



Back End Development 1 ○

+

Sesi 20



Secure Springboot+App

Secure Your Springboot App - Sesi 20

Introductions

Security adalah salah satu fitur yang sangat penting dalam membangun sebuah aplikasi. Tanpa adanya fitur ini maka aplikasi yang dibuat menjadi tidak bagus.

Biasanya fitur security dalam aplikasi menggunakan mekanisme Authentication dan Authorization. Authentication adalah fitur security yang menangani siapa user yang masuk ke dalam system. Sedangkan Authorization adalah setelah user tersebut berhasil masuk ke dalam system, lalu user tersebut bisa mengakses kemana saja.

Dalam Spring Security fitur tersebut sudah dibungkus dalam satu library yaitu spring-boot-startersecurity. Pada sesi ini, masih menggunakan source code pada sesi 19

Edit file pom.xml dan tambahkan library tersebut.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```



```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5
6     <modelVersion>4.0.0</modelVersion>
7     <parent>
8         <groupId>org.springframework.boot</groupId>
9         <artifactId>spring-boot-starter-parent</artifactId>
10        <version>2.1.12.RELEASE</version>
11        <relativePath/> <!-- lookup parent from repository -->
12    </parent>
13
14    <groupId>id.learn</groupId>
15    <artifactId>webservices-restful</artifactId>
16    <version>0.0.1-SNAPSHOT</version>
17    <name>webservices-restful</name>
18    <description>Demo project for Spring Boot</description>
19
20    <properties>
21        <java.version>1.8</java.version>
22    </properties>
23
24    <dependencies>
25        <dependency>
26            <groupId>org.springframework.boot</groupId>
27            <artifactId>spring-boot-starter-data-jpa</artifactId>
28        </dependency>
29
30        <dependency>
31            <groupId>org.springframework.boot</groupId>
32            <artifactId>spring-boot-starter-web</artifactId>
33        </dependency>
34
35        <dependency>
36            <groupId>mysql</groupId>
37            <artifactId>mysql-connector-java</artifactId>
38            <scope>runtime</scope>
39        </dependency>
40

```



```

41     <dependency>
42         <groupId>org.projectlombok</groupId>
43         <artifactId>lombok</artifactId>
44         <optional>true</optional>
45     </dependency>
46
47     <dependency>
48         <groupId>org.springframework.boot</groupId>
49         <artifactId>spring-boot-starter-test</artifactId>
50         <scope>test</scope>
51     </dependency>
52
53     <dependency>
54         <groupId>org.springframework.boot</groupId>
55         <artifactId>spring-boot-starter-security</artifactId>
56     </dependency>
57 </dependencies>
58
59 <build>
60     <plugins>
61         <plugin>
62             <groupId>org.springframework.boot</groupId>
63             <artifactId>spring-boot-maven-plugin</artifactId>
64         </plugin>
65     </plugins>
66 </build>
67
68 </project>

```

Dalam library spring-boot-starter-security, terdapat tiga komponen penting yang dijadikan dalam satu bundle, yaitu :

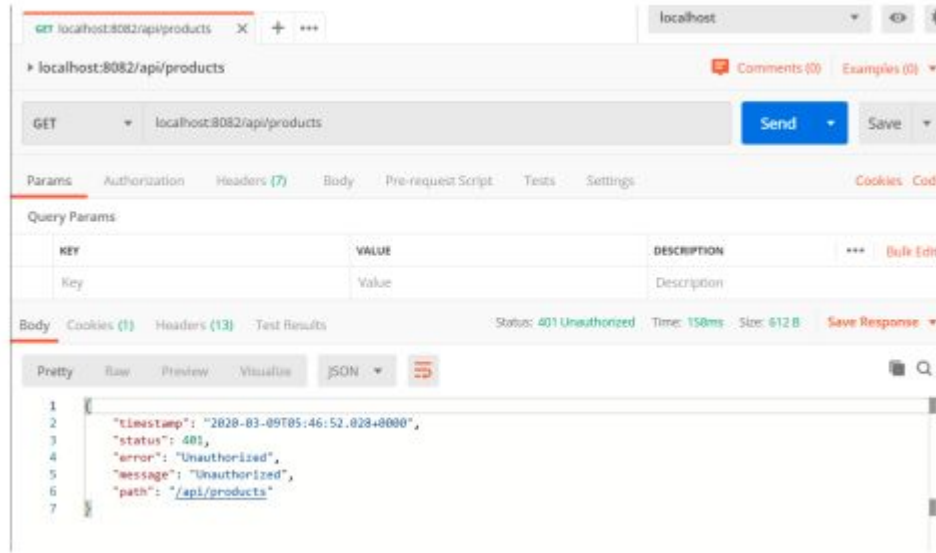
- Spring Security Config
- Spring Security Core
- Spring Security Web



