {
		Product currentProduct = productService.getProductById(id);
		if (currentProduct == null) {
			throw new NotFoundException(id);
		}

		currentProduct.setName(request.getName());
		currentProduct.setPrice(request.getPrice());

		productService.updateProduct(currentProduct);
```

```
			return new ResponseEntity<Product>(currentProduct, HttpStatus.OK);
		}

		@RequestMapping(path = "/{id}", method = RequestMethod.DELETE)
		public ResponseEntity<Void> deleteProduct(@PathVariable Long id) {
			// Getting the requiring product; or throwing exception if not found
			final Product product = productService.getProductById(id);

			if(product == null)
				throw new NotFoundException(id);

			// Deleting product from the application...
			productService.deleteProduct(product);

			return new ResponseEntity<>(HttpStatus.NO_CONTENT); //아래와 동일한 기능
			//return ResponseEntity.noContent().build();

		}

		@Getter
		@Setter
		static class ProductDto {

		@NotNull(message = "name is required")
		@Size(message = "name must be equal to or lower than 300", min = 1, max = 300)
		private String name;

		@NotNull(message = "name is required")
		@Min(0)
		private Double price;
		}
}
```

<CategoryController.java>

```
@RestController
@RequestMapping(path = "/api/categories")
public class CategoryController {


		@Autowired
		private CategoryService categoryService;

		@RequestMapping(method = RequestMethod.GET)
		public ResponseEntity<?> retrieveAllCategories() {
			// Getting all categories in application...
			final List<Category> categories = categoryService.getAllCategories();

			if (categories.isEmpty()) {
```

```
                        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
                }

                return ResponseEntity.ok(categories);
                // return new ResponseEntity<List<Category>>(categories, HttpStatus.OK);

        }

        @RequestMapping(path = "/{id}", method = RequestMethod.GET)
        public ResponseEntity<?> retrieveCategory(@PathVariable Long id) {
                final Category category = categoryService.getCategoryById(id);
                if (category == null) {
                        throw new NotFoundException(id);
                }
                return ResponseEntity.ok(category);
        }

        // DTO(Data Transfer Object) : 계층간 데이터 교환을 위한 객체, 여기서는 클라이언트(Postman)에서 오는 데이터를
        // 수신할 목적으로 사용
        // Category와 CategoryDto와의 차이를 비교해서 살펴보기 바람
        @RequestMapping(method = RequestMethod.POST)
        public ResponseEntity<?> createCategory(@RequestBody @Valid CategoryDto request) {

                // Creating a new category in the application...
                final Category category = categoryService.createCategory(request.getName());

                // return new ResponseEntity<Category>(category, HttpStatus.CREATED);
                return ResponseEntity.status(HttpStatus.CREATED).body(category);
        }

        @RequestMapping(path = "/{id}", method = RequestMethod.PUT)
        public ResponseEntity<?> updateCategory(@PathVariable Long id, @RequestBody @Valid CategoryDto request) {
                Category currentCategory = categoryService.getCategoryById(id);

                if (currentCategory == null) {
                        throw new NotFoundException(id);
                }

                currentCategory.setName(request.getName());

                categoryService.updateCategory(currentCategory);

                return ResponseEntity.ok(currentCategory);

        }

        @RequestMapping(path = "/{id}", method = RequestMethod.DELETE)
        public ResponseEntity<?> deleteCategory(@PathVariable Long id) {
                // Getting the requiring category; or throwing exception if not found
                final Category category = categoryService.getCategoryById(id);

                if (category == null)
                        throw new NotFoundException(id);

                // Deleting category from the application...
                categoryService.deleteCategory(category);

                return ResponseEntity.noContent().build();
                // return new ResponseEntity<>(HttpStatus.NO_CONTENT);

        }

        static class CategoryDto {
                @NotNull(message = "name is required")
                @Size(message = "name must be equal to or lower than 100", min = 1, max = 100)
                private String name;

                public String getName() {
                        return name;
                }

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

}
```

<CategorySubcategoriesController.java>

```
@RestController
@RequestMapping(path = "/api/categories/{parentid}/subcategories")
public class CategorySubcategoriesController {


    @Autowired
    private CategoryService categoryService;

    @RequestMapping(method = RequestMethod.GET)
    public ResponseEntity<?> retrieveAllSubcategories(@PathVariable Long parentid) {

        // Getting the requiring category; or throwing exception if not found
        final Category parent = categoryService.getCategoryById(parentid);
        if( parent == null)
         throw   new NotFoundException(parentid);

        // Getting all categories in application...
        final Set<Category> subcategories = parent.getChildCategories();

        return ResponseEntity.ok(subcategories);
    }

    @RequestMapping(path = "/{childid}", method = RequestMethod.POST)
    public ResponseEntity<?> addSubcategory(@PathVariable Long parentid, @PathVariable Long childid) {
         final Category parent = categoryService.getCategoryById(parentid);
         if (parent == null)
                            throw new NotFoundException(parentid);

         final Category child = categoryService.getCategoryById(childid);
        if( child == null)
         throw   new NotFoundException(childid);

        if (categoryService.isChildCategory(child, parent)) {
            throw new IllegalArgumentException("category " + parent.getId() + " already contains subcategory " + child.getId());
        }

        categoryService.addChildCategory(child, parent);

        return ResponseEntity.status(HttpStatus.CREATED).body(child);
    }

    @RequestMapping(path = "/{childid}", method = RequestMethod.DELETE)
    public ResponseEntity<?> removeSubcategory(@PathVariable Long parentid, @PathVariable Long childid) {

        // Getting the requiring category; or throwing exception if not found
        final Category parent = categoryService.getCategoryById(parentid);
        if( parent == null)
         throw   new NotFoundException(parentid);

        // Getting the requiring category; or throwing exception if not found
        final Category child = categoryService.getCategoryById(childid);
        if( child == null)
         throw   new NotFoundException(childid);

        // Validating if association exists...
        if (!categoryService.isChildCategory(child, parent)) {
            throw   new   IllegalArgumentException("category   "   +   parent.getId()   +   "   does   not   contain   subcategory   "   +
child.getId());
        }

        // Dis-associating parent with subcategory...
        categoryService.removeChildCategory(child, parent);

        return ResponseEntity.noContent().build();
    }
```

```
}
```

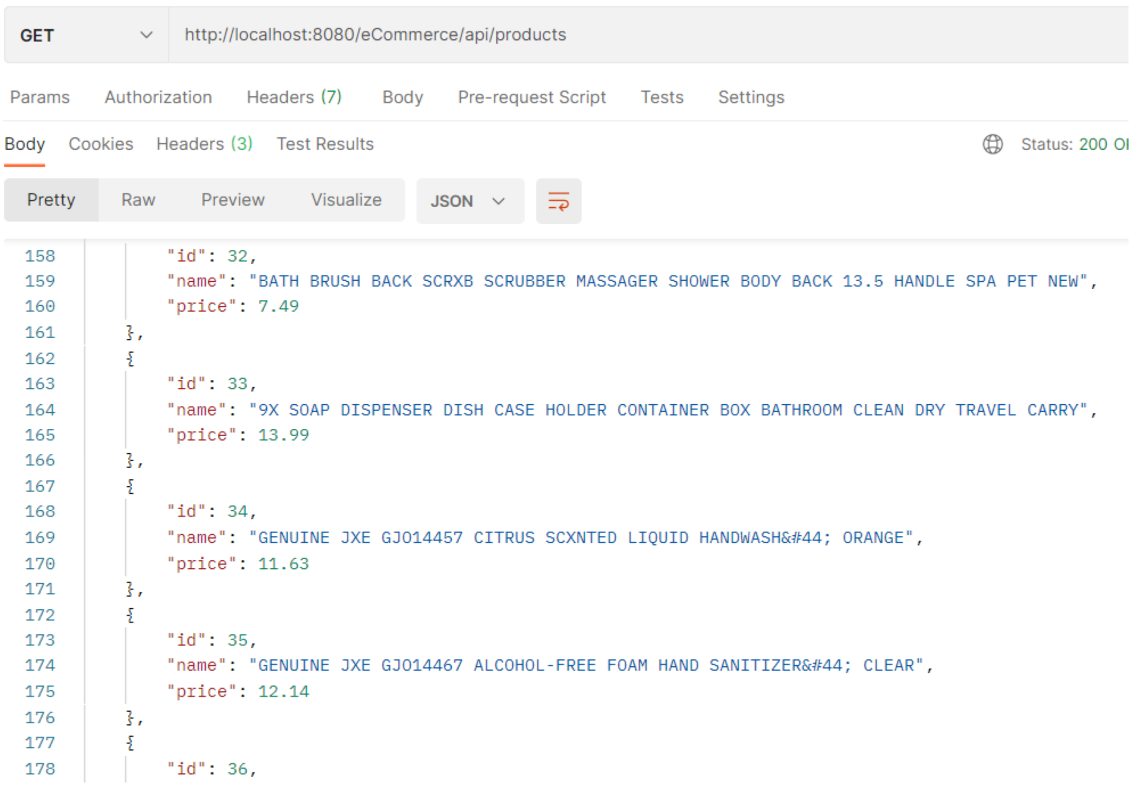
<CategoryProductsController.java>

