1.Write a program to demonstrate standard tests.

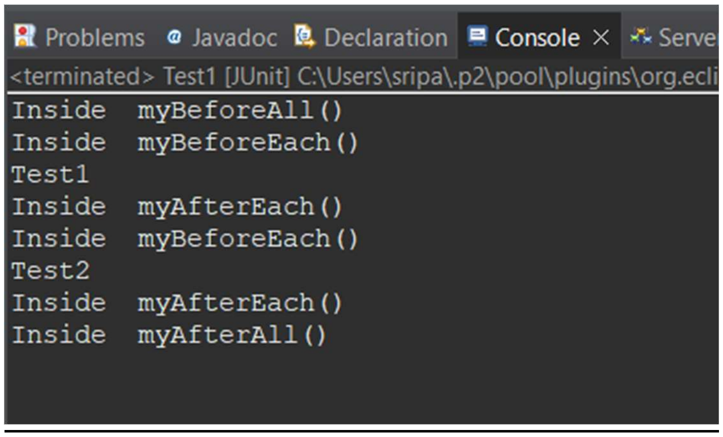
Test1.java

```
package com.ecommerce.junit;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

@DisplayName("This is my first Test class")
class Test1 {

      @BeforeAll
      static void myBeforeAll() {
            // Create a Connectn Obj here that will used in the Test cases
Test1,  Test2,....
            System.out.println("Inside  myBeforeAll()");
      }

      @AfterAll
      static void myAfterAll() {
            // Close the Connectn Obj here.
            System.out.println("Inside  myAfterAll()");
      }

      @BeforeEach
      void myBeforeEach() {
            // Create a Statement Obj here that will used in the Test cases
Test1,  Test2,....
            System.out.println("Inside  myBeforeEach()");
      }

      @AfterEach
      void myAfterEach() {
            // close Statement Obj here so that it will release system
resources.
            System.out.println("Inside  myAfterEach()");
      }


      @Test
      @DisplayName("This is my first Test case Test 1")
      void test1() {
            // Test a JDBC SQL Query select * for eproduct
            System.out.println("Test1 ");
      }

      @Test
      @DisplayName("This is my second Test case Test 2")
      void test2() {
            // Test a JDBC SQL Query for eproduct where price>1000
            System.out.println("Test2 ");
      }

}
```

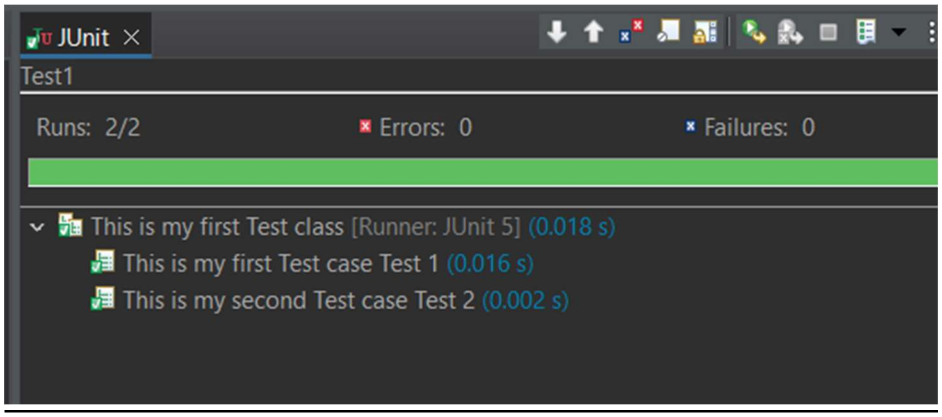
Pom.xml

```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>

    <groupId>com</groupId>
    <artifactId>ecommerce.junit</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>ecommerce.junit</name>
    <url>http://www.example.com</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>17</java.version>
        <maven.compiler.source>${java.version}</maven.compiler.source>
        <maven.compiler.target>${java.version}</maven.compiler.target>
        <maven.compiler.release>${java.version}</maven.compiler.release>

        <junit>5.9.1</junit>

        <!-- Plugin versions -->
        <maven.shade>3.2.2</maven.shade>
        <maven.clean>3.1.0</maven.clean>
        <maven.resources>3.1.0</maven.resources>
        <maven.compiler>3.8.1</maven.compiler>
        <maven.surefire>3.0.0-M5</maven.surefire>
        <maven.jar>3.2.0</maven.jar>
        <maven.install>3.0.0-M1</maven.install>
    </properties>

    <dependencies>
        <!-- Dependencies -->


        <!-- Testing dependencies-->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-params</artifactId>
            <version>${junit}</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.0.0-M4</version>
            </plugin>
            <plugin>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.2.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-install-plugin</artifactId>
                <version>3.0.0-M1</version>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>${maven.shade}</version>
                <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                    <goal>shade</goal>
                    </goals>
                    <configuration>
                    <transformers>
                        <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTrans
former">
                        <mainClass>com.ecommerce.junit.App</mainClass>
                        </transformer>
                    </transformers>
                    </configuration>
                </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

OUTPUT