Secure Your Springboot App - Sesi 20

Basic Authentication

Secara default ketika kita menambahkan plugin spring boot security, maka semua endpoint yang telah dibuat bersifat authenticated atau user harus login terlebih dahulu. Bisa kita check, akses endpoint `/api/products` dan hasil nya akan mengembalikan status 401 Unauthorized.



Ketika sebuah resource diamankan menggunakan basic authentication maka kita perlu mengirim username dan password untuk melakukan request authentication.

Tapi jika kita tidak mengirim, maka secara default Spring Boot membuatkan password generator dengan menggunakan basic username yaitu user .

Untuk password generator sendiri, bisa dilihat pada log seperti berikut.

```
2020-03-09 12:46:02.180 WARN 1868 --- [main] oWebConfiguration$JpaWebMvcConfiguration : spring.jpa.open-in-view is enabled by de
2020-03-09 12:46:02.410 INFO 1868 --- [main] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: 1455cd17-05cf-457c-abe8-ac45503ad4a6

2020-03-09 12:46:02.535 INFO 1868 --- [main] o.s.s.web.DefaultSecurityFilterChain : Creating filter chain: any request, [org
2020-03-09 12:46:02.630 INFO 1868 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8082 (http) w
2020-03-09 12:46:02.630 INFO 1868 --- [main] i.l.w.WebservicesRestfulApplication : Started WebservicesRestfulApplication in
2020-03-09 12:46:51.982 INFO 1868 --- [nio-8082-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'd
2020-03-09 12:46:51.982 INFO 1868 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2020-03-09 12:46:51.987 INFO 1868 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 5 ms
```

Lakukan pengetesan kembali menggunakan postman client, masih menggunakan endpoint yang sama tapi dalam tab Authorization menggunakan tipe Basic Auth, masukan username dengan nilai user dan password dengan nilai dari password generator yang ada di log



GET

localhost:8082/api/products

Send

Save

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authentication](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username

User

Password

1455cd17-05cf-457c-abe8-ac45503ad4a6

☒ Show Password

Body

Cookies (1)

Headers (12)

Test Results

Status: 200 OK

Time: 515ms

Size: 577 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

JS

```
1 {
2   {
3     "id": 3,
4     "nama": "Keyboard gaming",
5     "hargaBel1": 50000,
6     "hargaJual": 85000
7   },
8   {
9     "id": 4,
10    "nama": "Headset gaming",
11    "hargaBel1": 100000,
12    "hargaJual": 905000
13  }
14 }
```



Lalu muncul pertanyaan, bagaimana jika kita ingin mendefinisikan sendiri username dan password nya sehingga tidak perlu menggunakan password generator.

Untuk itu masukan konfigurasi berikut pada file application.properties nya.

```
#Security Config  
spring.security.user.name=user  
spring.security.user.password=@123
```

application.properties

#MySQL Connection

```
spring.datasource.url = jdbc:mysql://localhost:3306/spring-webservice
spring.datasource.username = root
spring.datasource.password = root
```

#JPA Properties

```
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driverClassName=com.mysql.jdbc.Driver
```

#HikariCP

```
spring.datasource.hikari.connection-timeout=20000
spring.datasource.hikari.minimum-idle=5
spring.datasource.hikari.maximum-pool-size=12
spring.datasource.hikari.idle-timeout=300000
spring.datasource.hikari.max-lifetime=120000
spring.datasource.hikari.auto-commit=true
```

#Security Config

```
spring.security.user.name=user
spring.security.user.password=@123
```

```
server.port=8082
```



Jalankan kembali aplikasinya, dan lakukan pengetestan kembali menggunakan postman dan jangan lupa gunakan username dan password yang sudah didefinisikan pada file application.properties.