```
@RestController
@RequestMapping(path = "/api/categories/{categoryid}/products")
public class CategoryProductsController {

	@Autowired
	private CategoryService categoryService;

	@Autowired
	private ProductService productService;

	@RequestMapping(method = RequestMethod.GET)
	public ResponseEntity<?> retrieveAllProducts(@PathVariable Long categoryid) {
		final Category category = categoryService.getCategoryById(categoryid);
		if (category == null) {
			throw new NotFoundException(categoryid);
		}

		Set<Product> products = category.getProducts();

		return ResponseEntity.ok(products);
	}

	@RequestMapping(path = "/{productid}", method = RequestMethod.POST)
	public ResponseEntity<?> addProduct(@PathVariable Long categoryid, @PathVariable Long productid) {

		// Getting the requiring category; or throwing exception if not found
		final Category category = categoryService.getCategoryById(categoryid);
		if (category == null)
			throw new NotFoundException(categoryid);

		// Getting the requiring product; or throwing exception if not found
		final Product product = productService.getProductById(productid);
		if (product == null)
			throw new NotFoundException(productid);

		// Validating if association does not exist...
		if (productService.hasCategory(product, category)) {
			throw new IllegalArgumentException(
						"product " + product.getId() + " already contains category " + category.getId());
		}

		// Associating product with category...
		productService.addCategory(product, category);
```

```
                return ResponseEntity.status(HttpStatus.CREATED).body(product);
        }

        @RequestMapping(path = "/{productid}", method = RequestMethod.DELETE)
        public ResponseEntity<?> removeProduct(@PathVariable Long categoryid, @PathVariable Long productid) {
                final Category category = categoryService.getCategoryById(categoryid);
                if (category == null)
                        throw new NotFoundException(categoryid);

                final Product product = productService.getProductById(productid);
                if (product == null)
                        throw new NotFoundException(productid);

                if (!productService.hasCategory(product, category)) {
                        throw new IllegalArgumentException(
                                        "product " + product.getId() + " does not contain category " + category.getId());
                }

                productService.removeCategory(product, category);

                return ResponseEntity.noContent().build();
        }

}
```

Products:

1) Get, http://localhost:8080/eCommerce/api/products


GET http://localhost:8080/eCommerce/api/products

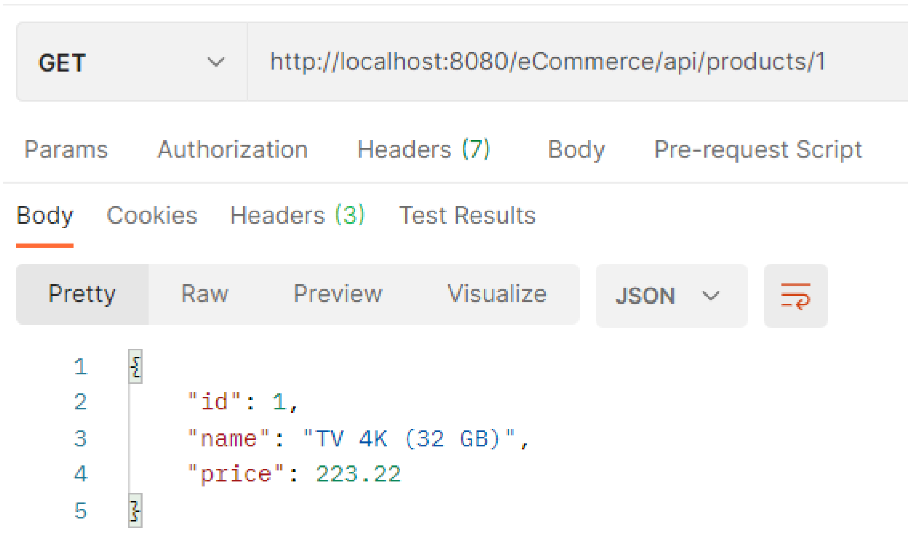
Params Authorization Headers (7) Body Pre-request Script Tests Settings

Body Cookies Headers (3) Test Results Status: 200 OI

Pretty Raw Preview Visualize JSON

```
        "id": 32,
        "name": "BATH BRUSH BACK SCRXB SCRUBBER MASSAGER SHOWER BODY BACK 13.5 HANDLE SPA PET NEW",
        "price": 7.49
    },
    {
        "id": 33,
        "name": "9X SOAP DISPENSER DISH CASE HOLDER CONTAINER BOX BATHROOM CLEAN DRY TRAVEL CARRY",
        "price": 13.99
    },
    {
        "id": 34,
        "name": "GENUINE JXE GJ014457 CITRUS SCXNTED LIQUID HANDWASH&#44; ORANGE",
        "price": 11.63
    },
    {
        "id": 35,
        "name": "GENUINE JXE GJ014467 ALCOHOL-FREE FOAM HAND SANITIZER&#44; CLEAR",
        "price": 12.14
    },
    {
        "id": 36,
```

2) Get, http://localhost:8080/eCommerce/api/products/1

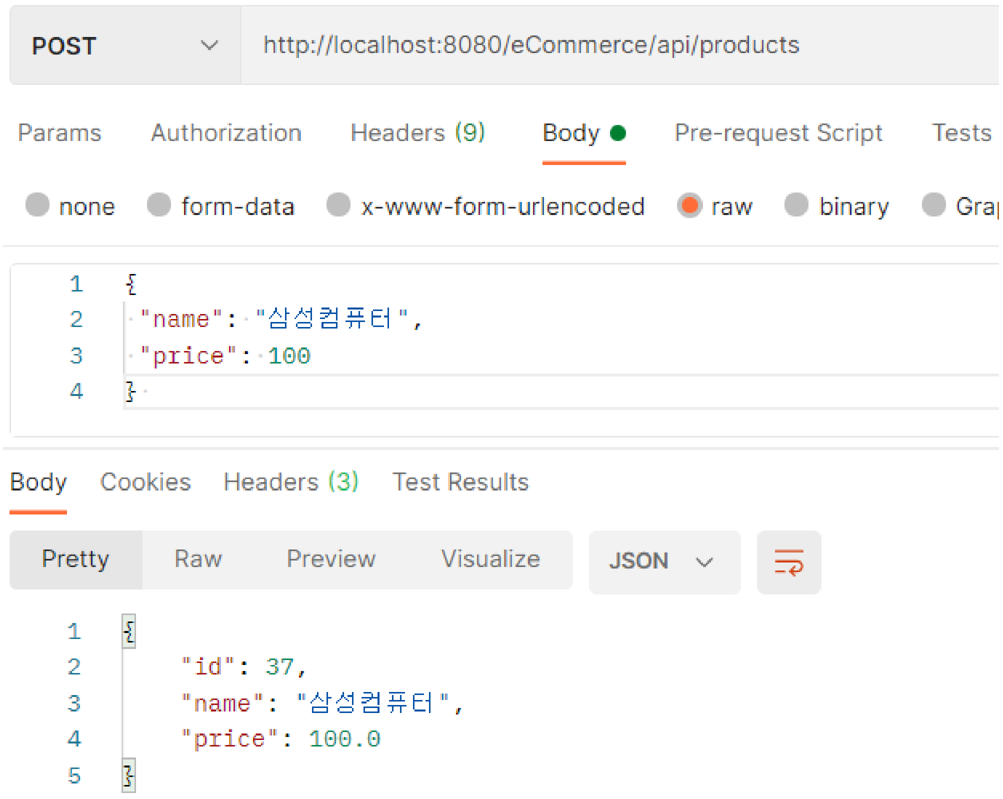


http://localhost:8080/eCommerce/api/products/1

