PROCESSO DE DESENVOLVIMENTO DOCUMENTAÇÃO DE PROJETO		
Nome do curso: Testes Automatizados	**Aula 6** – Estudo de Mercado	**Responsável**: Ludmila Varela

Olá, seja bem-vindo(a)!

Infelizmente, estamos quase chegando ao fim do nosso curso. Até aqui, você aprendeu o que é e como funciona um teste automatizado a partir da metodologia ágil de software. Além disso, você aprendeu sobre TDD e BDD, colocando em prática os conceitos dessas abordagens.

Esta aula, tem como objetivo mostrar para você as principais ferramentas e frameworks para aplicação de TDD e BDD, na atualidade. Por isso, é importante que você conheça o que é mais utilizado no mercado, levando em consideração diferentes tecnologias, linguagens e negócios.

Para testes de back-end, normalmente, as ferramentas incluem os “test runners”, ou seja, os componentes responsáveis por rodar o teste, relatórios de cobertura e bibliotecas relacionadas. Essas ferramentas são importantes para gerar uma melhor visualização do sistema, por parte da equipe não técnica e do cliente. No contexto de Phyton, por exemplo, temos as seguintes ferramentas:

- **Green:** ferramenta de teste que possui uma boa impressão na saída, para facilitar a leitura e compreensão dos resultados;
- **Requestium:** ferramenta que mescla a biblioteca “Requests” com o Selenium, para facilitar a execução de testes automatizados no navegador;
- **Coverage.py:** ferramenta para medir a cobertura de código ao executar testes;
- **Pytest:** ferramenta de uso facilitado que exige apenas funções para escrever testes, e que já possui um plugin para reportar a cobertura de testes do sistema.

Perceba que, a partir desses exemplos, você tem um conjunto de possibilidades com as ferramentas. A escolha e aplicação dessas ferramentas dependem do contexto do software que está sendo desenvolvido.

Para os testes de front-end, as ferramentas dividem-se de modo que algumas fornecem, para você, apenas uma funcionalidade; e outras, uma combinação delas. Por isso, nesse contexto, normalmente, utiliza-se mais de uma ferramenta, já que a maioria delas possuem “test launchers”, que são responsáveis por iniciar os testes no navegador e ajudam a organizar os seus arquivos de teste. Atualmente, os testes, em geral, são organizados em uma estrutura BDD que suporta o desenvolvimento orientado a comportamento.

É importante ressaltar que também existem as ferramentas de “mocks” de teste. Elas são usadas para falsificar determinados módulos ou comportamentos, para testar diferentes partes de um sistema. Ou seja, elas isolam partes dos testes para capturar seus efeitos colaterais.

Você pode estar se perguntando: “Mas, a partir de tantas possibilidades, como eu devo escolher as ferramentas de teste para o meu projeto?” Bem, comece escolhendo a estrutura de teste e a sintaxe que você gosta, a biblioteca de funções de asserção e, depois, decida como deseja executar os testes. Assim, algumas estruturas, como Jest, Jasmine, TestCafe e Cypress, fornecem apenas algumas das funcionalidades e uma combinação de bibliotecas deve ser usada.

Na maioria dos casos, é interessante que você tenha dois processos de teste diferentes: um para executar testes de unidade e integração; e outro para testes funcionais. Isso ocorre porque os testes funcionais, geralmente, levam mais tempo, especialmente ao executar o conjunto de testes em vários navegadores.

Portanto, outros exemplos de ferramentas para testes front-end são:

- **Jsdom:** simula o ambiente de um navegador, sem executar nada além de javascript simples;
- **Electron:** permite escrever aplicativos desktop de plataforma cruzada, usando Javascript, HTML e CSS;
- **Instabul:** informa quanto do seu código é coberto por testes unitários;

- **Karma:** hospeda um servidor de teste, com uma página da web especial, para executar seus testes no ambiente da página;
- **Chai:** é a biblioteca de asserção mais popular do mercado e possui muitos plugins e extensões;
- **Unexpected:** biblioteca de asserções com sintaxe ligeiramente diferente da do Chai;
- **Sinon.js:** possui estruturas de teste independentes e muito vantajosas para Javascript, que funcionam com qualquer conjunto de testes de unidade;
- **Wallaby:** executa testes relevantes para as alterações do seu código, indicando se alguma coisa falha, em tempo real, ao lado da escrita do código.

Com relação aos frameworks, de maneira que possamos unir testes de unidade e de integração, temos alguns que se destacam:

- **Jest:** Framework criado e mantido pelo Facebook, ganhando popularidade e se tornando a biblioteca mais usada ao longo do ano 2017;
- **Jasmine:** Framework mais consolidado no mercado, possuindo uma enorme quantidade de artigos, ferramentas e suporte em fóruns. A comunidade Angular, o sugere como primeira opção.

Se você precisar escolher uma ferramenta para teste funcional, existe a possibilidade de usar o Selenium, que, hoje, domina o mercado nesse ramo. O Selenium não foi escrito especificamente para teste e pode controlar um navegador para muitos propósitos, expondo um driver que os controla, além de usar suplementos e extensões. Este é diferente do Selenium WebDriver, que pode ser acessado de várias maneiras, usando uma variedade de linguagens de programação. Ah, você pode utilizar o Selenium, importando suas bibliotecas, na automação de algum processo, como para o acesso e seleção de um elemento na página, o que está demonstrado no exemplo a seguir:

```
import   org.junit.Before;
import   org.junit.Test;

public class Teste {
	public static Webdriver driver;

	@Before
	public void setUp() throws InterruptedException {
		driver= new FirefoxDriver();
		driver.get("https://freepdfconvert.com/pt/pdf-word");
		driver.window();
	}

	@Test
	public void findElement() {
		driver.findElement(By.id("clientUpload"));

	}
}
```

Nesta aula, você conheceu um panorama das principais ferramentas para automação de testes. Vale destacar que, para uma melhor fixação do conteúdo, é essencial que você coloque em prática o que aprendeu na aula, pesquisando, por exemplo, mais informações sobre cada ferramenta e framework apresentado aqui, além de outros disponíveis no mercado. Fique à vontade para explorar as possibilidades e proceder da melhor forma que se adeque às suas necessidades.

Por hoje é só! Bons estudos!

Referências

First Steps in Frontend Testing with TDD/BDD. Disponível em: <https://medium.com/@aeh.herman/first-steps-in-frontend-testing-with-tdd-bdd-7ddab8796ad6 >. Acesso em: 27 mai. 2020.

Top 15 UI Test Automation Best Practices You Should Follow. Disponível em: <https://www.blazemeter.com/blog/top-15-ui-test-automation-best-practices-you-should-follow >. Acesso em: 27 mai. 2020.