03

Feedback pelo teste

Transcrição

Para que o *JUnit* possa reconhecer o contexto dos objetos do *Spring*, precisamos adicionar uma nova biblioteca em nosso projeto, sendo esta a *Spring Test* por meio da declaração da mesma no arquivo `pom.xml`

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>4.1.0.RELEASE</version>
    <scope>test</scope>
</dependency>
```

Mas apenas isso não é o suficiente. Precisamos dizer agora que o *JUnit* deverá carregar configurações do *Spring Test* para poder executar os testes e fazemos isso realizando duas configurações por anotação, são elas:

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = {JPAConfiguration.class,ProdutoDAO.class})
```

A anotação `@RunWith` do próprio *JUnit* nos permite dizer que classe irá executar os testes encontrados na nossa suite de testes. Já a anotação `@ContextConfiguration` nos permite configurar quais são as classes de configurações para execução dos testes. Como estamos usando conexão com o banco de dados, precisamos da classe que configura a *JPA* e o `ProdutoDAO` neste caso.

Após este passo, podemos transformar o objeto `produtoDAO` em um atributo da classe `ProdutoDAOTest` , anota-lo com `@Autowired` e renomea-lo para `produtoDao` para que fique mais claro a mudança. Por último, precisamos que o método seja anotado com `@Transactional` porque o mesmo precisa de uma transação com o banco de dados para que tudo funcione corretamente. Assim teremos:

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = {JPAConfiguration.class,ProdutoDAO.class})
public class ProdutoDAOTest {

    @Autowired
    private ProdutoDAO produtoDao;

    @Test
    @Transactional
    public void deveSomarTodosOsPrecosPorTipoLivro() {

        List<Produto> livrosImpressos = ProdutoBuilder.newProduto(TipoPreco.IMPRESSO, BigDecima
        List<Produto> livrosEbook = ProdutoBuilder.newProduto(TipoPreco.EBOOK, BigDecimal.TEN).r

        livrosImpressos.stream().forEach(produtoDao::gravar);
        livrosEbook.stream().forEach(produtoDao::gravar);

        BigDecimal valor = produtoDao.somaPrecosPorTipo(TipoPreco.EBOOK);
        Assert.assertEquals(new BigDecimal(40).setScale(2), valor);
```

```
    }

}
```

Outros dois problemas que podem acontecer é o *Spring Test* não conseguir instanciar o objeto `precos` na classe `Produto`. Para garantir que não tenhamos tal problema, alteraremos a declaração deste objeto instanciando um objeto do tipo `ArrayList`, assim teremos na classe produto o seguinte:

```
@Entity
public class Produto {
    // [...]

    @ElementCollection
    private List<Preco> precos = new ArrayList<>();

    // [...]
}
```

Ao tentarmos executar o teste novamente, vemos que funciona, embora o mesmo não tenha passado. Nosso teste espera que o valor retornado seja 40, mas 127 foi o valor retornado do banco de dados.