```
{
    "id": 1,
    "name": "TV 4K (32 GB)",
    "price": 223.22
}
```

3) Post, http://localhost:8080/eCommerce/api/products, id 37에 생성되는지 확인

{ "name": "삼성컴퓨터", "price": 100 } // body는 postman에서 raw-json format으로 전송

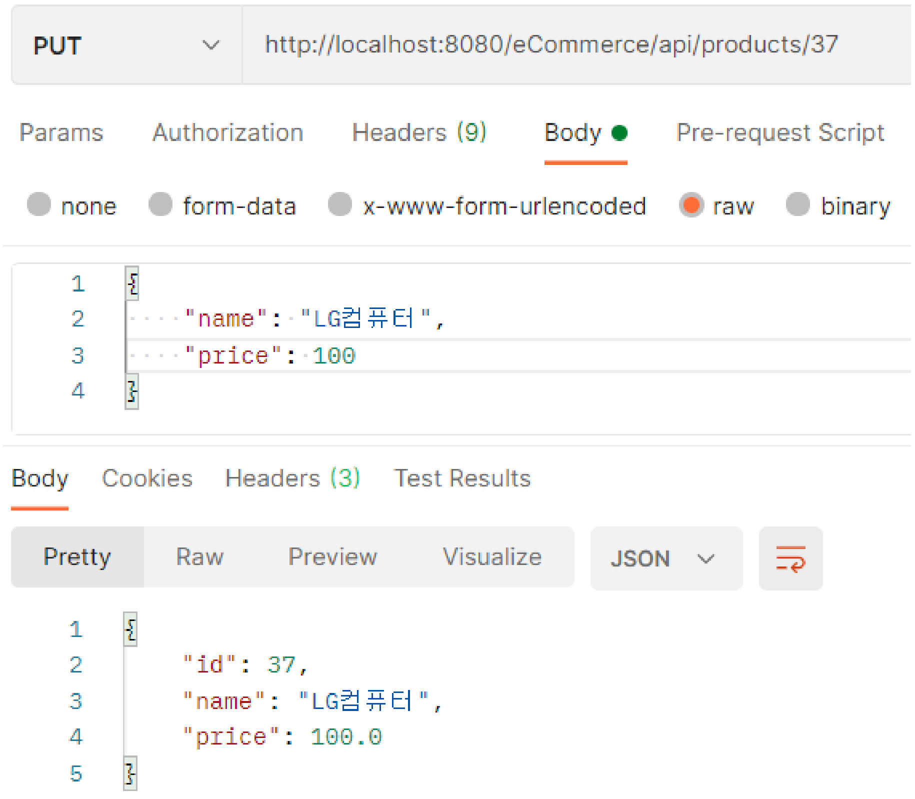


http://localhost:8080/eCommerce/api/products

```
{
 "name": "삼성컴퓨터",
 "price": 100
}
```

```
{
    "id": 37,
    "name": "삼성컴퓨터",
    "price": 100.0
}
```

4) Put, http://localhost:8080/eCommerce/api/products/37 { "name": "LG컴퓨터", "price": 100 }

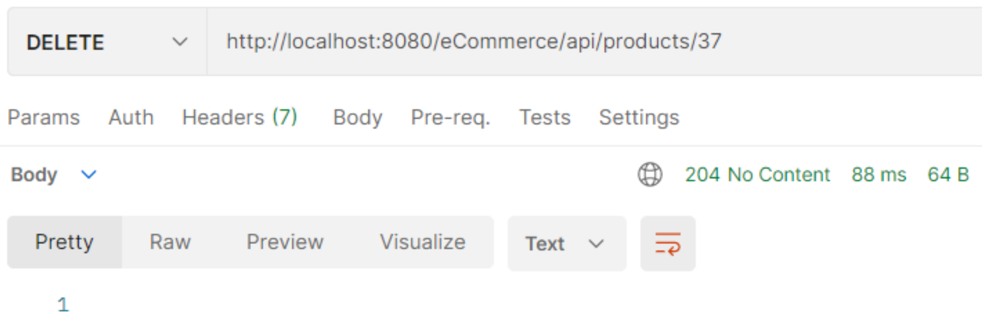


http://localhost:8080/eCommerce/api/products/37

