SOURCE CODE

com.ecommerce. 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.controllers","com.ecommerce.entity", "com.ecommerce.repositories" })
@EnableJpaRepositories
@SpringBootApplication
public class SpringBootBuildingRestApiServerApplication {
	public static void main(String[] args) {
		SpringApplication.run(SpringBootBuildingRestApiServerApplication.class, args);
	}
}
```

MainRestController.java

```
package com.ecommerce.controllers;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
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.controllers;

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;

import com.ecommerce.entity.EProduct;
import com.ecommerce.repositories.EProductRepository;


@RestController
@RequestMapping("/product")
```

```
public class ProductRestController {

	@Autowired
	EProductRepository 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";
		}
	}
}
```

EProduct.java

```
package com.ecommerce.entity;
import java.math.BigDecimal;
import java.sql.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;}
}
```

EProductRepository.java

```
package com.ecommerce.repositories;
```

```
import java.util.List;

import org.springframework.data.jpa.repository.*;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;

import com.ecommerce.entity.EProduct;

@Repository
public interface EProductRepository 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();
}
```