Create a project to demonstrate microservices with Spring Boot.

microservice-1

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>3.1.0</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>spring-boot-building-rest-api-server</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>spring-boot-building-rest-api-server</name>
	<description>Demo project for Spring Boot</description>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>

		<dependency>
			<groupId>com.h2database</groupId>
			<artifactId>h2</artifactId>
			<scope>runtime</scope>
		</dependency>

		<!--

	https://mvnrepository.com/artifact/jakarta.servlet.jsp.jstl/jakarta.ser
vlet.jsp.jstl-api -->
		<dependency>
			<groupId>jakarta.servlet.jsp.jstl</groupId>
			<artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
			<version>3.0.0</version>
		</dependency>
```

```
            <!-- JSP support in Spring -->
            <dependency>
                  <groupId>org.apache.tomcat.embed</groupId>
                  <artifactId>tomcat-embed-jasper</artifactId>
                  <scope>provided</scope>
            </dependency>

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

      <build>
            <plugins>
                  <plugin>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-maven-plugin</artifactId>
                  </plugin>
            </plugins>
      </build>

</project>
```

Application.properties

```
#JSP view resolver support
server.port=10001
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp

#Database H2
spring.datasource.url=jdbc:h2:C:/temp1/testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.h2.console.enabled=true
```

Eproduct.java

```
package com.ecommerce;


import java.math.BigDecimal;
import java.util.Date;
```

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;

@NamedQuery(name ="EProduct.findAllWherePriceIs1000", query="SELECT p from EProduct p where
p.price=1000")
@Entity
@Table(name="eproduct")
public class EProduct {

                    @Id
                    @GeneratedValue(strategy = GenerationType.IDENTITY)
                    @Column(name="id")
                    private long ID;

    private String name;
    private BigDecimal price;

    @Column(name="date_added")
    private Date dateAdded;

    public EProduct() {
    }

    public long getID() {return this.ID; }
    public String getName() { return this.name;}
    public BigDecimal getPrice() { return this.price;}
    public Date getDateAdded() { return this.dateAdded;}

    public void setID(long id) { this.ID = id;}
    public void setName(String name) { this.name = name;}
    public void setPrice(BigDecimal price) { this.price = price;}
    public void setDateAdded(Date date) { this.dateAdded = date;}
}
```

EproductRepositry

```
package com.ecommerce;

import java.util.List;

import org.springframework.data.jpa.repository.*;
```

```
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

@Repository
public interface EProductRepositry extends JpaRepository<EProduct, Integer>, JpaSpecificationExecutor {

	// Derived queries
	List<EProduct> findAllByName(String name);

	List<EProduct> findAllByPrice(float price);

	List<EProduct> findAllByPriceGreaterThan(float price);

	// JPQL queries
	@Query("SELECT p FROM EProduct p WHERE p.name LIKE %:name%")
	List<EProduct> findAllByHavingNameAnywhere(@Param("name") String name);

	@Query("SELECT p FROM EProduct p WHERE p.price > :minPrice and p.price < :maxPrice")
	List<EProduct> findAllWherePriceIsInBetween(float minPrice,float maxPrice);

	// SQL queries
	@Query(value="SELECT * FROM eproduct WHERE name LIKE %:name%", nativeQuery=true)
	List<EProduct> findAllByHavingNameAnywhereUsingSQL(String name);

	// Named Queries example
	List<EProduct> findAllWherePriceIs1000();
}
```

MainController

```
package com.ecommerce;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/main")
public class MainRestController {

	@GetMapping(path = "/apple", produces = "application/json")
	public ResponseEntity<Apple> displayApply() {
		Apple a = new Apple();
		a.name = "Shimla";
```

```
		a.weight = 10;

		return new ResponseEntity<Apple>(a, HttpStatus.OK);

	}

}

class Apple {

	public String name;
	public int weight;

}
```