```
{
    "name": "LG컴퓨터",
    "price": 100
}
```

```
{
    "id": 37,
    "name": "LG컴퓨터",
    "price": 100.0
}
```

5) Delete, http://localhost:8080/eCommerce/api/products/37

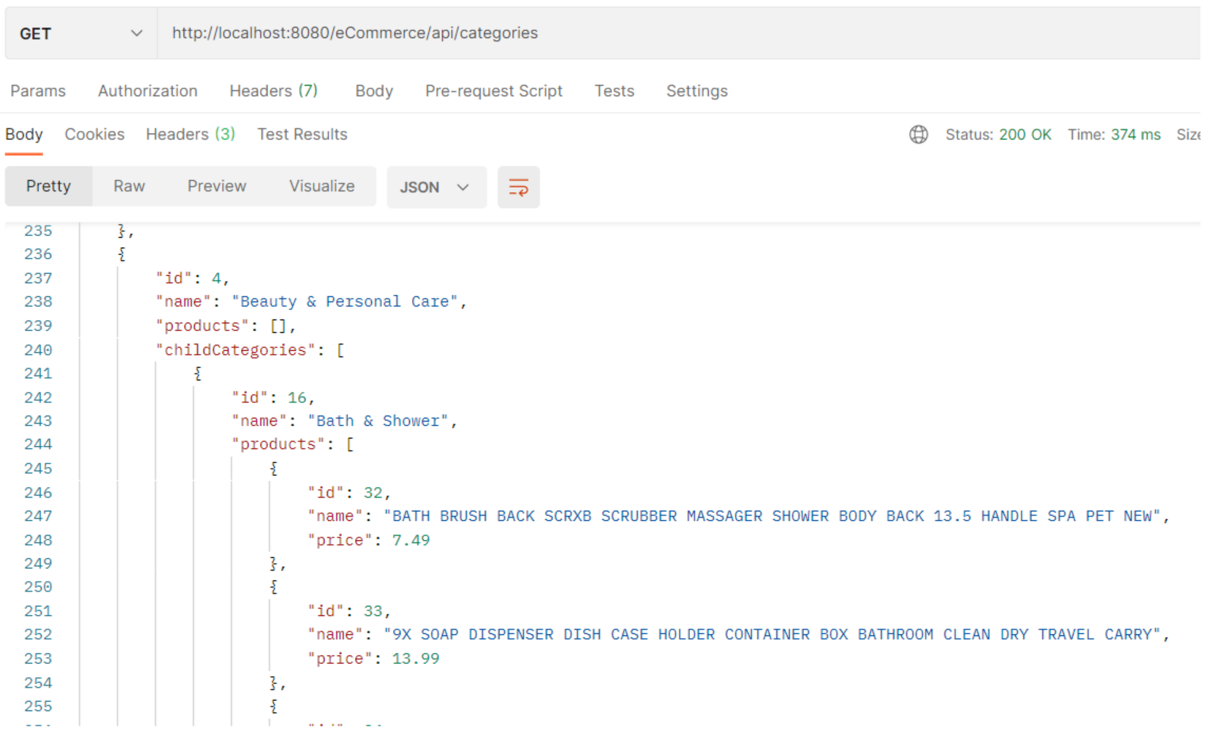


http://localhost:8080/eCommerce/api/products/37

204 No Content 88 ms 64 B

Category:

1-1) Get, http://localhost:8080/eCommerce/api/categories

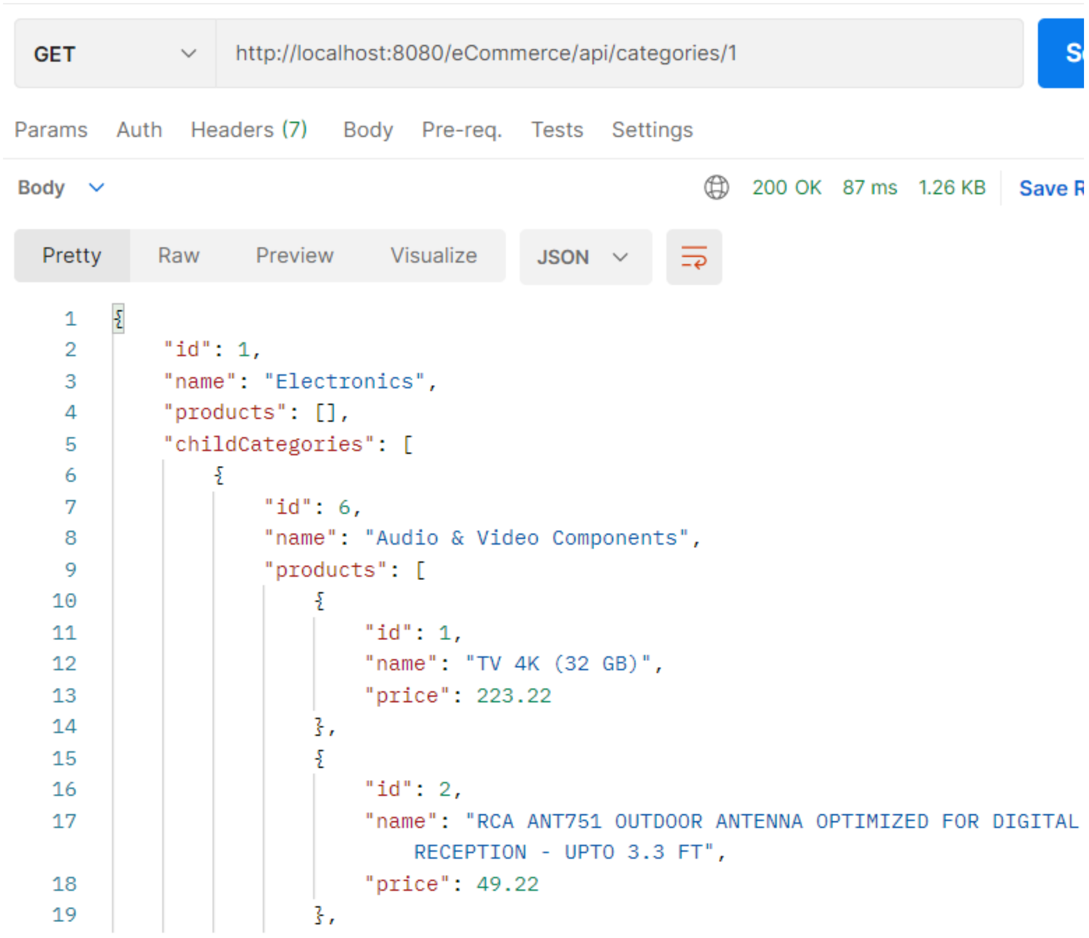


http://localhost:8080/eCommerce/api/categories

Status: 200 OK Time: 374 ms

```
    },
    {
        "id": 4,
        "name": "Beauty & Personal Care",
        "products": [],
        "childCategories": [
            {
                "id": 16,
                "name": "Bath & Shower",
                "products": [
                    {
                        "id": 32,
                        "name": "BATH BRUSH BACK SCRXB SCRUBBER MASSAGER SHOWER BODY BACK 13.5 HANDLE SPA PET NEW",
                        "price": 7.49
                    },
                    {
                        "id": 33,
                        "name": "9X SOAP DISPENSER DISH CASE HOLDER CONTAINER BOX BATHROOM CLEAN DRY TRAVEL CARRY",
                        "price": 13.99
                    },
                    {
```

1-2) Get, http://localhost:8080/eCommerce/api/categories/1



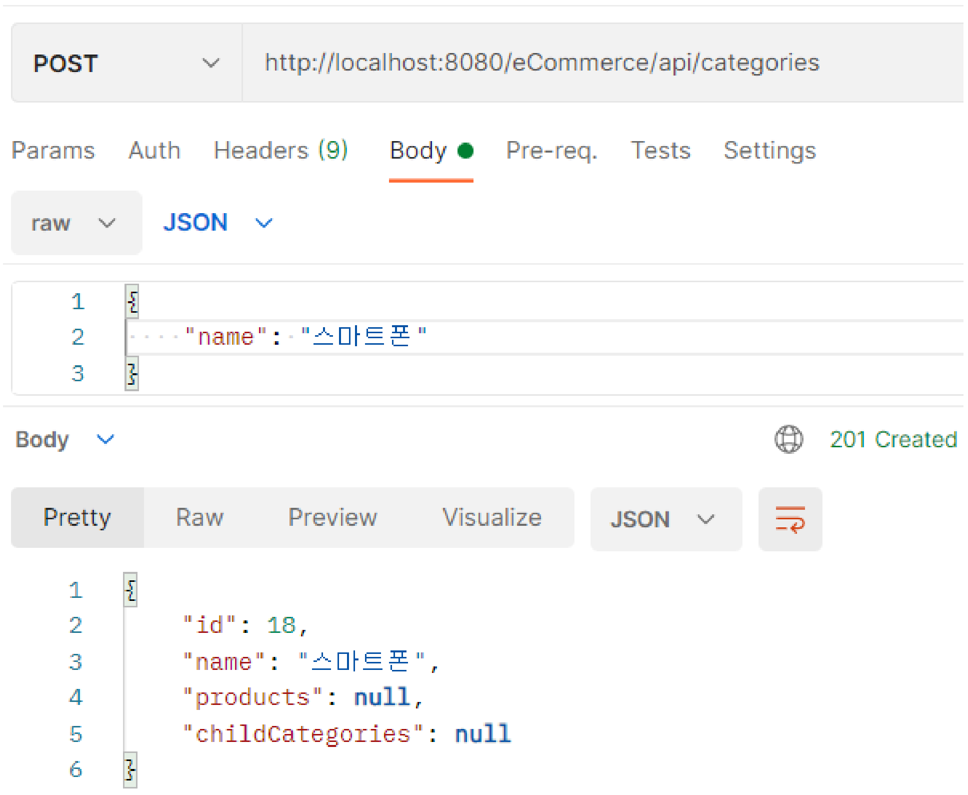
http://localhost:8080/eCommerce/api/categories/1

200 OK 87 ms 1.26 KB

```
{
    "id": 1,
    "name": "Electronics",
    "products": [],
    "childCategories": [
        {
            "id": 6,
            "name": "Audio & Video Components",
            "products": [
                {
                    "id": 1,
                    "name": "TV 4K (32 GB)",
                    "price": 223.22
                },
                {
                    "id": 2,
                    "name": "RCA ANT751 OUTDOOR ANTENNA OPTIMIZED FOR DIGITAL
                        RECEPTION - UPTO 3.3 FT",
                    "price": 49.22
                },
```

1-3) Post, http://localhost:8080/eCommerce/api/categories, id 18 카테고리 생성됨

{ "name": "스마트폰" }



POST http://localhost:8080/eCommerce/api/categories

Params Auth Headers (9) Body ● Pre-req. Tests Settings

raw JSON

