09

Mãos na massa: Testes

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Para que o JUnit fique disponível no seu projeto, adicione o seguinte XML dentro do seu arquivo **pom.xml** dentro da *tag* `<dependencies>` :

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>4.1.0.RELEASE</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <version>4.0.0.RELEASE</version>
    <scope>test</scope>
</dependency>
```

2) Na classe `ProdutoDAO` , crie o método `somaPrecosPorTipo` :

```
public BigDecimal somaPrecosPorTipo(TipoPreco tipoPreco) {
    TypedQuery<BigDecimal> query = manager
        .createQuery("select sum(preco.valor) from Produto p "
            + "join p.precos preco where preco.tipo = :tipoPreco",
                BigDecimal.class);
    query.setParameter("tipoPreco", tipoPreco);

    return query.getSingleResult();
}
```

3) Crie a classe `ProdutoBuilder` no pacote `br.com.casadocodigo.loja.builders` :

```
public class ProdutoBuilder {

    private List<Produto> produtos = new ArrayList<>();

    private ProdutoBuilder(Produto produto) {
        produtos.add(produto);
    }

    public static ProdutoBuilder newProduto(TipoPreco tipoPreco,
            BigDecimal valor) {
```

```
        Produto livro = create("Livro 1", tipoPreco, valor);
        return new ProdutoBuilder(livro);
    }

    public static ProdutoBuilder newProduto() {
        Produto livro = create("Livro 1", TipoPreco.COMBO,
            BigDecimal.TEN);
        return new ProdutoBuilder(livro);
    }

    private static Produto create(String titulo, TipoPreco tipoPreco,
            BigDecimal valor) {

        Produto livro = new Produto();
        livro.setTitulo(titulo);
        livro.setDataLancamento(Calendar.getInstance());
        livro.setPaginas(150);
        livro.setDescricao("Ótimo livro sobre testes");

        Preco preco = new Preco();
        preco.setTipoPreco(tipoPreco);
        preco.setValor(valor);

        livro.getPrecos().add(preco);

        return livro;
    }

    public ProdutoBuilder mais(int quantidade) {
        Produto base = produtos.get(0);
        Preco preco = base.getPrecos().get(0);
        for (int i = 0; i < quantidade; i++) {
            produtos.add(create("Livro " + i,
                preco.getTipoPreco(), preco.getValor()));
        }
        return this;
    }

    public Produto buildOne() {
        return produtos.get(0);
    }

    public List<Produto> buildAll() {
        return produtos;
    }
}
```

4) Crie a base de dados **casadocodigo_test**:

```
mysql -u root
mysql> create database casadocodigo_test
```

5) No *source folder* **casadocodigo/src/test/java** e no pacote `br.com.casadocodigo.loja.conf`, crie a classe `DataSourceConfigurationTest`:

```
public class DataSourceConfigurationTest {

    @Bean
    @Profile("test")
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setUrl("jdbc:mysql://localhost:3306/casadocodigo_test");
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUsername("root");
        dataSource.setPassword("");

        return dataSource;
    }

}
```

6) Organize melhor a classe **`JPAConfiguration`**, separando o código do *data source* e das *properties* e dois métodos diferentes. Depois, anote o método `dataSource` com `@Profile("dev")`:

```
@EnableTransactionManagement
public class JPAConfiguration {

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource dataSource) {

        LocalContainerEntityManagerFactoryBean factoryBean =
            new LocalContainerEntityManagerFactoryBean();
        factoryBean.setPackagesToScan("br.com.casadocodigo.loja.models");
        factoryBean.setDataSource(dataSource);

        JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
        factoryBean.setJpaVendorAdapter(vendorAdapter);
        factoryBean.setJpaProperties(additionalProperties());

        return factoryBean;
    }

    @Bean
    @Profile("dev")
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        dataSource.setUsername("root");
        dataSource.setPassword("root");
        dataSource.setUrl("jdbc:mysql://localhost:3306/casadocodigo");
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");

        return dataSource;
    }

    private Properties additionalProperties() {
        Properties props = new Properties();
        props.setProperty("hibernate.dialect",
            "org.hibernate.dialect.MySQL5Dialect");
        props.setProperty("hibernate.show_sql", "true");
        props.setProperty("hibernate.hbm2ddl.auto", "update");
```

```
        return props;
    }

    @Bean
    public JpaTransactionManager transactionManager(EntityManagerFactory emf) {
        return new JpaTransactionManager(emf);
    }

}
```

7) Novamente no *source folder* casadocodigo/src/test/java e no pacote `br.com.casadocodigo.loja.daos`, crie a classe de teste `ProdutoDAOTest`:

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

    @Autowired
    private ProdutoDAO produtoDao;

    @Test
    @Transactional
    public void deveSomarTodosPrecosPorTipoLivro() {

        List<Produto> livrosImpressos = ProdutoBuilder
                .newProduct(TipoPreco.IMPRESSO, BigDecimal.TEN)
                .mais(3).buildAll();
        livrosImpressos.stream().forEach(produtoDao::gravar);

        List<Produto> livrosEbook = ProductBuilder
                .newProduct(TipoPreco.EBOOK, BigDecimal.TEN)
                .mais(3).buildAll();
        livrosEbook.stream().forEach(produtoDao::gravar);

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

    }
}
```

8) Na classe `ServletSpringMVC`, crie o método `onStartup` como abaixo:

```
@Override
public void onStartup(ServletContext servletContext) throws ServletException {
    super.onStartup(servletContext);
    servletContext.addListener(RequestContextListener.class);
    servletContext.setInitParameter("spring.profiles.active", "dev");
}
```

9) No *source folder* casadocodigo/src/test/java e no pacote `br.com.casadocodigo.loja.controllers`, crie a classe de teste `ProdutosControllerTest`:

```
@WebAppConfiguration
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes = {JPAConfiguration.class, AppWebConfiguration.class,
    DataSourceConfigurationTest.class, SecurityConfiguration.class})
@ActiveProfiles("test")
public class ProdutosControllerTest {

    @Autowired
    private WebApplicationContext wac;

    @Autowired
    private Filter springSecurityFilterChain;

    private MockMvc mockMvc;

    @Before
    public void setup(){
        this.mockMvc = MockMvcBuilders
                        .webAppContextSetup(wac)
                        .addFilter(springSecurityFilterChain)
                        .build();
    }

    @Test
    public void deveRetornarParaHomeComOsLivros()
            throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.get("/"))
            .andExpect(MockMvcResultMatchers
                .model().attributeExists("produtos"))
            .andExpect(MockMvcResultMatchers
                .forwardedUrl("/WEB-INF/views/home.jsp"));
    }

    @Test
    public void somenteAdminDeveAcessarProdutosForm()
            throws Exception {
        mockMvc.perform(MockMvcRequestBuilders.get("/produtos/form")
            .with(SecurityMockMvcRequestPostProcessors
                .user("user@casadocodigo.com.br").password("123456")
                .roles("USUARIO")))
            .andExpect(MockMvcResultMatchers.status().is(403));
    }

}
```