```
<terminated> Test1 [JUnit] C:\Users\sripa\.p2\pool\plugins\org.ecli
Inside  myBeforeAll()
Inside  myBeforeEach()
Test1
Inside  myAfterEach()
Inside  myBeforeEach()
Test2
Inside  myAfterEach()
Inside  myAfterAll()
```



```
Test1
Runs:  2/2          Errors:  0          Failures:  0
This is my first Test class [Runner: JUnit 5] (0.018 s)
    This is my first Test case Test 1 (0.016 s)
    This is my second Test case Test 2 (0.002 s)
```

2.Write a program to demonstrate assertions.

Calculator.java class

```java
package com.ecommerce.junit;

public class Calculator {

      public int add(int a,int b) {
            return a+b;
      }

}
```

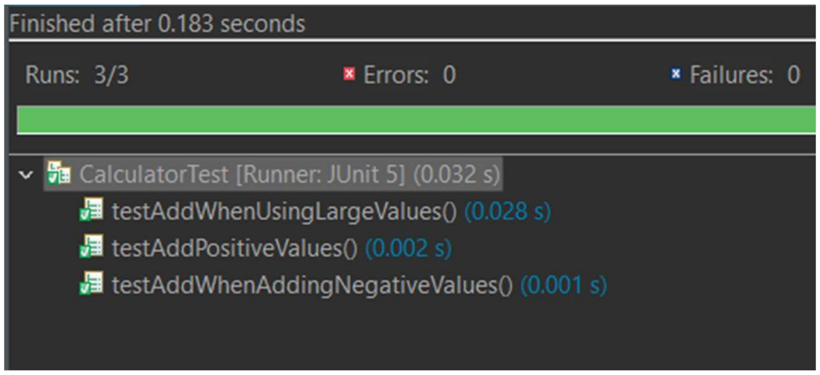
AssertionsTestDemo.java test

```java
package com.ecommerce.junit;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class AssertionsTestDemo {

      @Test
      void test() {

            String str = null;
            String str2 = "some value";

            String[] a1 = { "A", "B" };
            String[] a2 = { "A", "B" };

            int a=4;
            int b=0;

            assertTrue(a > b);
            assertFalse(5 < 1);

            assertNull(str);
        assertNotNull(str2);

        assertSame(str, str);
        assertNotSame(str, str2);

        assertEquals(5, 5);
        assertNotEquals(5, 6);

        assertArrayEquals(a1, a2);
        assertThrows(RuntimeException.class, () -> {
            throw new RuntimeException();
    });

      }

}
```

CalculatorTest.java test

```
package com.ecommerce.junit;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class CalculatorTest {