```
{
    "name": "스마트폰"
}
```

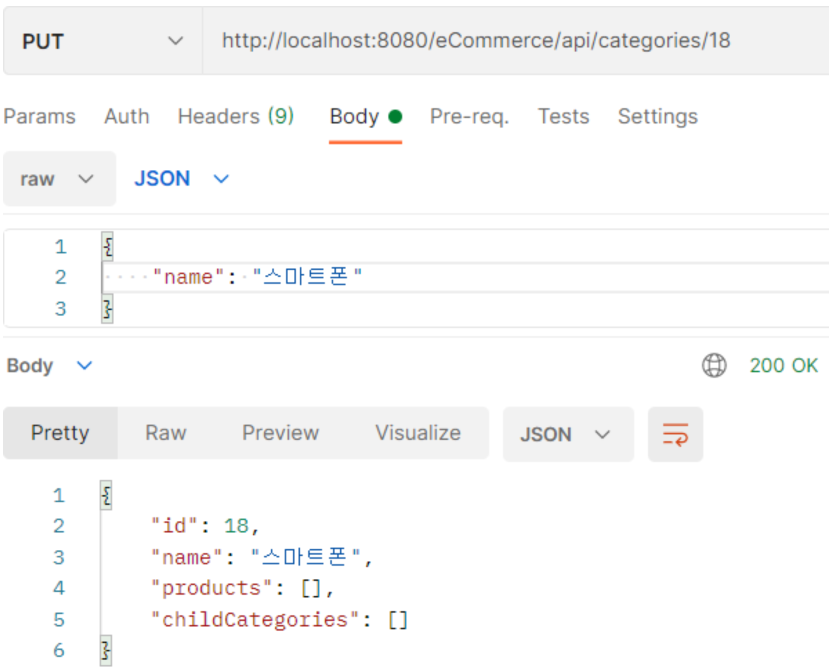
Body 201 Created

Pretty Raw Preview Visualize JSON

```
{
    "id": 18,
    "name": "스마트폰",
    "products": null,
    "childCategories": null
}
```

1-4) Put, http://localhost:8080/eCommerce/api/categories/18

{ "name": "스마트폰" }



PUT http://localhost:8080/eCommerce/api/categories/18

Params Auth Headers (9) Body ● Pre-req. Tests Settings

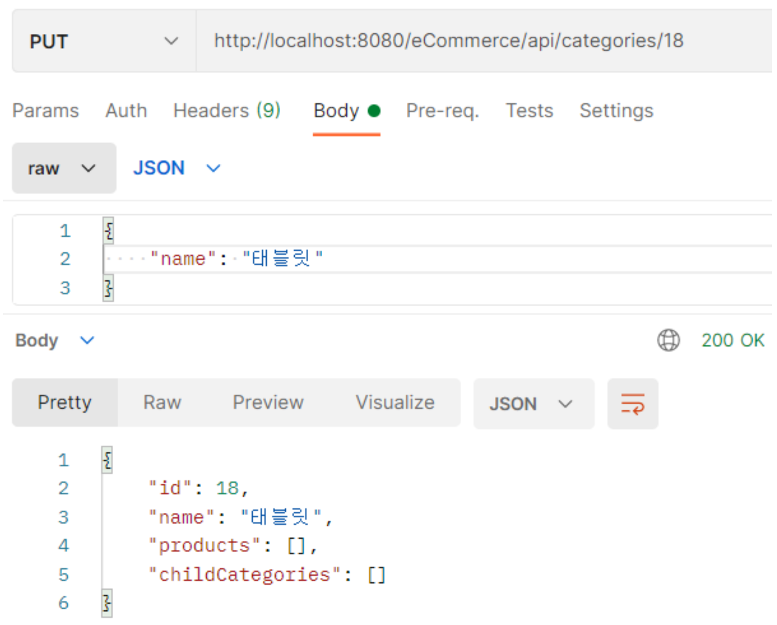
raw JSON

```
{
    "name": "스마트폰"
}
```

Body 200 OK

Pretty Raw Preview Visualize JSON

```
{
    "id": 18,
    "name": "스마트폰",
    "products": [],
    "childCategories": []
}
```


PUT http://localhost:8080/eCommerce/api/categories/18

Params Auth Headers (9) Body Pre-req. Tests Settings

raw JSON

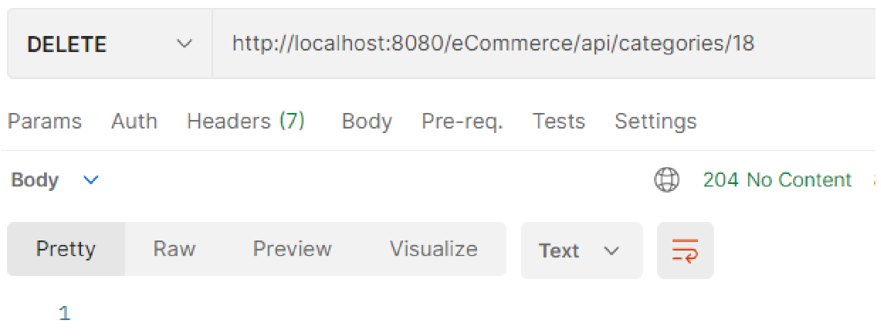
```
{
    "name": "태블릿"
}
```

Body 200 OK

Pretty Raw Preview Visualize JSON

```
{
    "id": 18,
    "name": "태블릿",
    "products": [],
    "childCategories": []
}
```

1-5) Delete, http://localhost:8080/eCommerce/api/categories/18


DELETE http://localhost:8080/eCommerce/api/categories/18

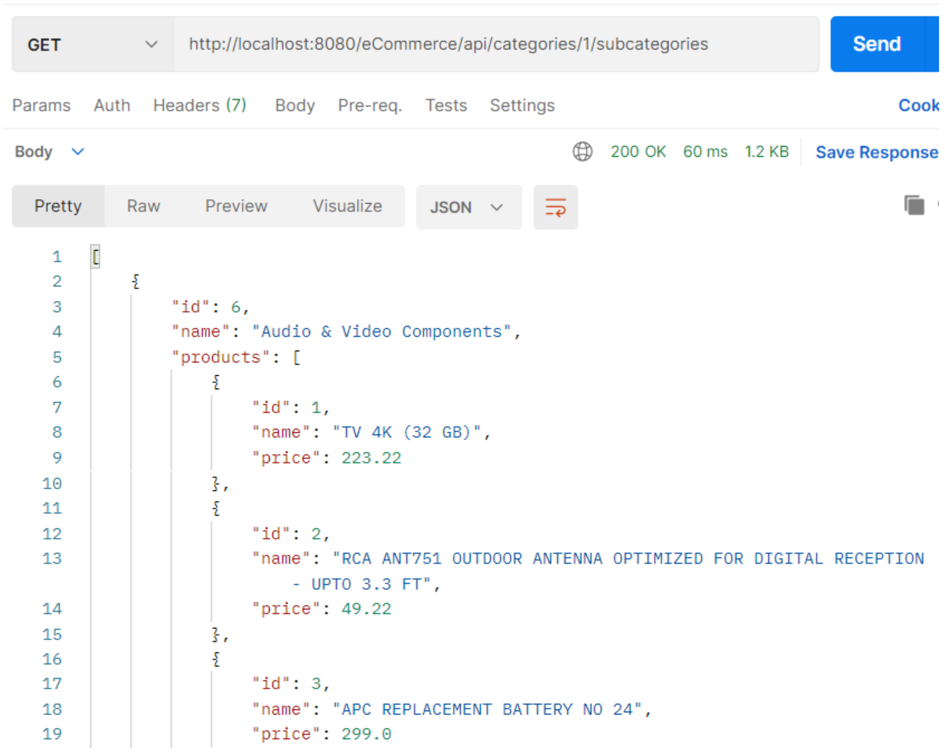
Params Auth Headers (7) Body Pre-req. Tests Settings

Body 204 No Content

Pretty Raw Preview Visualize Text

