TESTING

REVIEW SERVICE - TEST (4)

```
package se.magnus.microservices.core.review;

import static org.springframework.boot.test.context.SpringBootTest.WebEnvironment.RANDOM_PORT;
import static org.springframework.http.HttpStatus.BAD_REQUEST;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;
import static org.springframework.http.MediaType.APPLICATION_JSON;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=RANDOM_PORT)
public class ReviewServiceApplicationTests {

   @Autowired
   private WebTestClient client;

   @Test
   public void getReviewsByProductId() {

      int productId = 1;

      client.get()
         .uri("/review?productId=" + productId)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isOk()
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.length()").isEqualTo(3)
         .jsonPath("$[0].productId").isEqualTo(productId);
   }

   @Test
   public void getReviewsMissingParameter() {

      client.get()
         .uri("/review")
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(BAD_REQUEST)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/review")
         .jsonPath("$.message").isEqualTo("Required int parameter 'productId' is not present");
   }
   @Test
   public void getReviewsInvalidParameter() {

      client.get()
         .uri("/review?productId=no-integer")
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(BAD_REQUEST)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/review")
         .jsonPath("$.message").isEqualTo("Type mismatch.");
   }
   @Test
   public void getReviewsNotFound() {

      int productIdNotFound = 213;

      client.get()
         .uri("/review?productId=" + productIdNotFound)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isOk()
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.length()").isEqualTo(0);
   }

   @Test
   public void getReviewsInvalidParameterNegativeValue() {

      int productIdInvalid = -1;

      client.get()
         .uri("/review?productId=" + productIdInvalid)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(UNPROCESSABLE_ENTITY)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/review")
         .jsonPath("$.message").isEqualTo("Invalid productId: " + productIdInvalid);
   }
}
```

RECOMMENDATION SERVICE – TEST (5)

```java
package se.magnus.microservices.core.recommendation;

import static org.springframework.boot.test.context.SpringBootTest.WebEnvironment.RANDOM_PORT;
import static org.springframework.http.HttpStatus.BAD_REQUEST;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;
import static org.springframework.http.MediaType.APPLICATION_JSON;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=RANDOM_PORT)
public class RecommendationServiceApplicationTests {

   @Autowired
   private WebTestClient client;

   @Test
   public void getRecommendationsByProductId() {

      int productId = 1;

      client.get()
         .uri("/recommendation?productId=" + productId)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isOk()
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.length()").isEqualTo(3)
         .jsonPath("$[0].productId").isEqualTo(productId);
   }
   @Test
   public void getRecommendationsMissingParameter() {

      client.get()
         .uri("/recommendation")
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(BAD_REQUEST)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/recommendation")
         .jsonPath("$.message").isEqualTo("Required int parameter 'productId' is not present");
   }
   @Test
   public void getRecommendationsInvalidParameter() {

      client.get()
         .uri("/recommendation?productId=no-integer")
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(BAD_REQUEST)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/recommendation")
         .jsonPath("$.message").isEqualTo("Type mismatch.");
   }

   @Test
   public void getRecommendationsNotFound() {

      int productIdNotFound = 113;

      client.get()
         .uri("/recommendation?productId=" + productIdNotFound)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isOk()
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.length()").isEqualTo(0);
   }

   @Test
   public void getRecommendationsInvalidParameterNegativeValue() {

      int productIdInvalid = -1;

      client.get()
         .uri("/recommendation?productId=" + productIdInvalid)
         .accept(APPLICATION_JSON)
         .exchange()
         .expectStatus().isEqualTo(UNPROCESSABLE_ENTITY)
         .expectHeader().contentType(APPLICATION_JSON)
         .expectBody()
         .jsonPath("$.path").isEqualTo("/recommendation")
         .jsonPath("$.message").isEqualTo("Invalid productId: " + productIdInvalid);
   }
}
```