ProductRestController

```
package com.ecommerce;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping("/product")
public class ProductRestController {

	@Autowired
	EProductRepositry eProductRepo;

	// List all the products
	@GetMapping(path="/list", produces = "application/json")
	public List<EProduct> listProducts(){
		List<EProduct> products = eProductRepo.findAll();

		return products;
	}
```

```
	// Adding a new product
	@PostMapping(path="/add", consumes="application/json" , produces = "application/json")
	public EProduct addProduct(@RequestBody EProduct eProduct){
		eProduct = eProductRepo.save(eProduct);
		return eProduct;
	}

	// Finding a single product and fetching its details
	@GetMapping(path="/details/{id}", produces = "application/json")
	public Object showProduct(@PathVariable("id") int id){

		Optional<EProduct> productFromRepo = eProductRepo.findById(id);

		if (productFromRepo.isPresent()) {
			EProduct product = productFromRepo.get();
			return product;
		}else {
			return "Product with id = "+ id + " not found";
		}
	}

	//Delete a Product
	@GetMapping(path="/delete/{id}", produces = "application/json")
	public Object deleteProduct(@PathVariable("id") int id){

		Optional<EProduct> productFromRepo = eProductRepo.findById(id);

		if (productFromRepo.isPresent()) {
			eProductRepo.deleteById(id);
			return "Product with id = "+ id + " found and deleted";
		}else {
			return "Product with id = "+ id + " not found";
		}
	}



}
```

SpringBootBuildingRestApiServerApplication

```
package com.ecommerce;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@ComponentScan({"com.ecommerce","com.ecommerce.controllers","com.ecommerce.entity",
"com.ecommerce.repositries" })
@EnableJpaRepositories
@SpringBootApplication
public class SpringBootBuildingRestApiServerApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootBuildingRestApiServerApplication.class, args);
	}

}
```

microservice-2

SpringBootConsumeRestWebserviceDemoApplication

```
package com.ecommerce;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringBootConsumeRestWebserviceDemoApplication {

	public static void main(String[] args) {
		SpringApplication.run(SpringBootConsumeRestWebserviceDemoApplication.class, args);
	}

}
```

RestServiceConsumerController

```
package com.ecommerce;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.client.RestTemplate;

import com.ecommerce.entity.EProduct;
```

```
@Controller
public class RestServiceConsumerController {

	@GetMapping("/abc/{id}")
	public String productdetails(@PathVariable("id") int id, Model model) {

			RestTemplate restTemplate = new RestTemplate();
			// Get a product from the first micro service.
			EProduct product =
restTemplate.getForObject("http://localhost:8080/product/details/"+id, EProduct.class);

			model.addAttribute("product", product);

			return "product-details"; // product-details.jsp
	}

}
```

Eproduct

```
package com.ecommerce.entity;

import java.math.BigDecimal;
import java.sql.Date;

public class EProduct {



			private long ID;
	private String name;
	private BigDecimal price;
	private Date dateAdded;

	public EProduct() {
	}

	public long getID() {return this.ID; }
	public String getName() { return this.name;}
	public BigDecimal getPrice() { return this.price;}
	public Date getDateAdded() { return this.dateAdded;}

	public void setID(long id) { this.ID = id;}
	public void setName(String name) { this.name = name;}
	public void setPrice(BigDecimal price) { this.price = price;}
	public void setDateAdded(Date date) { this.dateAdded = date;}
}
```