```
1
```

2-1) Get, http://localhost:8080/eCommerce/api/categories/1/subcategories


GET http://localhost:8080/eCommerce/api/categories/1/subcategories Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cook

Body 200 OK 60 ms 1.2 KB Save Response

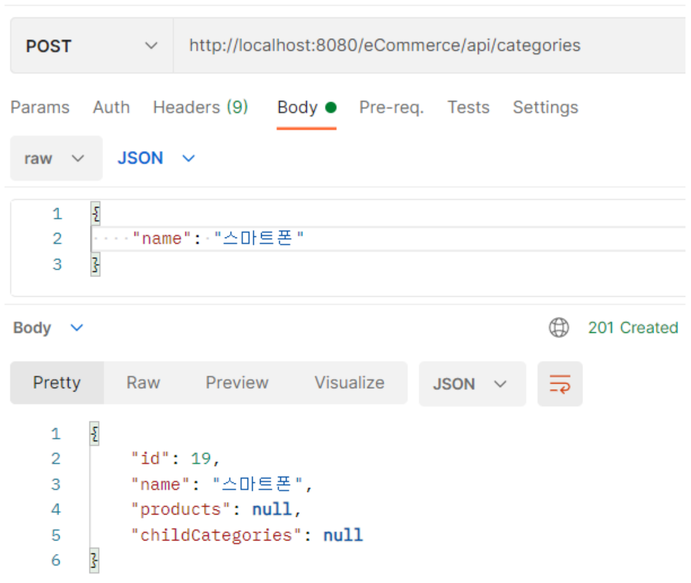
Pretty Raw Preview Visualize JSON

```
[
    {
        "id": 6,
        "name": "Audio & Video Components",
        "products": [
            {
                "id": 1,
                "name": "TV 4K (32 GB)",
                "price": 223.22
            },
            {
                "id": 2,
                "name": "RCA ANT751 OUTDOOR ANTENNA OPTIMIZED FOR DIGITAL RECEPTION
                    - UPTO 3.3 FT",
                "price": 49.22
            },
            {
                "id": 3,
                "name": "APC REPLACEMENT BATTERY NO 24",
                "price": 299.0
```

2-2) subcategory를 생성한 후(id=19), category(id=1)에 연결한다

Post, http://localhost:8080/eCommerce/api/categories

{ "name": "스마트폰" }



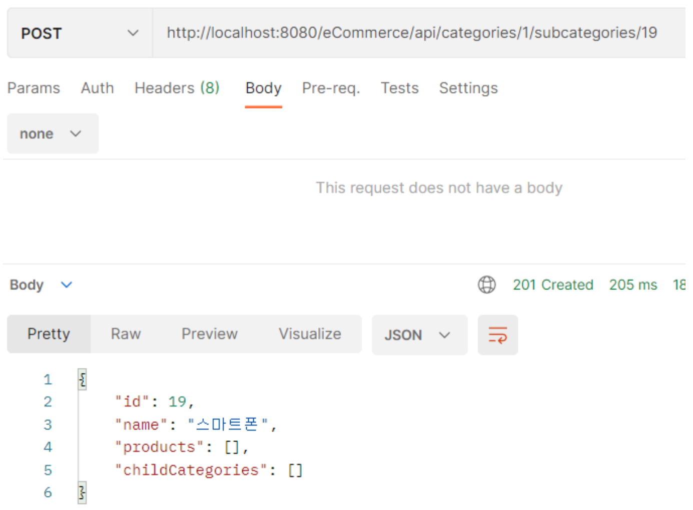
POST http://localhost:8080/eCommerce/api/categories

```
{
    "name": "스마트폰"
}
```

201 Created

```
{
    "id": 19,
    "name": "스마트폰",
    "products": null,
    "childCategories": null
}
```

Post, http://localhost:8080/eCommerce/api/categories/1/subcategories/19

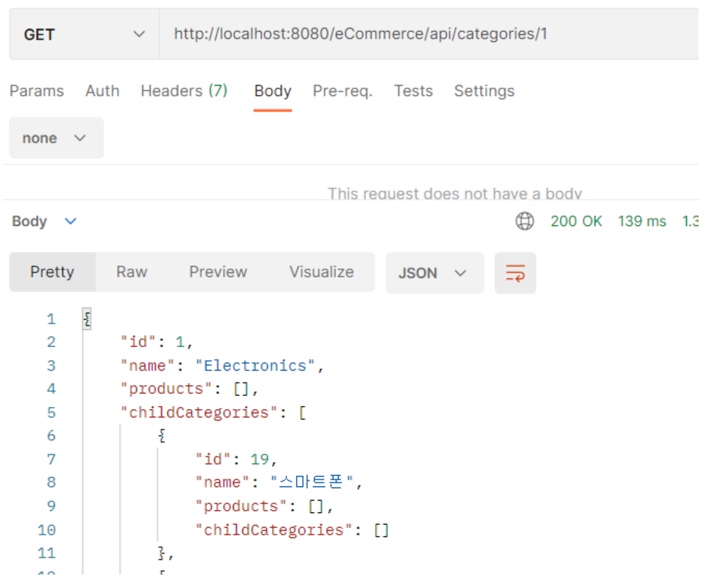


POST http://localhost:8080/eCommerce/api/categories/1/subcategories/19

This request does not have a body

201 Created 205 ms 18

```
{
    "id": 19,
    "name": "스마트폰",
    "products": [],
    "childCategories": []
}
```



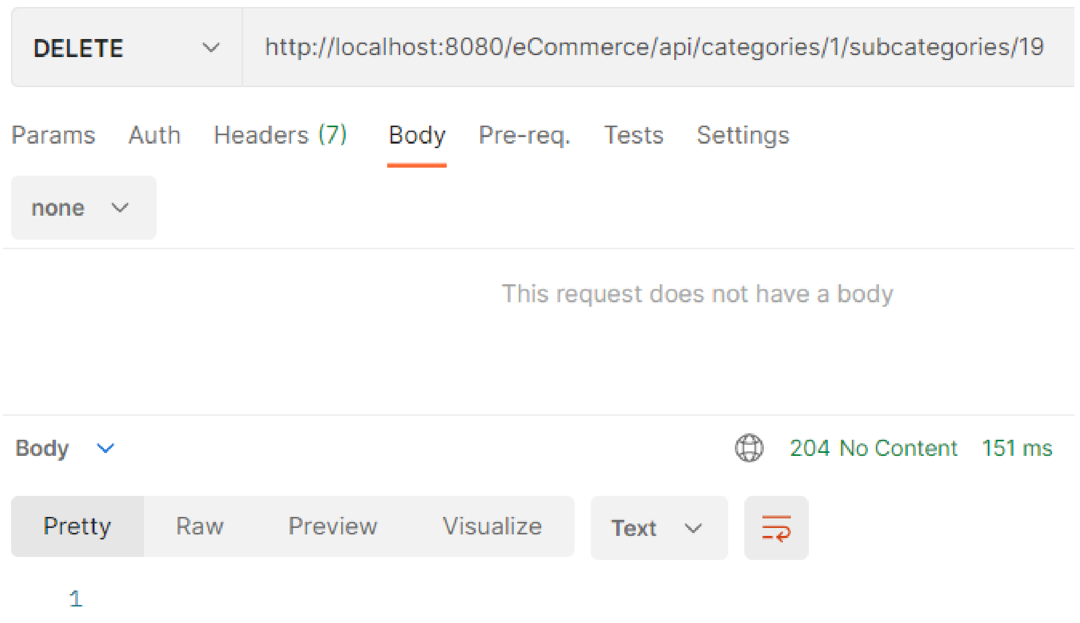
GET http://localhost:8080/eCommerce/api/categories/1

This request does not have a body

200 OK 139 ms 1.3