      @Test
      void testAddPositiveValues() {

            Calculator cal = new Calculator();

            int a=2;
            int b=5;

            assertEquals(7, cal.add(a, b));
      }

      @Test
      void testAddWhenAddingNegativeValues() {

            Calculator cal = new Calculator();

            int a=-2;
            int b=-3;

            assertEquals(-5, cal.add(a, b));
      }

      @Test
      void testAddWhenUsingLargeValues() {

            Calculator cal = new Calculator();

            int a=2500;
            int b=1000;

            assertEquals(3500, cal.add(a, b));
      }

}
```

OUTPUT



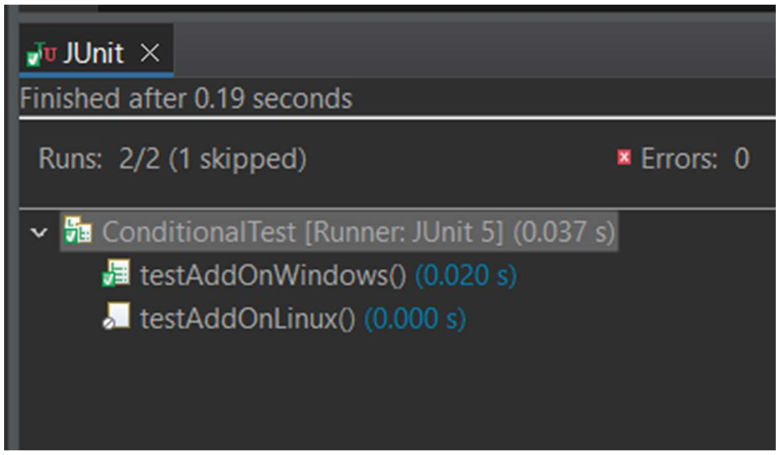
```
Finished after 0.183 seconds

Runs:  3/3          Errors:  0          Failures:  0

CalculatorTest [Runner: JUnit 5] (0.032 s)
    testAddWhenUsingLargeValues() (0.028 s)
    testAddPositiveValues() (0.002 s)
    testAddWhenAddingNegativeValues() (0.001 s)
```

3.Write a program to demonstrate conditional test executions.

ConditionalTest.java

```
package com.ecommerce.junit;
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.condition.EnabledOnOs;
import org.junit.jupiter.api.condition.OS;
class ConditionalTest {

      @Test
      @EnabledOnOs({OS.WINDOWS})
      public void testAddOnWindows() {
      Calculator cal =new Calculator();

      int x=2;
      int y=5;

      assertEquals(7, cal.add(x, y));

      }

      @Test
      @EnabledOnOs({OS.LINUX})
      public void testAddOnLinux() {
      Calculator cal =new Calculator();

      int x=2;
      int y=5;

      assertEquals(7, cal.add(x, y));
      }

}
```

OUTPUT



```
JUnit
Finished after 0.19 seconds
Runs: 2/2 (1 skipped)          Errors: 0
ConditionalTest [Runner: JUnit 5] (0.037 s)
    testAddOnWindows() (0.020 s)
    testAddOnLinux() (0.000 s)
```

4.Write a program to demonstrate nested and repeated tests

Test cases

RepeatedTestDemo.java

```
package com.ecommerce.junit;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.RepeatedTest;

class RepeatedTestDemo {

	@RepeatedTest(5)
	void testAddPositiveValues() {

		Calculator cal = new Calculator();

		int a=2;
		int b=5;

		assertTrue(7 ==cal.add(a, b));
	}

}
```

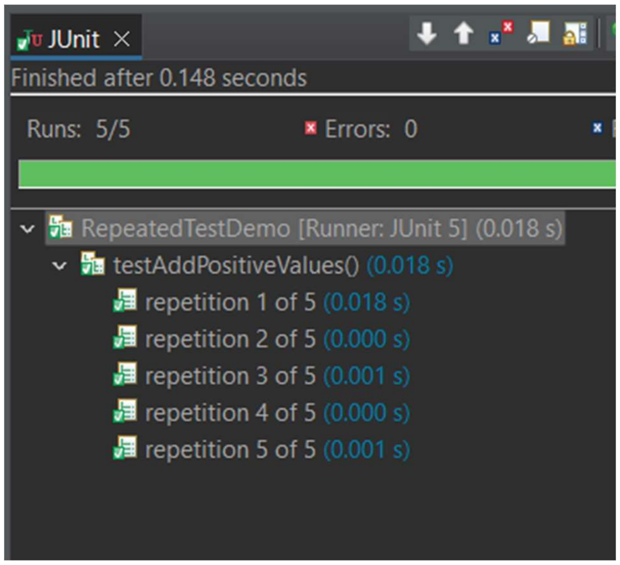
NestedTestDemo.java

```
package com.ecommerce.junit;

