TESTE E QUALIDADE DE SOFTWARE

João Choma Neto

joao.choma@gmail.com

Unicesumar – Maringá – 2025/2

TESTE INTEGRAÇÃO

TESTE DE INTEGRAÇÃO

- Os sistemas de software geralmente consistem em várias partes ou módulos que precisam trabalhar juntos para fornecer funcionalidade completa.

TESTE DE INTEGRAÇÃO

- Um teste de integração é uma fase de teste de software que se concentra na interação entre diferentes componentes de um sistema ou aplicativo para garantir que eles funcionem juntos de forma eficaz

TESTE DE INTEGRAÇÃO

- O principal objetivo do teste de integração é identificar problemas que possam surgir quando os componentes individuais do software são combinados.

UNIDADE VS INTEGRAÇÃO

- O teste de unidade verifica se cada componente funciona individualmente.
- O teste de integração verificar se os componentes ainda funcionam quando se comunicam com outros componentes.

INTEGRAÇÃO

- Verificação de interfaces: Garantir que os diferentes componentes se comuniquem corretamente e troquem dados da maneira esperada.

- "interface" se refere à maneira como diferentes partes ou componentes de um sistema de software se conectam e interagem entre si

- APIs (Interfaces de Programação de Aplicativos), chamadas de função, mensagens, troca de dados por meio de banco de dados

INTEGRAÇÃO

- Detecção de problemas de interoperabilidade: Identificar conflitos ou inconsistências que possam surgir quando os componentes interagem.

- Validação de fluxo de dados: Certificar-se de que os dados fluem corretamente entre os componentes e que não há perda de informações importantes.

INTEGRAÇÃO

- Teste de funcionalidade combinada: Verificar se as funções e recursos do sistema funcionam conforme o esperado quando os componentes são combinados.

- Identificação de erros de comunicação: Encontrar problemas de comunicação, como atrasos ou perda de dados, que podem afetar o desempenho do sistema.

@Autoriwed

- @Autowired é uma anotação que o Spring usa para injetar dependências automaticamente, ou seja, quando você quer que o Spring forneça um objeto para uma classe sem você precisar instanciá-lo.
- Desde o Spring 4.3, o framework passou a injetar automaticamente os beans quando há apenas um construtor público na classe.

@Autoriwed

Disso

```
@Service
public class ClienteService {

    @Autowired
    private ClienteRepository repo;
}
```

para isso

```
@Service
public class ClienteService {

    private final ClienteRepository repo;

    public ClienteService(ClienteRepository repo) {
        this.repo = repo;
    }
}
```

@Transactional

- Controla transações em banco de dados

- Se tudo der certo, a transação é confirmada (commit).
- Se algo falhar, o Spring desfaz tudo (rollback) automaticamente.

assertEquals vs AsserThat

Aspecto	assertEquals (JUnit)	assertThat (AssertJ)
Sintaxe	assertEquals(expected, actual)	assertThat(actual).isEqualTo(expected)
Legibilidade	Menos natural	Mais fluente e legível
Encadeamento	Não suporta	Suporta (.isNotNull().startsWith("A"))
Mensagens de erro	Simples	Ricas e descritivas
Padrão atual	Antigo (JUnit 4)	Recomendado (JUnit 5 + Spring Boot)

assertEquals vs AsserThat

Método	O que verifica
isNull()/isNotNull()	Nulo ou não nulo
isTrue()/isFalse()	Booleano
isEqualTo(x)/isNotEqualTo(x)	Igualdade
contains(x)/doesNotContain(x)	Elementos em lista
hasSize(n)	Tamanho de lista
startsWith(x)/endsWith(x)	Texto
isInstanceOf(Classe.class)	Tipo de exceção ou objeto

@WebMvcTest

- Permite simular requisições HTTP sem subir o servidor de verdade

ObjectMapper

- ObjectMapper é uma classe da biblioteca Jackson, usada para converter objetos Java ↔ JSON.