PRODUCT SERVICE – TEST (6)

```
package se.magnus.microservices.core.product;

import static org.springframework.boot.test.context.SpringBootTest.WebEnvironment.RANDOM_PORT;
import static org.springframework.http.HttpStatus.BAD_REQUEST;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;
import static org.springframework.http.MediaType.APPLICATION_JSON;

@RunWith(SpringRunner.class) //it will initialize our application similar to SpringBoot Application (application
context will be setup before executing the tests via Component Scanning and Auto-configuration)
@SpringBootTest(webEnvironment=RANDOM_PORT)

public class ProductServiceApplicationTests {

    @Autowired
    private WebTestClient client;

   @Test
   public void getProductById() {

      int productId = 1;

        client.get()
            .uri("/product/" + productId)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isOk()
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.productId").isEqualTo(productId);
   }

   @Test
   public void getProductInvalidParameterString() {

        client.get()
            .uri("/product/no-integer")
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isEqualTo(BAD_REQUEST)
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.path").isEqualTo("/product/no-integer")
            .jsonPath("$.message").isEqualTo("Type mismatch.");
   }
   @Test
   public void getProductNotFound() {

      int productIdNotFound = 13;

        client.get()
            .uri("/product/" + productIdNotFound)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isNotFound()
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.path").isEqualTo("/product/" + productIdNotFound)
            .jsonPath("$.message").isEqualTo("No product found for productId: " + productIdNotFound);
   }
   @Test
   public void getProductInvalidParameterNegativeValue() {

        int productIdInvalid = -1;

        client.get()
            .uri("/product/" + productIdInvalid)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isEqualTo(UNPROCESSABLE_ENTITY)
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.path").isEqualTo("/product/" + productIdInvalid)
            .jsonPath("$.message").isEqualTo("Invalid productId: " + productIdInvalid);
   }
}
```

PRODUCT COMPOSITE SERVICE – TEST (7)

```java
package se.magnus.microservices.composite.product;

import static java.util.Collections.singletonList;
import static org.mockito.Mockito.when;
import static org.springframework.boot.test.context.SpringBootTest.WebEnvironment.RANDOM_PORT;
import static org.springframework.http.HttpStatus.UNPROCESSABLE_ENTITY;
import static org.springframework.http.MediaType.APPLICATION_JSON;

@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment=RANDOM_PORT)
public class ProductCompositeServiceApplicationTests {

   private static final int PRODUCT_ID_OK = 1;
   private static final int PRODUCT_ID_NOT_FOUND = 2;
   private static final int PRODUCT_ID_INVALID = 3;

    @Autowired
    private WebTestClient client;

   @MockBean
   private ProductCompositeIntegration compositeIntegration;

   @Before
   public void setUp() {

      when(compositeIntegration.getProduct(PRODUCT_ID_OK)).
         thenReturn(new Product(PRODUCT_ID_OK, "name", 1, "mock-address"));
      when(compositeIntegration.getRecommendations(PRODUCT_ID_OK)).
         thenReturn(singletonList(new Recommendation(PRODUCT_ID_OK, 1, "author", 1, "content", "mock address")));
      when(compositeIntegration.getReviews(PRODUCT_ID_OK)).
         thenReturn(singletonList(new Review(PRODUCT_ID_OK, 1, "author", "subject", "content", "mock address")));

      when(compositeIntegration.getProduct(PRODUCT_ID_NOT_FOUND)).thenThrow(new NotFoundException("NOT FOUND: " +
PRODUCT_ID_NOT_FOUND));

      when(compositeIntegration.getProduct(PRODUCT_ID_INVALID)).thenThrow(new InvalidInputException("INVALID: " +
PRODUCT_ID_INVALID));
   }

   @Test
   public void contextLoads() {
   }
   @Test
   public void getProductById() {

        client.get()
            .uri("/product-composite/" + PRODUCT_ID_OK)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isOk()
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.productId").isEqualTo(PRODUCT_ID_OK)
            .jsonPath("$.recommendations.length()").isEqualTo(1)
            .jsonPath("$.reviews.length()").isEqualTo(1);
   }



   @Test
   public void getProductNotFound() {

        client.get()
            .uri("/product-composite/" + PRODUCT_ID_NOT_FOUND)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isNotFound()
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.path").isEqualTo("/product-composite/" + PRODUCT_ID_NOT_FOUND)
            .jsonPath("$.message").isEqualTo("NOT FOUND: " + PRODUCT_ID_NOT_FOUND);
   }

   @Test
   public void getProductInvalidInput() {

        client.get()
            .uri("/product-composite/" + PRODUCT_ID_INVALID)
            .accept(APPLICATION_JSON)
            .exchange()
            .expectStatus().isEqualTo(UNPROCESSABLE_ENTITY)
            .expectHeader().contentType(APPLICATION_JSON)
            .expectBody()
            .jsonPath("$.path").isEqualTo("/product-composite/" + PRODUCT_ID_INVALID)
            .jsonPath("$.message").isEqualTo("INVALID: " + PRODUCT_ID_INVALID);
   }
}
```