TYPE
Basic Auth

The authorization header will be automatically generated when you send the request. [Learn more about authentication](#)

Preview Request

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)

Username: user

Password: @123

☒ Show Password

Body Cookies (1) Headers (12) Test Results

Status: 200 OK Time: 542ms Size: 577 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 3,
4     "nama": "Keyboard gaming",
5     "hargaBeli": 50000,
6     "hargaJual": 85000
7   },
8   {
9     "id": 4,
10    "nama": "Headset gaming",
11    "hargaBeli": 100000,
12    "hargaJual": 90000
13  }
14 }
```



UNIT TESTING +

Introductions

Unit Testing adalah salah satu level dari proses testing dalam software development.

Unit testing adalah pengujian dasar yang menguji setiap unit atau component baik itu dari segi functional atau behavior. Sebagai programmer, sudah menjadi suatu kewajiban selain bisa membuat program, juga bisa membuat unit testing dari program yang telah di tulis.

Kenapa ?

karena kalau tidak melakukan unit testing kita tidak bisa meyakinkan orang lain bahwa kode yang sudah kita tulis itu sudah sesuai dengan proses bisnis atau tidak dan bagaimana kualitas code yang ditulis.

Namun kebanyakan programmer khusus nya pemula, suka melewati pembelajaran unit testing ini padahal hal ini sangat lah penting dalam software development.

Pdas sesi ini akan dijelaskan dasar bagaimana membuat Unit Testing menggunakan tools yang sering digunakan yaitu JUnit dan Mockito.



Secure Your Springboot App - Sesi 20

Plugin Unit Test, Commons & Mapper

Plugin Unit Testing Tambahkan beberapa plugin berikut untuk membuat unit testing yaitu JUnit dan Mockito.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
  <exclusions>
    <exclusion>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-params</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.platform</groupId>
  <artifactId>junit-platform-launcher</artifactId>
  <version>1.3.2</version>
  <scope>test</scope>
</dependency>
</dependency>
```



```
<groupId>org.junit.vintage</groupId>  
<artifactId>junit-vintage-engine</artifactId>  
<version>5.3.2</version>  
<scope>test</scope>  
</dependency>  
<dependency>  
  <groupId>org.mockito</groupId>  
  <artifactId>mockito-junit-jupiter</artifactId>  
  <scope>test</scope>  
</dependency>
```



Plugin Apache Commons

Plugin ini akan kita gunakan untuk menghasilkan data palsu misalkan membuat nama, title, alamat menggunakan class RandomStringUtils.

Karena generator fake data ini sangat diperlukan dalam unit testing juga

```
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.9</version>
</dependency>
```

Plugin Zalando Yang terakhir adalah plugin zalando yaitu mapper yang berfungsi untuk mengubah file json menjadi object.

Plugin ini nantinya akan digunakan untuk membuat unit testing dalam layer controller

```
<dependency>
  <groupId>org.zalando</groupId>
  <artifactId>problem-spring-web-starter</artifactId>
  <version>${problem-spring-web.version}</version>
  <type>pom</type>
</dependency>
```




```
<?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>2.1.12.RELEASE</version>
```



```
    <relativePath/> <!-- Lookup parent from repository -->
  </parent>
  <groupId>id.learn</groupId>
  <artifactId>webservices-restful</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>webservices-restful</name>
  <description>Demo project for Spring Boot</description>

  <properties>
    <java.version>1.8</java.version>
    <problem-spring-web.version>0.25.0</problem-spring-web.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>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
  </dependencies>
```



```
<!-- Random String generator -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.9</version>
</dependency>

<!-- For Testing-->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
  <exclusions>
    <exclusion>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
```



```

        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-params</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.platform</groupId>
        <artifactId>junit-platform-launcher</artifactId>
        <version>1.3.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.vintage</groupId>
        <artifactId>junit-vintage-engine</artifactId>
        <version>5.3.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <scope>test</scope>
    </dependency>

    <!-- mapper -->
    <dependency>
        <groupId>org.zalando</groupId>
        <artifactId>problem-spring-web-starter</artifactId>
        <version>${problem-spring-web.version}</version>
        <type>pom</type>
    </dependency>
</dependencies>

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