import org.junit.jupiter.api.Nested;
import org.junit.jupiter.api.Test;

class NestedTestDemo {

	@Test
	void test() {
		System.out.println("Inside  test()");
	}

	@Nested
	class GroupA {

		@Test
		void testA1() {
			System.out.println("Inside  testA1()");
		}

		@Test
		void testA2() {
			System.out.println("Inside  testA2()");
		}

	}
```

```java
        @Nested
        class GroupB {

                @Test
                void testB1() {
                        System.out.println("Inside  testB1()");
                }

                @Test
                void testB2() {
                        System.out.println("Inside  testB2()");
                }
        }

}
```

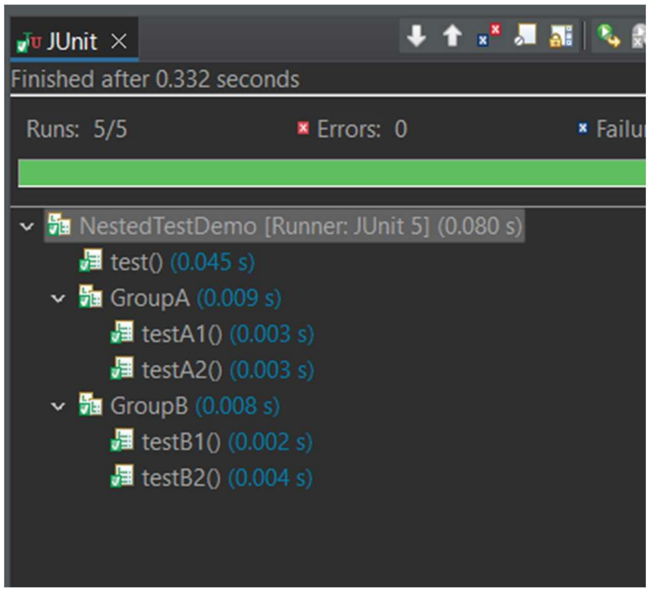
Output

repeatedtestDemo

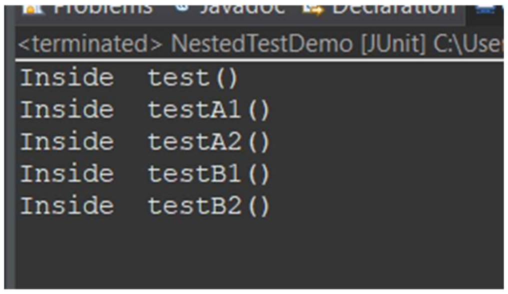


```
JUnit
Finished after 0.148 seconds
Runs: 5/5          Errors: 0
RepeatedTestDemo [Runner: JUnit 5] (0.018 s)
  testAddPositiveValues() (0.018 s)
    repetition 1 of 5 (0.018 s)
    repetition 2 of 5 (0.000 s)
    repetition 3 of 5 (0.001 s)
    repetition 4 of 5 (0.000 s)
    repetition 5 of 5 (0.001 s)
```

NestedTestDemo