@MockBean

@MockBean cria um mock (objeto falso) de um bean gerenciado pelo Spring.

Dicas

- @Autowired → injeta componentes reais configurados pelo Spring Boot no contexto de teste (MockMvc, ObjectMapper, etc.)
- @MockBean → cria componentes falsos (mocks) que simulam o comportamento das dependências reais, permitindo testes isolados da camada web.

ESTUDO DE CASO 01

ESTUDO DE CASO

- Cenário hipotético envolvendo um sistema de comércio eletrônico simples, com componentes de front-end e back-end.

ESTUDO DE CASO

- **Cenário: Teste de Integração em um Sistema de Comércio Eletrônico**
- **Componentes:**
  1. Front-end (Interface do usuário do site de compras).
  2. Back-end (Servidor que gerencia pedidos e produtos).

ESTUDO DE CASO

- **Objetivo do Teste:** Garantir que a interface do usuário do site possa se comunicar corretamente com o servidor de back-end para realizar transações de compra.

ESTUDO DE CASO

- **Caso de Teste de Integração: Caso de Teste 1 - Processo de Compra Bem-Sucedido**
  1. Abra o navegador e acesse a página inicial do site de compras.
  2. Navegue pelo site, selecione um produto e adicione-o ao carrinho de compras.
  3. Clique no carrinho de compras para revisar os itens.
  4. Clique no botão "Finalizar Compra".
  5. Preencha os detalhes do envio e pagamento.
  6. Clique em "Confirmar Pedido".

ESTUDO DE CASO

- **ITENS A VALIDAR:**
  - O produto selecionado deve aparecer no carrinho de compras.
  - Verifique se os detalhes do envio e pagamento foram registrados corretamente.
  - O servidor de back-end deve receber as informações do pedido e responder com uma confirmação de compra bem-sucedida.
  - O usuário deve ver uma confirmação de compra na interface do usuário.

ESTUDO DE CASO

- **Resultado Esperado:**
  - O pedido é registrado com sucesso no sistema.
  - O usuário vê uma confirmação de compra na interface do usuário.

ESTUDO DE CASO 02

ESTUDO DE CASO

- O objetivo deste teste é verificar se o sistema de empréstimo de livros funciona corretamente, incluindo a interação entre a biblioteca (Library) e o membro (Member).

ESTUDO DE CASO

- **Caso de Teste de Integração: Processo de Empréstimo de Livro**
- **Passos:**
  - Inicialização do sistema:
    - Criar uma instância da classe Library.
    - Criar uma instância da classe Member.
    - Criar um livro específico (por exemplo, "The Great Gatsby") e adicioná-lo à biblioteca.
  - Empréstimo de livro:
    - O membro tenta pegar emprestado o livro da biblioteca.

ESTUDO DE CASO

- Verificação:
  - Verificar se o livro foi emprestado com sucesso.
  - Verificar se o livro não está mais disponível na biblioteca.
  - Verificar se o livro está na lista de livros emprestados do membro.
- Tentativa de empréstimo duplicado:
  - O membro tenta pegar emprestado o mesmo livro novamente.
- Verificação:
  - Verificar que o segundo empréstimo é negado (o livro não pode ser emprestado novamente).

ESTUDO DE CASO

- Devolução do livro:
  - O membro devolve o livro à biblioteca.
- Verificação:
  - Verificar se o livro está novamente disponível na biblioteca.
  - Verificar se o livro foi removido da lista de livros emprestados do membro.

ESTUDO DE CASO

- **Resultado Esperado:**
  - No final do teste, o livro deve estar disponível na biblioteca.
  - O primeiro empréstimo deve ter sido bem-sucedido.
  - O segundo empréstimo deve ser negado.
  - A devolução do livro deve ser bem-sucedida.
  - A lista de livros emprestados do membro deve refletir com precisão os empréstimos e devoluções.