```



Secure Your Springboot App - Sesi 20

Create Fake Data Generator

Buatlah sebuah class dalam package src/test dengan nama TestObjectFactory. Dalam class tersebut kita membuat object dari class Product dan membuat object List dengan tipe Product juga.

```
id.learn.webserservicerestful  
  
package id.learn.webservicesrestful;  
  
import id.learn.webservicesrestful.model.Product;  
import org.apache.commons.lang3.RandomStringUtils;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;
```



```
public class TestObjectFactory {  
  
    public static Product createProduct() {  
        final Product product = new Product();  
        product.setId(new Random().nextLong());  
        product.setNama(RandomStringUtils.randomAlphabetic(10));  
        product.setHargaBeli(new Random().nextLong());  
        product.setHargaJual(new Random().nextLong());  
  
        return product;  
    }  
  
    public static List<Product> createProductList(final int size) {  
        final List<Product> result = new ArrayList<>();  
        for (int i = 0; i < size; i++) {  
            result.add(createProduct());  
        }  
        return result;  
    }  
}
```



Penjelasan kode diatas :

- Method `createProduct()` berfungsi untuk membuat object dari class `Product`.dalam method tersebut terlihat kita menggunakan class `RandomStringUtils` yang merupakan salah satu class dari library `org.apache.commons`.
- Method `createProductList` berfungsi untuk membuat object List dengan tipe data `Product` dengan menggunakan parameter `size` sehingga kita bisa menentukan jumlah data dalam List tersebut.



Unit Testing Service Layer

Dimulai dari service layer dimana layer tersebut adalah logic dari aplikasi ini dan bisa berkomunikasi dengan repository atau database.

Kenapa pada layer repository tidak di test ?

karena kita menggunakan Spring Data JPA untuk layer repositorynya maka secara otomatis sudah dites juga oleh pihak terkaitnya sehingga kita tidak perlu melakukan testing lagi untuk layer repository.

Buat sebuah class ProductServiceTest dengan base konfigurasi seperti dibawah ini.

id.learn.webserservicerestful.ProductServiceTest

```
@RunWith(SpringRunner.class)
public class ProductServiceTest{

    @InjectMocks
    private ProductService productService = new ProductServiceImpl();

    @Mock
    private ProductRepository productRepository;

    @Before
    public void setup() {

        MockitoAnnotations.initMocks(this);
        ReflectionTestUtils.setField(productService, "productRepository",
productRepository);
    }

}
```



Penjelasan kode diatas :

- `@RunWith(SpringRunner.class)` adalah sebuah alias dari class `SpringJUnit4ClassRunner` yang menghubungkan JUnit dan Spring TestContext. Dengan `SpringRunner`, kita dapat mengimplementasikan JUnit dan integration test.
- `@InjectMock` untuk membuat object dari class yang akan di test, dalam hal ini adalah `ProductServiceImpl`.
- `@Mock` membuat dependensi tiruan (mock) karena dalam class `ProductServiceImpl` terdapat dependensi ke `ProductRepository`. Sehingga nantinya kita seakan-akan berkomunikasi dengan database.
- `@Before` adalah anotasi untuk mengeksekusi pertama kali setiap dilakukan unit testing di panggil.
- `MockitoAnnotations.initMocks(this)` menginisialisasi setiap property atau field yang diberi anotasi `@Mock`.
- `ReflectionTestUtils` adalah salah satu bagian dari Spring Test Context yang merupakan kumpulan dari method-method utilitas berbasis refleksi yang dilakukan unit testing dan integration test untuk memanggil method private dan melakukan injection.



Testing findAll

Didalam class ProductServiceImpl terdapat method findAllProduct(),method ini akan kita coba buat unit testing nya.

Buatlah sebuah method dengan nama testFindAll() didalam class ProductServiceTest.

```
id.learn.webserservicerestful.ProductServiceTest
```

```
...
```

```
@Test
```

```
public void testFindAll() {
```

```
    final List<Product> datas = TestObjectFactory.createProductList(10);
```

```
    Mockito.when(productRepository.findAll()).thenReturn(datas);
```

```
    final List<Product> actual = productService.findAllProducts();
```

```
    MatcherAssert.assertThat(actual.size(), Matchers.equalTo(datas.size()));
```

```
}
```

```
...
```



Penjelasan kode diatas :