```
Finished after 0.332 seconds

Runs: 5/5        Errors: 0        Failu

NestedTestDemo [Runner: JUnit 5] (0.080 s)
    test() (0.045 s)
    GroupA (0.009 s)
        testA1() (0.003 s)
        testA2() (0.003 s)
    GroupB (0.008 s)
        testB1() (0.002 s)
        testB2() (0.004 s)
```



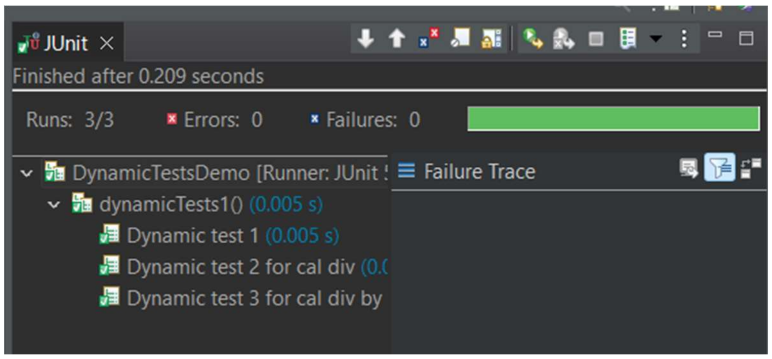
```
<terminated> NestedTestDemo [JUnit] C:\Use
Inside  test()
Inside  testA1()
Inside  testA2()
Inside  testB1()
Inside  testB2()
```

5.Write a program to demonstrate dynamic tests.

DynamicTestsDemo.java

```
package com.ecommerce.junit;
import static org.junit.jupiter.api.Assertions.assertThrows;
import static org.junit.jupiter.api.Assertions.assertTrue;
import static org.junit.jupiter.api.DynamicTest.dynamicTest;
import java.util.Arrays;
import java.util.Collection;

import org.junit.jupiter.api.DynamicTest;
import org.junit.jupiter.api.TestFactory;

class DynamicTestsDemo {

        @TestFactory
        Collection<DynamicTest> dynamicTests1() {

                return Arrays.asList(

                dynamicTest("Dynamic test 1", () -> assertTrue(7 ==new Calculator().add(2, 5))),

                dynamicTest("Dynamic test 2 for cal div", () -> assertTrue(2 ==new Calculator().divide(5,
2))),

                dynamicTest("Dynamic test 3 for cal div by 0", () ->
assertThrows(ArithmeticException.class, () -> new Calculator().divide(5,0)))

                );

        }
}
```

Output



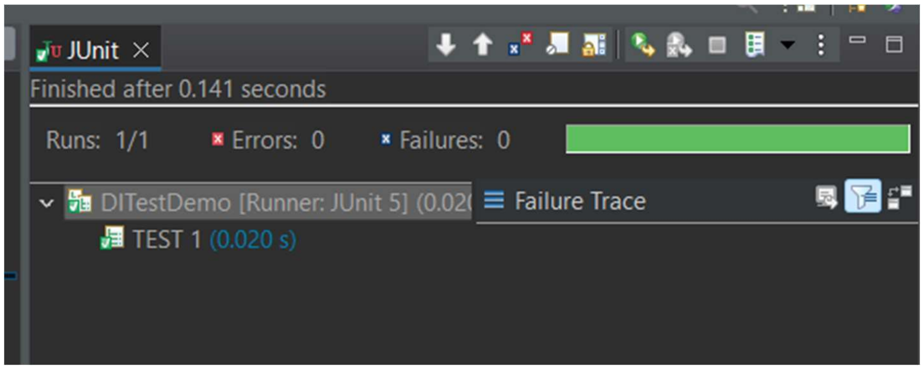
```
JUnit
Finished after 0.209 seconds
Runs: 3/3    Errors: 0    Failures: 0
DynamicTestsDemo [Runner: JUnit !
  dynamicTests1() (0.005 s)
    Dynamic test 1 (0.005 s)
    Dynamic test 2 for cal div (0.(
    Dynamic test 3 for cal div by
```

6. Write a program to demonstrate a dependency injection.

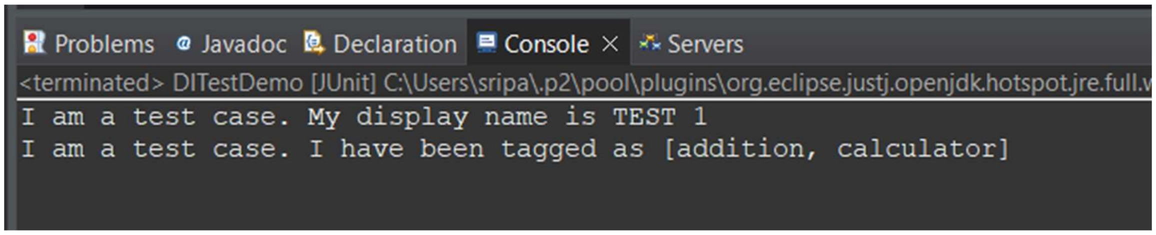
DITestDemo.java

