03

Selenium WebDriver

Transcrição

Olá pessoal, boas vindas ao curso de Selenium. Nesta aula falarei um pouco sobre o Selenium e sobre testes automatizados de modo a entendermos os conceitos antes de trabalharmos efetivamente no projeto.

Você já deve conhecer um pouco sobre testes automatizados, e até mesmo escrito alguns testes utilizando o JUnit (no caso do Java). Existem vários tipos de testes que você pode escrever, em especial se for fazer algum tipo de automatização.

Existem os mais simples, que são os "Testes de unidade" - também chamados de "Testes unitários" - nos quais basicamente se utiliza o framework JUnit para testar uma unidade de maneira isolada do resto do projeto. Por exemplo, uma classe que possua métodos com validações ou regras de negócio, e você deseja testá-los de maneira isolada das demais classes. Nesse caso, é possível escrever um teste de unidade - bem simples de escrever e de rápida execução - utilizando apenas o JUnit.

Outra categoria bastante popular é o "Teste de integração". Nele, queremos testar justamente a integração entre duas classes, dois módulos ou dois componentes, entendendo se tal integração entre os elementos está funcionando como esperado. È um teste um pouco mais abrangente, e você pode utilizar outras ferramentas.

Existem também outros testes, como "Teste de segurança", para verificar brechas, falhas ou vulnerabilidades na aplicação; "Teste de performance", para verificar se a aplicação está se comportando de acordo com a performance desejada; e "Teste de aceitação", categoria que também recebe nomes como "Teste de UI (User Interface)" ou "E2E" *(end-to-end*), um teste que simula o usuário utilizando a aplicação, passando por telas de login, preenchimento de formulários, cliques em botões e assim por diante. É um teste que vai de "ponta-a-ponta" na aplicaçãõ, da tela até o banco de dados (caso haja persistência), interagindo com as classes do projeto; justamente por isso, é um pouco mais trabalhoso de escrever, necessita de outras ferramentas e demanda mais tempo de execução.

Existe um conceito conhecido na área de testes automatizados, a "Pirâmide de Teste" ("Test Pyramid"), que estipula uma proporção ideal entre esses tipos de testes dentro de um projeto. Conforme você sobe na pirâmide, os testes vão ficando mais custosos e sua execução e manutenção mais lentas.

Em geral, ela possui três divisões: a base da pirâmide, onde se encontram os testes de unidade, que são mais rápidos e mais baratos de escrever e dar manutenção, portanto devem estar em concentração maior; a faixa central, de testes de serviço e integração entre os componentes; e no topo os testes de aceitação, que são o foco do nosso treinamento. Esses testes estão em menor quantidade, já que sãos difíceis de escrever, necessitam de ferramentas além do JUnit, são mais lentos de executar e sua manutenção é mais custosa.

Os times de desenvolvimento normalmente se baseiam nessa pirâmide para escrever os testes, seguindo as proporções recomendadas pela comunidade.

No mundo Java nós dispomos de diversas ferramentas para escrever esses variados tipos de testes. A principal e mais famosa delas é o JUnit, para testes de unidade, mas também temos o Mockito (utilizado para testes que simulam comportamento, os chamados "mocks"); DBUnit, para testes em que simulamos

ato de popular de um banco de dados; Cucumber, para testes de comportamento com uma linguagem mais natural/humana; e o Selenium, uma ferramenta para testes de interface com usuário. Obviamente existem outras ferramentas, inclusive concorrentes entre elas, mas estas são as mais populares.

O Selenium é uma biblioteca gratuita e opensource para testes end-to-end. É um projeto bem antigo da comunidade, nascido em 2004 e popular ainda hoje. Ao longo do seu desenvolvimento foram surgindo subprojetos, como o Selenium RC e o Selenium WebDriver, que nasceu por volta de 2007 e é focado em escrever testes utilizando linguagem de programação, como Java ou Python. O script escrito nessas linguagens pode ser integrado ao Selenium, responsável pela abstração do protocolo de comunicação com o browser.

Por exemplo, podemos escrever um código em Java e o Selenium dá conta de abrir o navegador, clicar em um botão e preencher um formulário, suportando diversos browsers (Chrome, Firefox, Opera, etc). Além disso, a API que ele fornece é bem simples de ser utilizada, facilitando o desenvolvimento desses testes.

Uma curiosidade: o nome "Selenium" é derivado de um elemento da tabela periódica, e justamente por isso o seu símbolo é o "Se". O time responsável pelo Selenium também foi responsável por uma ferramenta de testes anterior, o Mercúrio, e o Selênio ajuda no combate de envenenamentos por mercúrio.

No próximo vídeo começaremos a conhecer o projeto do curso, e em seguida passaremos a escrever nossos testes automatizados.