- `@Test` adalah anotasi JUnit yang menandakan bahwa method `testFindAll` adalah sebuah method yang digunakan untuk mengetest method tertentu, dalam hal ini method `findAllProduct`
- Kita siapkan dummy data dan simpan dalam variable `datas` dengan jumlah data sebesar 10 item.
- `Mockito.when(productRepository.findAll()).thenReturn(datas)` merupakan simulasi bahwa kita seakan-akan memanggil method `findAll` dari `productRepository` dan hasil dari pemanggilan itu kemudian mengembalikan variable `datas` yang telah diinisialisasi.
- `final List actual = productService.findAllProducts()` adalah simulasi bahwa eksekusi berikutnya seakan-akan kita memanggil method `findAllProducts()` dalam `productService`.
- Yang terakhir, kita lakukan pencocokan jumlah data antara data yang diinisialisasi dalam object `datas` dengan object `actual` dengan menggunakan class `MatcherAssert.assertThat(...)` harus sama-sama melempar jumlah 10;
- Run unit test method ini, jika berhasil tampilannya akan sebagai berikut.

```

C:\Program Files\Java\jdk-8.0.232\bin\java.exe
ProductServiceTest class: 77 ms
testFailed 77 ms

17:05:20.096 [main] DEBUG org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with [class id.learns.webserver
17:05:20.095 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context
17:05:20.095 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context
17:05:20.106 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [id.learns.webserver.restful.service
17:05:20.131 [main] INFO org.springframework.test.context.support.DefaultTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test
17:05:20.139 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to GenericXmlContextLoader to process context co
17:05:20.166 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [id.learns.webserver
17:05:20.167 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [id.learns.webserver
17:05:20.151 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to AnnotationConfigContextLoader to process conte
17:05:20.152 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [
17:05:20.266 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annotation declaring class' for annotation type [org.spring
17:05:20.247 [main] DEBUG org.springframework.test.context.support.DefaultTestContextBootstrapper - @TestExecutionListeners is not present for class [id.learns.webserver

```



Secure Your Springboot App - Sesi 20

Testing getProductById

Didalam class ProductServiceImpl terdapat method findProductById(),method ini akan kita coba buat unit testing nya.

Buatlah sebuah method dengan nama testProductById() didalam class ProductServiceTest.

```
id.learn.webserservicerestful.ProductServiceTest
```

```
...
```

```
@Test
```

```
public void testProductById() throws Exception {  
    final Long id = new Random().nextLong();  
    final Product product = TestObjectFactory.createProduct();
```

```
    Mockito.when(productRepository.findById(id)).thenReturn(Optional.of(product));
```

```
    final Product actual = productService.findProductById(id);
```

```
    MatcherAssert.assertThat(actual.getId(), Matchers.equalTo(product.getId()));
```

```
    MatcherAssert.assertThat(actual.getNama(),
```

```
    Matchers.equalTo(product.getNama()));
```

```
    MatcherAssert.assertThat(actual.getHargaBeli(),
```

```
    Matchers.equalTo(product.getHargaBeli()));
```

```
    MatcherAssert.assertThat(actual.getHargaJual(),
```



```
Matchers.equalTo(product.getHargaJual()));  
}
```

...

Penjelasan kode diatas :

- final Long id karena kita akan mensimulasikan findProductById maka kita harus menyiapkan variable id dengan tipe data Long dan inisialisasi dengan nilai Long secara random / acak.
- Buat object Product dan inisialisasi dengan TestObjectFactory.
- Mockito.when(productRepository.findById(id)).thenReturn(Optional. of(product)) eksekusi productRepository.findById() mock dan kembalikan object product tapi menggunakan class Optional. Karena jika kita lihat dalam method ProductService
- findProductById itu melempar object Optional.
- final Product actual buat object actual seakan-akan kita mengakses method productService.findById(id).



- Terakhir lakukan pencocokan data antara objek actual dan objek product.
- Jika berhasil maka hasilnya akan seperti ini

```
ProductServiceTest [main] INFO org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with [class id.learn.webservicesrestful.service.ProductServiceTest]
testProductStyle [main] INFO org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with [class id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.878 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.context.BootstrapUtils]
17:34:53.907 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.context.BootstrapContext(bootstrapContextLoader: org.springframework.test.context.CacheAwareContextLoaderDelegate)
17:34:53.926 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.958 [main] INFO org.springframework.test.context.support.DefaultTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for test class [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.958 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to GenericXmlContextLoader to process context class [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.968 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [id.learn.webservicesrestful.service.ProductServiceTest]: no resource found for [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.988 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [id.learn.webservicesrestful.service.ProductServiceTest]: no resource found for [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.970 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to AnnotationConfigContextLoader to process context class [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:53.970 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test class [id.learn.webservicesrestful.service.ProductServiceTest]: no configuration classes found in [id.learn.webservicesrestful.service.ProductServiceTest]
17:34:54.058 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find 'annotation declaring class' for annotation type [org.springframework.test.context.ActiveProfiles]
17:34:54.058 [main] DEBUG org.springframework.test.context.support.DefaultTestContextBootstrapper - @TestExecutionListeners is not present for class [id.learn.webservicesrestful.service.ProductServiceTest]
```