```
{
    "id": 1,
    "name": "Electronics",
    "products": [],
    "childCategories": [
        {
            "id": 19,
            "name": "스마트폰",
            "products": [],
            "childCategories": []
        },
```

2-3) Delete, http://localhost:8080/eCommerce/api/categories/1/subcategories/19

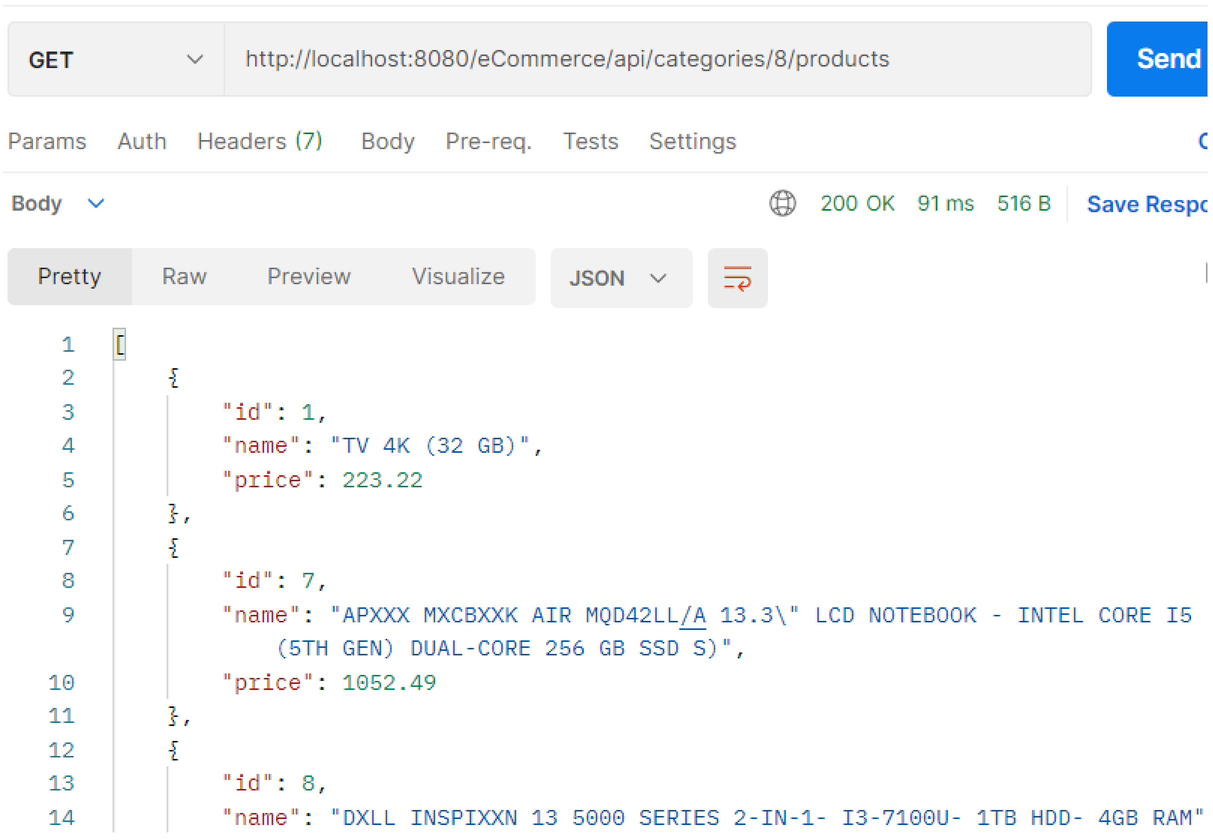


```
http://localhost:8080/eCommerce/api/categories/1/subcategories/19
```

This request does not have a body

204 No Content 151 ms

3-1) Get, http://localhost:8080/eCommerce/api/categories/8/products 먼저 "Computer" 카테고리에 존재하는 Product조회한다



```
http://localhost:8080/eCommerce/api/categories/8/products
```

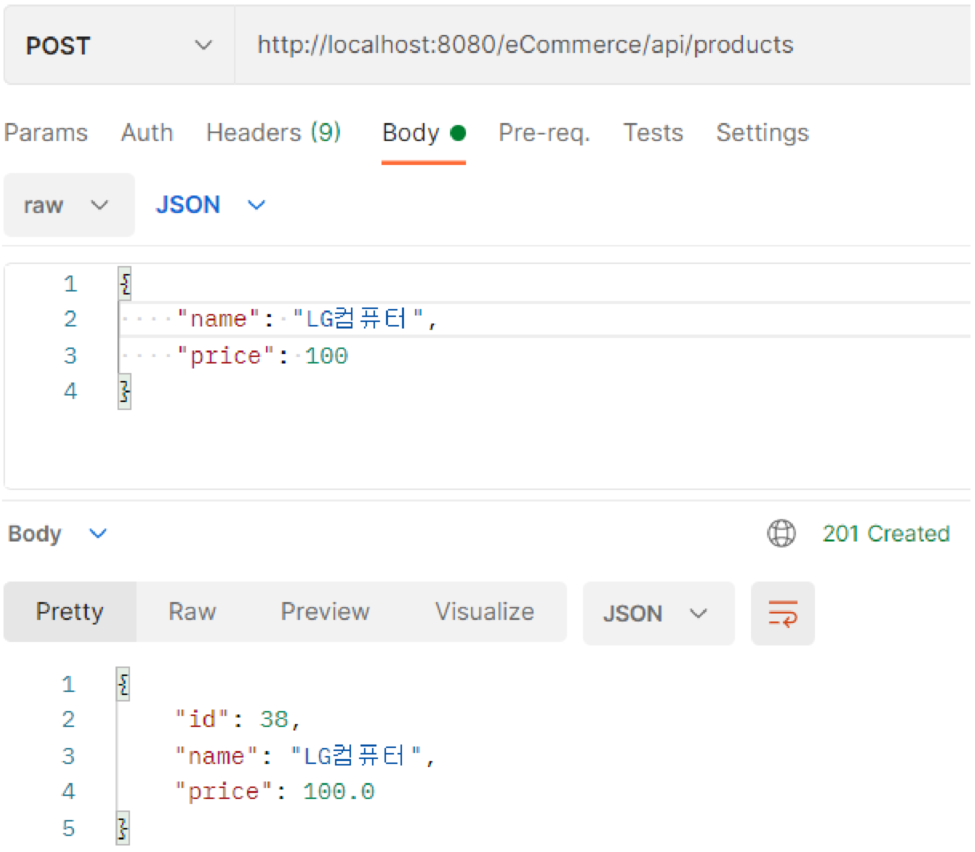
200 OK 91 ms 516 B

```
[
    {
        "id": 1,
        "name": "TV 4K (32 GB)",
        "price": 223.22
    },
    {
        "id": 7,
        "name": "APXXX MXCBXXK AIR MQD42LL/A 13.3\" LCD NOTEBOOK - INTEL CORE I5
            (5TH GEN) DUAL-CORE 256 GB SSD S)",
        "price": 1052.49
    },
    {
        "id": 8,
        "name": "DXLL INSPIXXN 13 5000 SERIES 2-IN-1- I3-7100U- 1TB HDD- 4GB RAM"
```

3-2) Product(id=38)를 생성한 후, “Computer” 카테고리에 저장한다.

Post, http://localhost:8080/eCommerce/api/products

{ "name": "LG컴퓨터", "price": 100 }



POST http://localhost:8080/eCommerce/api/products

Params Auth Headers (9) Body ● Pre-req. Tests Settings

raw JSON

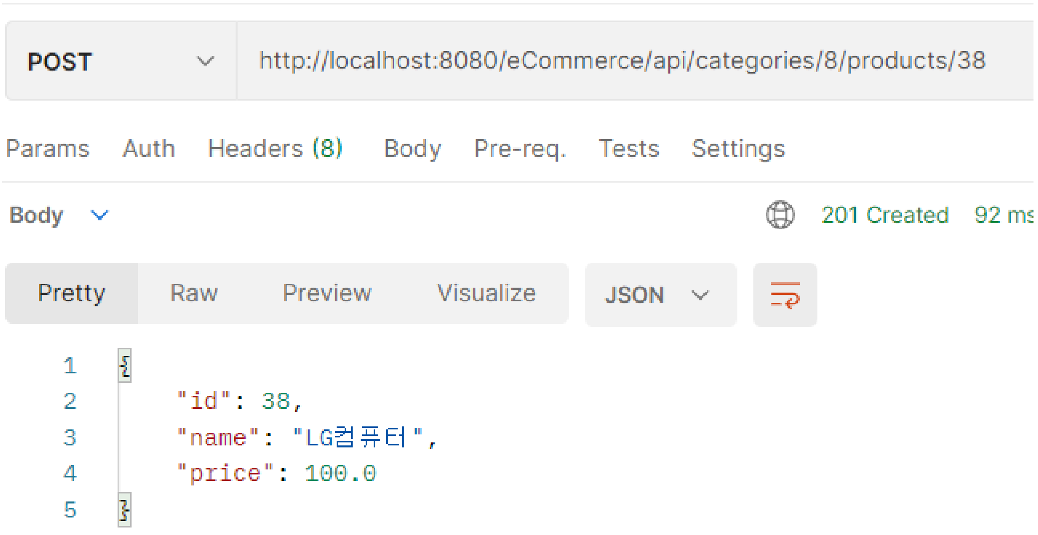
```
{
    "name": "LG컴퓨터",
    "price": 100
}
```

Body 201 Created

Pretty Raw Preview Visualize JSON

