02

O padrão Page Object

Transcrição

Na última aula implementamos os cenários de teste da funcionalidade de login, tanto autenticação quanto autorização. Citamos também um assunto importante, que são as boas práticas em programação e a organização do código. Existe um detalhe importante que será o assunto desse vídeo.

Ao analisarmos o código de teste, percebemos que ele está simples e sucinto, mas conforme criarmos mais cenários é possível que passemos por outros problemas de manutenção. Mesmo que tenhamos utilizado alguns métodos do JUnit para executarmos algumas ações antes e depois dos testes, ainda temos código duplicado em relação à API do Selenium.

Por exemplo, precisamos recuperar o campo do usuário na página de login, e para fazer isso estamos utilizando `browser.findElement(By.id())` passando o ID do input (`username`). Esse processo é repetido para `password` e `login-form` , e todos se repetem no próximo cenário.

Poderíamos seguir a mesma ideia de extrairmos essas duplicações, mas isso poderia poluir o nosso código de testes, enchendo-o com constantes que dificultem a legibilidade. Além disso, `LoginTest` é uma classe de testes e, como tal, o foco deveria ser somente códigos relacionados ao JUnit, o nosso framework principal de testes automatizados. Entretanto, estamos misturando a API do Selenium com os nossos testes, desenvolvendo cenários e funcionalidades ao

mesmo tempo em que executamos as verificações do teste em si. Tudo isso piora a legibilidade do nosso código.

O ideal seria buscarmos uma maneira de isolar a API do Selenium dos testes, mantendo somente o passo-a-passo da funcionalidade na classe de teste e deixando de nos preocupar com os detalhes de implementação da API. Outro motivo para isso é que, futuramente, é possível que desejemos mudar para uma biblioteca concorrente que disponha das mesmas funcionalidades. Se fizéssemos isso atualmente, seria necessário substituir todas as classes de teste, gerando uma dificuldade.

Esse foi um problema que os times de desenvolvimento passaram quando começaram a trabalhar no desenvolvimento de diversos testes E2E utilizando o Selenium. Como extrair esse código e separá-lo?

Criou-se então um padrão que hoje é uma recomendação do próprio Selenium: os *Page Objects*, que servem para melhorar a legibilidade do código e facilitar a manutenção. Na documentação do Selenium, é possível encontrar uma página de diretrizes e recomendações (https://www.selenium.dev/documentation/en/guidelines_and_recommendations/). Nela encontramos mais informações sobre esse padrão de projetos que visa separar as responsabilidades no projeto, isolando a API do Selenium, citando problemas e exemplificando os usos dos Page Objects.

Lembre-se que os testes de UI ficam no topo da pirâmide de testes, e que devemos reduzir ao máximo a sua quantidade, já que são difíceis de escrever, lentos de executar e mais caros de manter.

No próximo vídeo começaremos a implementar o Page Object, refatorando o código da nossa funcionalidade de login.