AUTOMAÇÃO DE TESTES COM CYPRESS

PÓS EM DESENVOLVIMENTO DE APLICAÇÕES WEB

MÓDULO TESTE DE SOFTWARE PARA SISTEMAS WEB

MSc. Rafael Ramos

2024

UNIPÊ
Centro Universitário
de João Pessoa

SUMÁRIO

1. O que é o Cypress
2. 10 Motivos para utiliza o Cypress
3. Ambientação
4. Práticas

O QUE É O CYPRESS

“Você pode escrever não apenas testes e2e, mas também testes de unidades, de componentes, de API, testes de regressão visual, e até mesmo uma combinação de todos eles.”

O Cypress.io é um framework de testes automatizados end-to-end, que usa o JavaScript e é executado no mesmo ciclo de execução da aplicação. Vale ressaltar, que por trás do Cypress existe um processo do servidor Node.js. O Cypress e o processo Node.js se comunicam, sincronizam e executam tarefas constantemente. Ter acesso a ambas as partes (back e front) dá ao Cypress uma capacidade de responder aos eventos da aplicação em tempo real, enquanto ao mesmo tempo, permite que o framework trabalhe fora do navegador em tarefas que exigem um privilégio mais alto.

10 MOTIVOS PARA UTILIZAR O CYPRESS

1. Zero configuração para começar.
2. Baixa curva de aprendizado.
3. Excelente experiência de desenvolvimento.
4. Tudo já vem empacotados para sair usando.
5. Esperas automáticas.
6. Controle do tráfego de rede.
7. Diversos tipos de testes automatizados.
8. Documentação Robusta.
9. Ajuda das(os) frontend developers.
10. Comunidade.

CYPRESS VS. SELENIUM

- **Qual é a principal diferença entre Selenium e Cypress?**

A principal diferença entre o Selenium e o Cypress é a abordagem de teste. O Selenium é baseado na abordagem tradicional de simular as interações do usuário manipulando o navegador, enquanto o Cypress adota uma abordagem mais moderna de executar testes no mesmo contexto do aplicativo que está sendo testado.

- **O que é mais fácil de aprender Cypress ou Selenium?**

O Cypress é geralmente considerado mais fácil de aprender do que o Selenium. Ele tem uma API mais simples e uma interface de teste mais intuitiva.

- **Qual é mais rápido Cypress ou Selenium?**

O Cypress é geralmente mais rápido que o Selenium. O Cypress executa testes no mesmo contexto do aplicativo que está sendo testado.

CYPRESS VS. SELENIUM

- **Quem tem melhor compatibilidade Selenium e Cypress?**

  O Selenium tem melhor compatibilidade com o navegador do que o Cypress. O Selenium oferece suporte a uma ampla variedade de navegadores.

- **Qual é mais adequado para testar aplicativos da Web modernos em Cypress ou Selenium?**

  O Cypress geralmente é mais adequado para testar aplicativos da Web modernos, pois além de possibilitar a realização de testes end-to-end, também permite a criação de testes de API.

"Uma diferença fundamental é que o Cypress como ferramenta é ideal para apresentar aos desenvolvedores a automação de testes, em vez de apenas um substituto para o Selenium. É por isso que o Cypress está entre as ferramentas de automação que mais crescem no mundo. Por outro lado, o Selenium é uma ferramenta de uso mais geral voltada para um público mais amplo."

AMBIENTAÇÃO

- Configuração:
  - Pré-requisito: Possuir o Node.js e o Npm instalados em sua máquina.
  - Comandos para verificar a versão do Node e Npm: node –v && npm -v

Link para baixar o node.js

https://nodejs.org/en/download

AMBIENTAÇÃO

- Configuração:
  - Execute o comando abaixo para inicializar o projeto node no diretório desejado.

```
npm init
```

  - Em seguida, configure o package.json com as informações básicas (Descripition, Keywords, License – MIT, Test command – cypress run).

AMBIENTAÇÃO

- Configuração:
  - Não há servidores, drivers ou quaisquer outras dependências para instalar ou configurar. Você pode escrever seu primeiro teste de aprovação em 60 segundos. Para isso, basta instalar o Cypress executando o comando abaixo.

```
npm install cypress --dev
```