Testing getProductByIdWithNullDataFromDB

Sebenarnya test ini sama dengan method diatas yaitu mencari data single product berdasarkan ID namun disimulasikan seakan-akan data dari ID yang dicari itu tidak ada di database

```
id.learn.webserservicerestful.ProductServiceTest
```

```
...
```

```
@Test
```

```
public void testProductByIdWithNullDataFromDB() throws Exception {  
    final Long id = new Random().nextLong();
```

```
    Mockito.when(productRepository.findById(id)).thenReturn(Optional.empty());
```

```
    final Product actual = productService.findProductById(id);
```

```
    MatcherAssert.assertThat(actual, Matchers.nullValue());
```

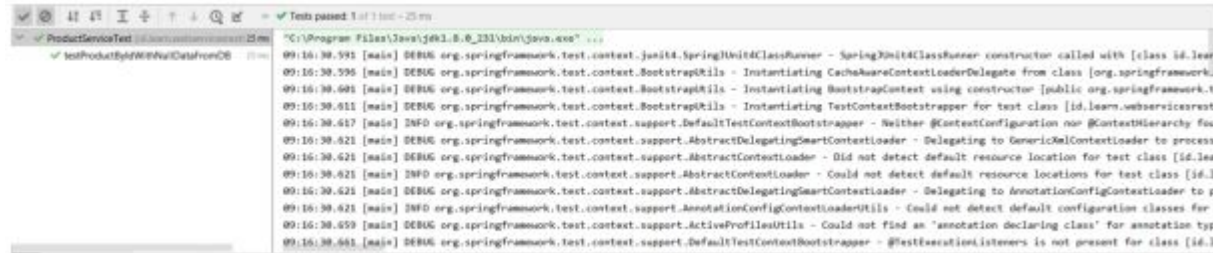
```
}
```

```
...
```



Penjelasan kode diatas :

- Saat memanggil productRepository maka method tersebut mengembalikan data null dengan menggunakan Optional.empty().
- final Product actual membuat object real seakan-akan kita mengakses method findProductById di ProductServiceImpl.
- Terakhir lakukan pencocokan antara object actual , lakukan pencocokan dengan nilai null karena itu yang kita akan test kemudian test.



```
Tests passed: 1 of 1 test - 25 ms
ProductServiceTest
testProductByIdWithDataSourceDB
09:16:38.593 [main] INFO org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with [class id.learn
09:16:38.595 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.t
09:16:38.605 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.t
09:16:38.611 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [id.learn.webservicesrest
09:16:38.617 [main] INFO org.springframework.test.context.support.DefaultTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy fou
09:16:38.621 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to GenericXmlContextLoader to process
09:16:38.621 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [id.lea
09:16:38.621 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [id.l
09:16:38.621 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to AnnotationConfigContextLoader to p
09:16:38.621 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for
09:16:38.659 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annotation declaring class' for annotation typ
09:16:38.661 [main] DEBUG org.springframework.test.context.support.DefaultTestContextBootstrapper - @TestExecutionListeners is not present for class [id.l]
```

Secure Your Springboot App - Sesi 20

Testing saveOrUpdateProduct

Didalam class ProductServiceImpl terdapat method saveOrUpdateProduct(),method ini akan kita coba buat unit testing nya.

Buatlah sebuah method dengan nama testSaveOrUpdateProduct() didalam class ProductServiceTest.

id.learn.webserservicerestful.ProductServiceTest

...