```
{
    "id": 38,
    "name": "LG컴퓨터",
    "price": 100.0
}
```

Post, http://localhost:8080/eCommerce/api/categories/8/products/38



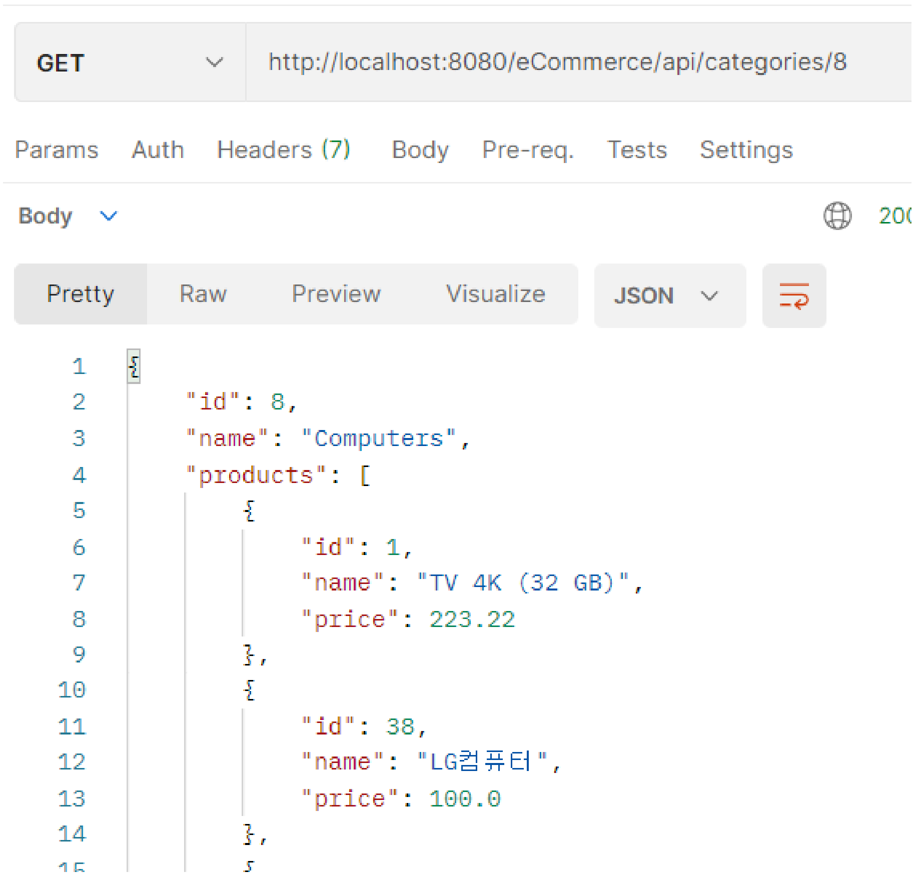
POST http://localhost:8080/eCommerce/api/categories/8/products/38

Params Auth Headers (8) Body Pre-req. Tests Settings

Body 201 Created 92 ms

Pretty Raw Preview Visualize JSON

```
{
    "id": 38,
    "name": "LG컴퓨터",
    "price": 100.0
}
```

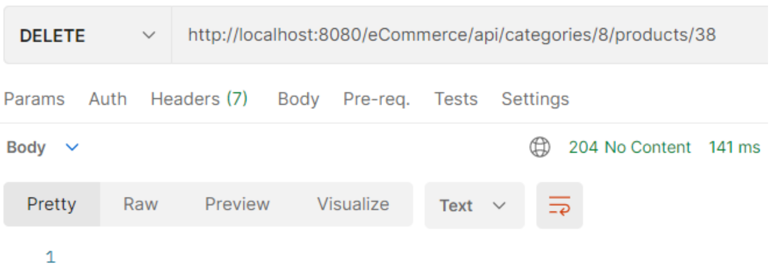


GET http://localhost:8080/eCommerce/api/categories/8

200

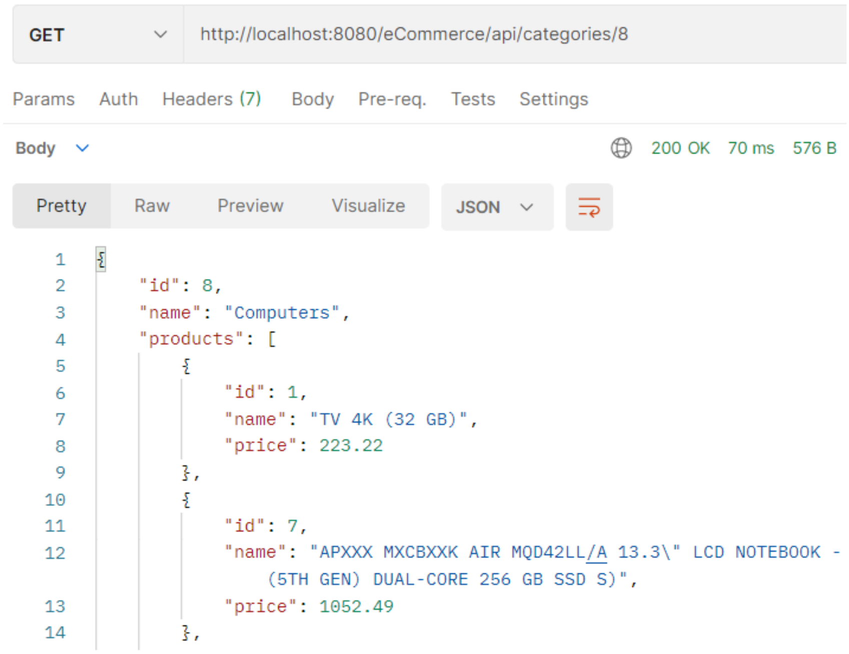
```
{
    "id": 8,
    "name": "Computers",
    "products": [
        {
            "id": 1,
            "name": "TV 4K (32 GB)",
            "price": 223.22
        },
        {
            "id": 38,
            "name": "LG컴퓨터",
            "price": 100.0
        },
```

3-3) Delete, http://localhost:8080/eCommerce/api/categories/8/products/38



DELETE http://localhost:8080/eCommerce/api/categories/8/products/38

204 No Content 141 ms



GET http://localhost:8080/eCommerce/api/categories/8

200 OK 70 ms 576 B

```
{
    "id": 8,
    "name": "Computers",
    "products": [
        {
            "id": 1,
            "name": "TV 4K (32 GB)",
            "price": 223.22
        },
        {
            "id": 7,
            "name": "APXXX MXCBXXK AIR MQD42LL/A 13.3\" LCD NOTEBOOK -
                (5TH GEN) DUAL-CORE 256 GB SSD S)",
            "price": 1052.49
        },
```