  - Após instalar o Cypress, execute o comando:

```
npx cypress open
```

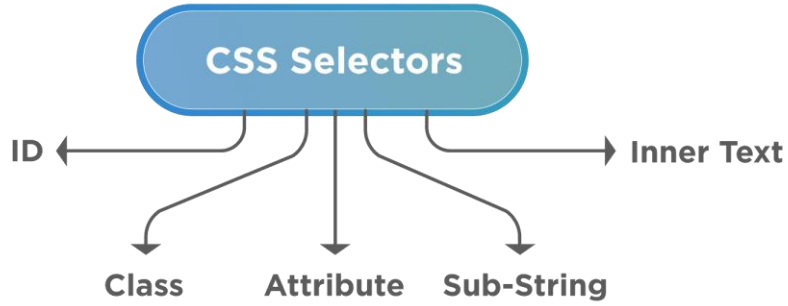
SELETORES CSS

- Em CSS, os seletores são usados para direcionar os elementos HTML em nossas páginas Web. Há uma grande variedade de seletores CSS disponíveis, permitindo uma precisão refinada ao selecionar elementos a serem estilizados.
- Um seletor CSS é a primeira parte de uma regra CSS. É um padrão de elementos e outros termos que informam ao navegador quais elementos HTML devem ser selecionados para que os valores de propriedades CSS dentro da regra sejam aplicados a eles.

```
<input class="search_query form-control ac_input" type="text" id="search_query_top" name="search_query"
placeholder="Search" value autocomplete="off"> == $0
```

SELETORES CSS

- Ao contrário do Selenium, o Cypress suporta apenas seletores CSS (muito semelhantes aos Seletores JQuery). Portanto, usamos seletores CSS em todo o nosso código de automação Cypress.
- Cypress também suporta seletores Xpath. No entanto, isso não vem por padrão. Em outras palavras, precisamos dos plug-ins externos 'Cypress-Xpath' para auxiliar este seletor.
- Podemos escrever seletores CSS de várias maneiras. Por exemplo, usando o id, class ou qualquer outro atributo de um elemento da web que pode nos ajudar a escrever seletores CSS personalizados.



ALGUNS COMANDOS

- cy.visit(url)
- cy.get(selector)
  - .type(text)
  - .click()
  - .contains(content)
  - .should(chainers)

- cy.visit("http://www.google.com.br")
- cy.get('#first-name')
  - cy.get('#first-name').type('Rafael')
  - cy.get('.btn').click
  - cy.get('.nav').contains('About')
  - cy.get('Login').should('be.visible')

PRÁTICA I

Projeto exemplo

- Vamos começar com um teste simples, que navega para um aplicativo de compra e venda de ingressos.
  - O teste e2e em Cypress, deve realizar os seguintes passos:
    - Navegar até a URL https://ticket-box.s3.eu-central-1.amazonaws.com/index.html
    - Preencher campo First Name
    - Preencher campo Last Name
    - Validar se First Name e Last Name são exibidos em Purchase Agreement.

PRÁTICA I

```
describe ("Tickets", () =>{

    beforeEach(() => cy.visit("https://ticket-box.s3.eu-central-1.amazonaws.com/index.html"));

    //Interação com Inputs
    it(“Input First Name and Last Name", () => {
        const firstName = "Rafael";
        const lastName = “Ramos";
        const fullName = `${firstName} ${lastName}`;
        cy.get("#first-name").type(firstName);
        cy.get("#last-name").type(lastName);
        cy.get(".agreement p").should("contain",`I, ${fullName}, wish to buy 2 VIP tickets.`);
      });
});
```

PRÁTICA II

Efetue a compra de Tickets com sucesso, preenchendo todos os campos apresentados no formulário.

Desafios:

- Preencher o campo Special Requests com um texto longo e busque otimizar o tempo de digitação para que o teste tenha melhor performance.
- Tente otimizar o preenchimento dos campos mandatórios, criando um comando Cypress personalizado.

PRÁTICA III

Crie um teste chamado faz uma requisição HTTP - GET. Tal teste deve fazer uso da funcionalidade cy.request(), para fazer uma requisição do tipo GET, para a seguinte URL: https://httpbin.org/get

Com a resposta da requisição, verifique que o status retornou 200, o statusText retornou OK.

PRÁTICA IV

Crie um teste chamado faz uma requisição HTTP - POST. Tal teste deve fazer uso da funcionalidade cy.request(), para fazer uma requisição do tipo POST, para a seguinte URL: https://httpbin.org/post.

Deverá ser informado no campo na requisição os dados (name e age) e com a resposta , verifique que o status retornou 200, o statusText retornou OK.

PRÁTICA V

Crie uma pipeline de testes automatizados. Vá até o GitHub e veja sua mudança disparando o pipeline (e se tudo der certo, veja seus testes passando).

DESAFIO

- Primeiramente acesse e conheça a aplicação CAC TAT (https://cac-tat.s3.eu-central-1.amazonaws.com/index.html), que trata-se de um formulário para simular o envio de mensagens à uma central de atendimento ao cliente.
- Em seguida, crie um Teste funcional automatizado utilizando Cypress para que seja possível enviar com sucesso um chamado para a central de atendimento. Para isto, todos os campos obrigatórios devem ser preenchidos e também deverá ser anexado um arquivo ao seu chamado através do campo “Choose File”.

OBRIGADO!

RAFAEL RAMOS

Software Quality Assurance
rafael.anderson@assert.ifpb.edu.br
+55 83 994053557