@Test

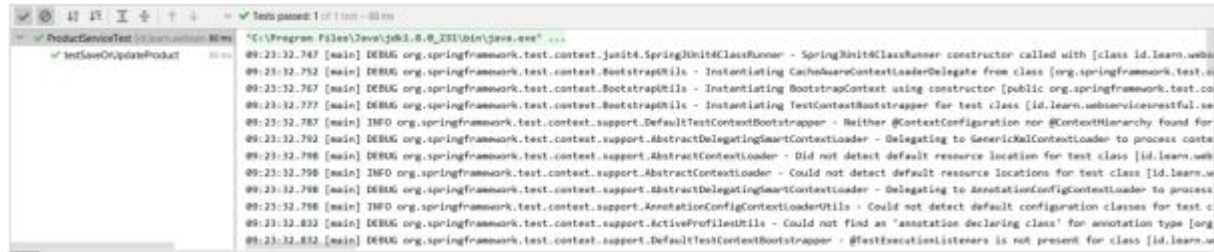
```
public void testSaveOrUpdateProduct() {  
    final Product product = TestObjectFactory.createProduct();  
  
    Mockito.when(productRepository.save(product)).thenReturn(product);  
  
    final Product actual = productService.saveOrUpdateProduct(product);  
  
    MatcherAssert.assertThat(actual, Matchers.notNullValue());  
}
```

...



Penjelasan kode diatas :

- Kita membuat object product menggunakan factory dummy.
- Panggil `productRepository.save()` dan seakan-akan mengembalikan data product dari database
- Objek real actual dipanggil seakan-akan kita memanggil method `saveOrUpdateProduct` dari layer service.
- Dan terakhir lakukan pencocokan antara object actual dengan `Matchers.notNullValue()` karena kita menghendaki nilai dari product itu tidak null. Kenapa kita bisa memastikan objek product itu tidak null ? karena kita sudah menginisialisasi menggunakan class `TestObjectFactory`



```
00:23:32.747 [main] DEBUG org.springframework.test.context.junit4.SpringJUnit4ClassRunner - SpringJUnit4ClassRunner constructor called with [class id.learn.web...
00:23:32.752 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating CacheAwareContextLoaderDelegate from class [org.springframework.test.c...
00:23:32.767 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating BootstrapContext using constructor [public org.springframework.test.co...
00:23:32.777 [main] DEBUG org.springframework.test.context.BootstrapUtils - Instantiating TestContextBootstrapper for test class [id.learn.webservices.restful.se...
00:23:32.787 [main] INFO org.springframework.test.context.support.DefaultTestContextBootstrapper - Neither @ContextConfiguration nor @ContextHierarchy found for...
00:23:32.792 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to GenericXmlContextLoader to process conte...
00:23:32.798 [main] DEBUG org.springframework.test.context.support.AbstractContextLoader - Did not detect default resource location for test class [id.learn.web...
00:23:32.798 [main] INFO org.springframework.test.context.support.AbstractContextLoader - Could not detect default resource locations for test class [id.learn.w...
00:23:32.798 [main] DEBUG org.springframework.test.context.support.AbstractDelegatingSmartContextLoader - Delegating to AnnotationConfigContextLoader to process...
00:23:32.798 [main] INFO org.springframework.test.context.support.AnnotationConfigContextLoaderUtils - Could not detect default configuration classes for test c...
00:23:32.832 [main] DEBUG org.springframework.test.context.support.ActiveProfilesUtils - Could not find an 'annotation declaring class' for annotation type [org...
00:23:32.832 [main] DEBUG org.springframework.test.context.support.DefaultTestContextBootstrapper - #testExecutionListeners is not present for class [id.learn.w...
```



Secure Your Springboot App - Sesi 20

Testing Delete Product

Didalam class ProductServiceImpl terdapat method deleteProduct(),method ini akan kita coba buat unit testing nya.

Buatlah sebuah method dengan nama testDeleteProduct() didalam class ProductServiceTest.

```
id.learn.webserservicerestful.ProductServiceTest
```

```
...
```

```
@Test
public void testdeleteProduct() {

    final Long id = new Random().nextLong();

    Product product = TestObjectFactory.createProduct();

    Mockito.when(productRepository.findById(id)).thenReturn(Optional.of(product));
    Mockito.doNothing().when(productRepository).delete(product);

    productService.deleteProduct(id);

    Mockito.verify(productRepository, times(1)).delete(product);
}
```



Penjelasan kode diatas :