```java
package com.ecommerce.junit;

import static org.junit.jupiter.api.Assertions.assertTrue;

import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.TestInfo;

class DITestDemo {

	@Test
	@DisplayName("TEST 1")
	@Tag("addition")
	@Tag("calculator")
	void test(TestInfo testinfo) {
		// ...as usual test our calculator add functionality

		System.out.println("I am a test case. My display name is "+
testinfo.getDisplayName());

		System.out.println("I am a test case. I have been tagged as "+
testinfo.getTags());

		assertTrue(testinfo.getTags().contains("addition"));

	}

}
```

Output



```
Finished after 0.141 seconds
Runs: 1/1    Errors: 0    Failures: 0
DITestDemo [Runner: JUnit 5] (0.02
    TEST 1 (0.020 s)
```



```
<terminated> DITestDemo [JUnit] C:\Users\sripa\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.w
I am a test case. My display name is TEST 1
I am a test case. I have been tagged as [addition, calculator]
```

7. You are given a project to demonstrate RESTful with Spring Boot.

Pom.xml

```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>
```

```xml
                  <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

```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){
```

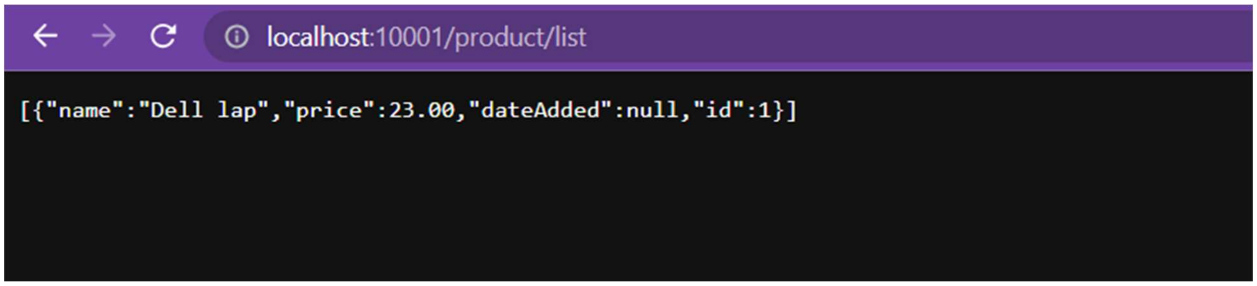
```
                eProduct = eProductRepo.save(eProduct);
                return eProduct;
        }

        // 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

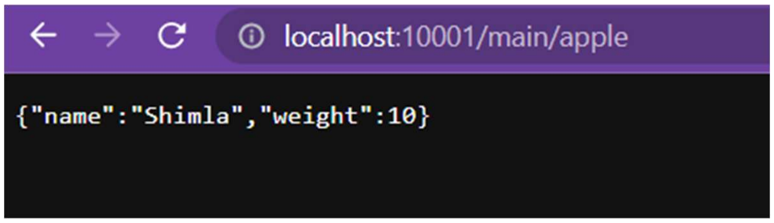
```
package com.ecommerce;


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);
	}

}
```

OUTPUT



```
[{"name":"Dell lap","price":23.00,"dateAdded":null,"id":1}]
```



```
{"name":"Shimla","weight":10}
```