day12_SpringTest

(pom.xml):

```xml
<dependencies>
   <!-- Spring Boot Starter Web -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
   </dependency>
   <!-- Spring Data JPA -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
   </dependency>
   <!-- MySQL Driver -->
   <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
   </dependency>
   <!-- Lombok (Optional) -->
   <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
   </dependency>
   <!-- Testing (JUnit + Mockito) -->
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
```

```
        <scope>test</scope>
    </dependency>
</dependencies>
```

Product.java

```
package com.example.productorder.entity;


import jakarta.persistence.*;
import lombok.Data;


@Data
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long productId;


    private String name;
    private double price;
    private int availableQuantity;
}
```

Order.java

```
package com.example.productorder.entity;


import jakarta.persistence.*;
import lombok.Data;
import java.time.LocalDateTime;
```

day12_SpringTest

```
@Data
@Entity
public class Order {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long orderId;

   @ManyToOne
   @JoinColumn(name = "product_id")
   private Product product;

   private LocalDateTime orderDate;
   private int quantityOrdered;
}
```

ProductRepository.java

```
package com.example.productorder.repository;

import com.example.productorder.entity.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product, Long> {}
```

OrderRepository.java

```
package com.example.productorder.repository;

import com.example.productorder.entity.Order;
import org.springframework.data.jpa.repository.JpaRepository;

public interface OrderRepository extends JpaRepository<Order, Long> {}
```

day12_SpringTest

ProductService.java

```
package com.example.productorder.service;


import com.example.productorder.entity.Product;
import com.example.productorder.repository.ProductRepository;
import org.springframework.stereotype.Service;
import java.util.List;


@Service
public class ProductService {
    private final ProductRepository productRepo;


    public ProductService(ProductRepository productRepo) {
        this.productRepo = productRepo;
    }


    public Product addProduct(Product product) {
        return productRepo.save(product);
    }


    public List<Product> getAllProducts() {
        return productRepo.findAll();
    }


    public Product updateStock(Long productId, int quantity) {
        Product product = productRepo.findById(productId).orElseThrow();
        product.setAvailableQuantity(product.getAvailableQuantity() + quantity);
        return productRepo.save(product);
    }
}
```

OrderService.java

```
package com.example.productorder.service;


import com.example.productorder.entity.Order;
import com.example.productorder.entity.Product;
import com.example.productorder.repository.OrderRepository;
import com.example.productorder.repository.ProductRepository;
import org.springframework.stereotype.Service;
import java.time.LocalDateTime;


@Service
public class OrderService {
    private final OrderRepository orderRepo;
    private final ProductRepository productRepo;


    public OrderService(OrderRepository orderRepo, ProductRepository productRepo) {
        this.orderRepo = orderRepo;
        this.productRepo = productRepo;
    }


    public Order placeOrder(Long productId, int quantity) {
        Product product = productRepo.findById(productId).orElseThrow();


        if (product.getAvailableQuantity() < quantity) {
            throw new RuntimeException("Insufficient stock!");
        }

        product.setAvailableQuantity(product.getAvailableQuantity() - quantity);
        productRepo.save(product);

        Order order = new Order();
        order.setProduct(product);
        order.setQuantityOrdered(quantity);
        order.setOrderDate(LocalDateTime.now());

        return orderRepo.save(order);
    }
}
```

ProductController.java

```
package com.example.productorder.controller;

import com.example.productorder.entity.Product;
import com.example.productorder.service.ProductService;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {
    private final ProductService productService;
    public ProductController(ProductService productService) {
        this.productService = productService;
    }

    @PostMapping
    public Product addProduct(@RequestBody Product product) {
        return productService.addProduct(product);
    }

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }

    @PutMapping("/{id}/stock")
    public Product updateStock(@PathVariable Long id, @RequestParam int qty) {
        return productService.updateStock(id, qty);
    }
}
```

OrderController.java

```
package com.example.productorder.controller;

import com.example.productorder.entity.Order;
import com.example.productorder.service.OrderService;
import org.springframework.web.bind.annotation.*;

@RestController
```

day12_SpringTest

```
@RequestMapping("/api/orders")
public class OrderController {
    private final OrderService orderService;


    public OrderController(OrderService orderService) {
        this.orderService = orderService;
    }


    @PostMapping
    public Order placeOrder(@RequestParam Long productId, @RequestParam int quantity) {
        return orderService.placeOrder(productId, quantity);
    }
}
```

ProductServiceTest.java

```
package com.example.productorder.service;


import com.example.productorder.entity.Product;
import com.example.productorder.repository.ProductRepository;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.springframework.boot.test.context.SpringBootTest;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;


@SpringBootTest
```

day12_SpringTest

```
public class ProductServiceTest {
    @Mock
    private ProductRepository productRepo;

    @InjectMocks
    private ProductService productService;

    @Test
    public void testAddProduct() {
        Product product = new Product();
        product.setName("Laptop");
        product.setPrice(999.99);
        product.setAvailableQuantity(10);

        when(productRepo.save(product)).thenReturn(product);

        Product savedProduct = productService.addProduct(product);
        assertEquals("Laptop", savedProduct.getName());
    }

    @Test
    public void testUpdateStock() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(5);

        when(productRepo.findById(1L)).thenReturn(java.util.Optional.of(product));
        when(productRepo.save(product)).thenReturn(product);
        Product updatedProduct = productService.updateStock(1L, 3);
        assertEquals(8, updatedProduct.getAvailableQuantity());
    }
}
```

OrderServiceTest.java

```
package com.example.productorder.service;

import com.example.productorder.entity.Order;
import com.example.productorder.entity.Product;
import com.example.productorder.repository.OrderRepository;
import com.example.productorder.repository.ProductRepository;
import org.junit.jupiter.api.Test;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.springframework.boot.test.context.SpringBootTest;
import java.time.LocalDateTime;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;

@SpringBootTest
public class OrderServiceTest {
    @Mock
    private OrderRepository orderRepo;

    @Mock
    private ProductRepository productRepo;
```

day12_SpringTest

```
    @InjectMocks
    private OrderService orderService;

    @Test
    public void testPlaceOrder_Success() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(10);

        when(productRepo.findById(1L)).thenReturn(java.util.Optional.of(product));
        when(orderRepo.save(any(Order.class))).thenAnswer(invocation -> 
invocation.getArgument(0));

        Order order = orderService.placeOrder(1L, 3);
        assertNotNull(order);
        assertEquals(3, order.getQuantityOrdered());
    }

    @Test
    public void testPlaceOrder_InsufficientStock() {
        Product product = new Product();
        product.setProductId(1L);
        product.setAvailableQuantity(2);

        when(productRepo.findById(1L)).thenReturn(java.util.Optional.of(product));

        assertThrows(RuntimeException.class, () -> orderService.placeOrder(1L, 3));
    }
}
```