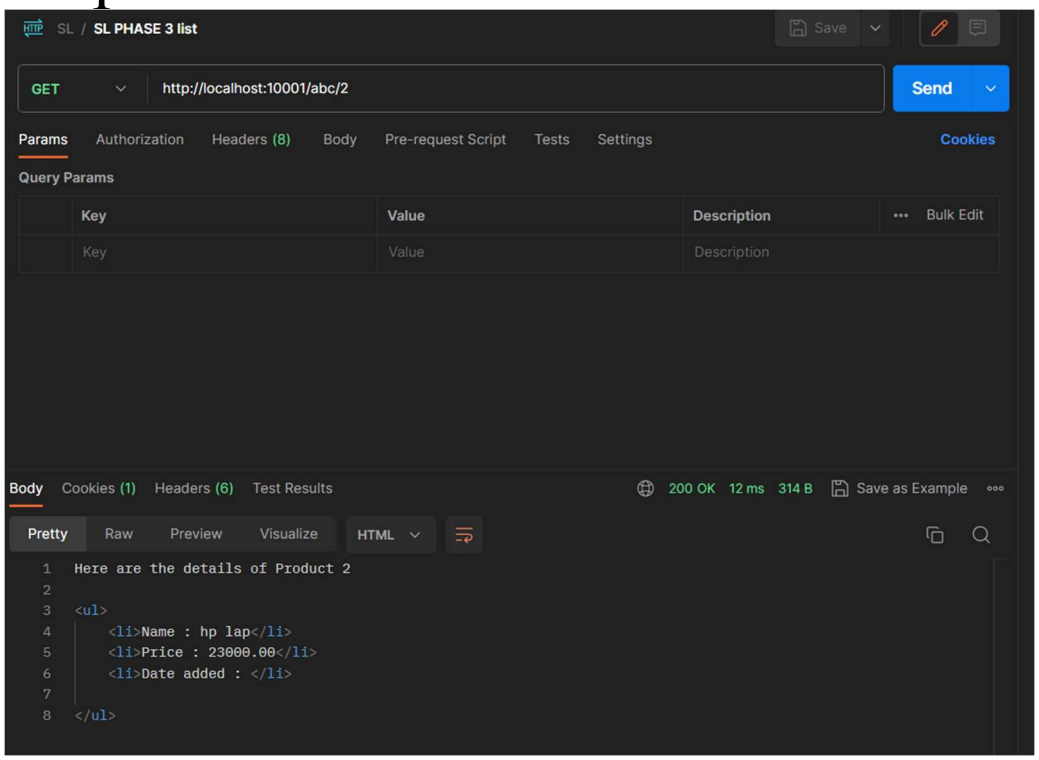
Product-details.jsp

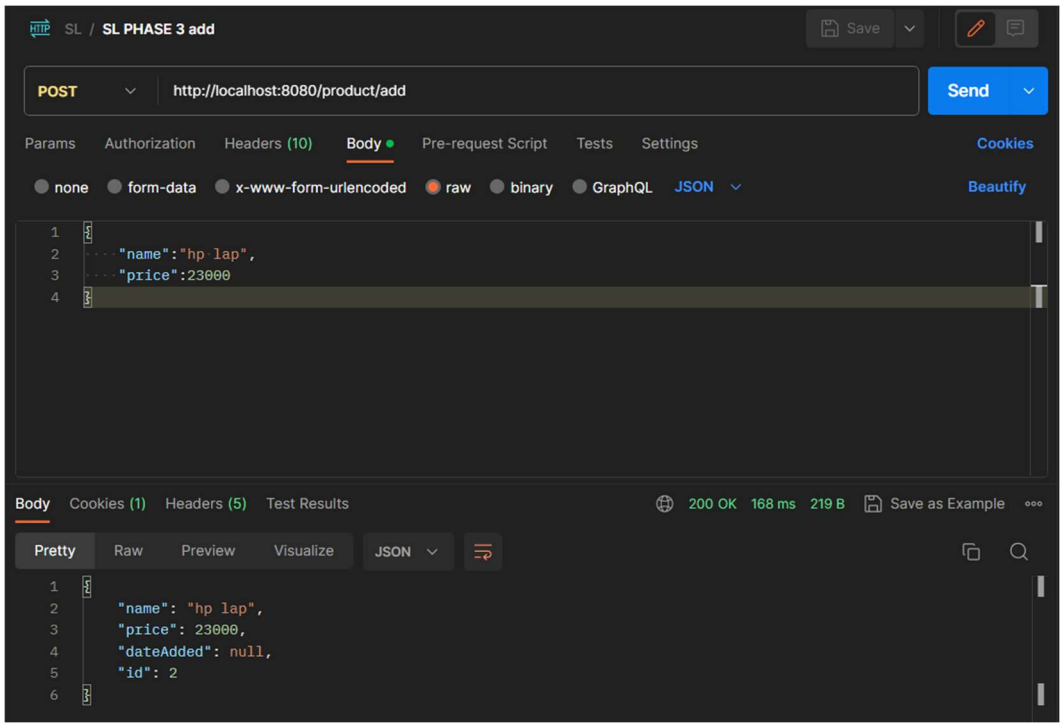
```
Here are the details of Product ${product.ID}

<ul>
<li>Name : ${product.name}</li>
<li>Price : ${product.price}</li>
<li>Date added : ${product.dateAdded}</li>

</ul>
```

Application.properties

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
server.port=10001
```

output


SL / SL PHASE 3 list
Save
GET
http://localhost:10001/abc/2
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Body
Cookies (1)
Headers (6)
Test Results
200 OK
12 ms
314 B
Save as Example
Pretty
Raw
Preview
Visualize
HTML
1 Here are the details of Product 2
2
3
4 Name : hp lap
5 Price : 23000.00
6 Date added :
7
8


SL / SL PHASE 3 add
Save
POST
http://localhost:8080/product/add
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "name":"hp lap",
3 "price":23000
4 }
Body
Cookies (1)
Headers (5)
Test Results
200 OK
168 ms
219 B
Save as Example
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "name": "hp lap",
3 "price": 23000,
4 "dateAdded": null,
5 "id": 2
6 }