- Object id akan digunakan sebagai simulasi id yang dilempar dari client
- Object product diinisialisasi oleh class TestObjectFactory jadi seakan-akan object ini tidak berisi null
- `productRepository.findById(id)` repository memanggil method `findById` dengan parameter dari object id dan mengembalikan object product.
- `Mockito.doNothing().when(productRepository).delete(product)` menggunakan method `doNothing()` karena pada saat melakukan eksekusi kode ini, tidak melempar data apapun.
- `productService.deleteProduct(id)` kita langsung eksekusi method ini karena sifat nya tidak mengembalikan tipe data apapun (void – bisa di cek di class `ProductService`).
- `Mockito.verify(productRepository, times(1)).delete(product)` adalah sintak untuk memastikan bahwa kode telah mengeksekusi



```
package id.learn.webservicesrestful.service;

import id.learn.webservicesrestful.TestObjectFactory;
import id.learn.webservicesrestful.model.Product;
import id.learn.webservicesrestful.repository.ProductRepository;
import id.learn.webservicesrestful.service.impl.ProductServiceImpl;
import org.hamcrest.MatcherAssert;
import org.hamcrest.Matchers;
import org.junit.Before;
import org.junit.Test;
import org.junit.jupiter.api.BeforeEach;
import org.junit.runner.RunWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.MockitoAnnotations;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.util.ReflectionTestUtils;

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

import static org.mockito.Mockito.times;

@RunWith(SpringRunner.class)
public class ProductServiceTest {

    @InjectMocks
    private ProductService productService = new ProductServiceImpl();

    @Mock
    private ProductRepository productRepository;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
        ReflectionTestUtils.setField(productService, "productRepository",
            productRepository);
    }
}
```



```

    }

    @Test
    public void testFindAll() {
        final List<Product> datas = TestObjectFactory.createProductList(10);

        Mockito.when(productRepository.findAll()).thenReturn(datas);

        final List<Product> actual = productService.findAllProducts();

        MatcherAssert.assertThat(actual.size(), Matchers.equalTo(datas.size()));
    }

    @Test
    public void testProductById() throws Exception {
        final Long id = new Random().nextLong();
        final Product product = TestObjectFactory.createProduct();

        Mockito.when(productRepository.findById(id)).thenReturn(Optional.of(product));

        final Product actual = productService.findProductById(id);

        MatcherAssert.assertThat(actual.getId(),
            Matchers.equalTo(product.getId()));
        MatcherAssert.assertThat(actual.getNama(),
            Matchers.equalTo(product.getNama()));
        MatcherAssert.assertThat(actual.getHargaBeli(),
            Matchers.equalTo(product.getHargaBeli()));
        MatcherAssert.assertThat(actual.getHargaJual(),
            Matchers.equalTo(product.getHargaJual()));
    }
}

```




```
@Test
public void testProductByIdWithNullDataFromDB() throws Exception {
    final Long id = new Random().nextLong();
```

```
Mockito.when(productRepository.findById(id)).thenReturn(Optional.empty());
```

```
    final Product actual = productService.findProductById(id);
```

```
    MatcherAssert.assertThat(actual, Matchers.nullValue());
```

```
}
```

```
@Test
public void testSaveOrUpdateProduct() {
    final Product product = TestObjectFactory.createProduct();
```

```
    Mockito.when(productRepository.save(product)).thenReturn(product);
```

```
    final Product actual = productService.saveOrUpdateProduct(product);
```

```
    MatcherAssert.assertThat(actual, Matchers.notNullValue());
```

```
}
```

```
@Test
public void testdeleteProduct() {
```



```

        final Long id = new Random().nextLong();

        Product product = TestObjectFactory.createProduct();

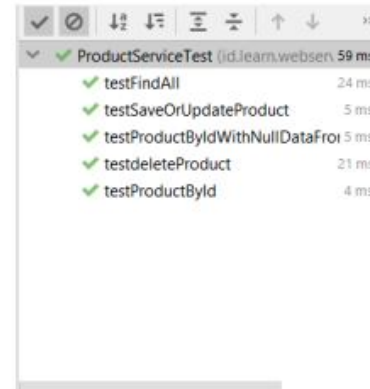
        Mockito.when(productRepository.findById(id)).thenReturn(Optional.of(product));
        Mockito.doNothing().when(productRepository).delete(product);

        productService.deleteProduct(id);

        Mockito.verify(productRepository, times(1)).delete(product);
    }
}

```

Dan jika kita eksekusi secara keseluruhan maka hasilnya akan seperti berikut.



Test Method	Duration
ProductServiceTest (junit.team.websen)	59 ms
testFindAll	24 ms
testSaveOrUpdateProduct	5 ms
testProductByIdWithNullDataFrom	5 ms
testdeleteProduct	21 ms
testProductById	4 ms

