Cypress

Automação E2E com JavaScript

CTG-Brasil 2020
Leandro Wanderley

Agenda

1. O que é Cypress?
2. Por que Cypress?
3. Configurando o ambiente.
4. Características.
5. Principais comandos
6. Executando os testes

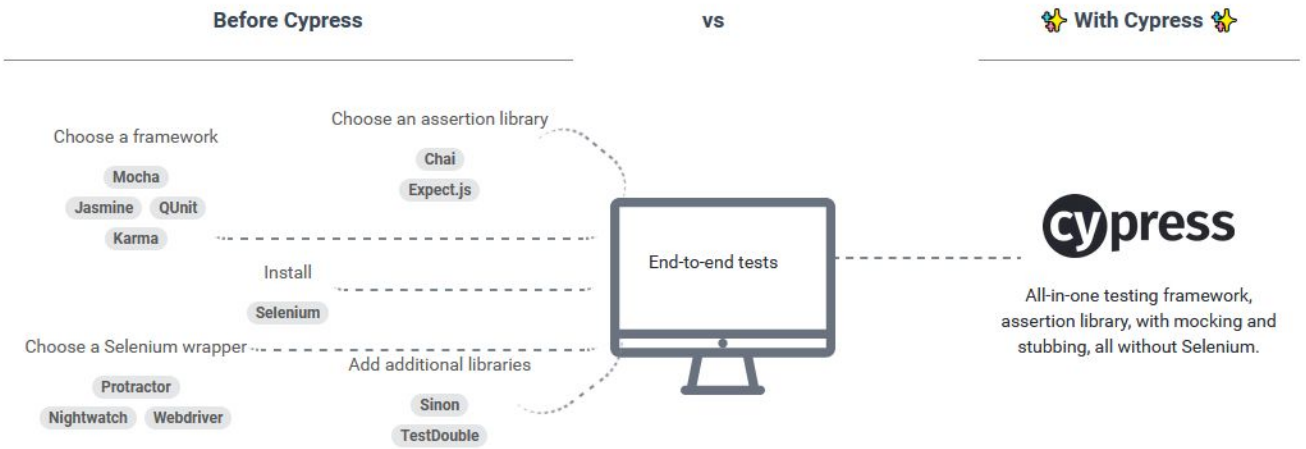
1. O que é Cypress?

1. O que é Cypress?

O ***Cypress*** é um *framework* para automação de testes *end-to-end* para *web* utilizando *JavaScript*, que ganhou muita popularidade nos últimos anos. É *open source* e possui uma comunidade bastante ativa.

2. Por que Cypress?

Você pode estar pensando, *“Mas já existe o Selenium e todo mundo usa, por que eu utilizaria outra ferramenta?”*



2. Por que Cypress?

Dentre as principais características do *Cypress* temos:

- Tempo automático de espera para carregamento de elementos no site; ⏰
- Carregamento das páginas e elementos em tempo real; ⏳
- *Debugabilidade* do teste em execução; 💻
- Resultados consistentes e simples; 📄
- *Screenshots* e vídeos de toda a execução. 📷

3. Configurando o ambiente

1. Escolha uma IDE, eu costumo usar o *VS Code*;
2. Instale o *Node.js*;
    a. Após instalar o *Node.js* verifique se está funcionando corretamente, abra o terminal em seu computador e digite: ***node -v*** e ***npm -v***
3. Usando o *Visual Studio Code*, em *File > Open Folder...* em seguida usando o terminal da *IDE* vamos executar o comando: ***npm init -y***
4. Agora vamos instalar o *Cypress* na pasta do projeto:

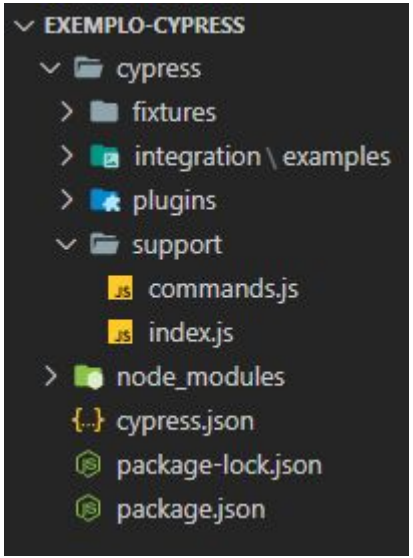
npm install cypress --save-dev

4. Características

Com isso instalamos o *Cypress Desktop App* e o *Cypress CLI*. O primeiro é a interface gráfica usada para executar os testes em um *Browser* e o segundo possibilita executar os testes via linha de comando, sem abrir a interface gráfica e nem *o Browser*.

npx cypress open

4. Características

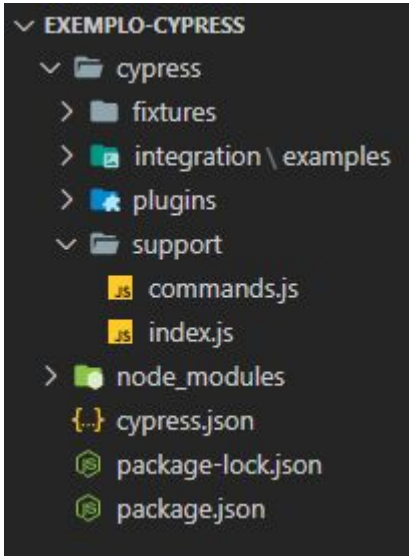


fixtures - É a pasta que contém todos os arquivos necessários durante a execução dos testes. Por exemplo; precisamos de credenciais de login como nome de usuário e senha para fazer o login, que são mantidos dentro desta pasta como um arquivo.

integration - Esta pasta de integração inclui todos os arquivos de teste. Podemos criar quantas pastas quisermos e também pode ser uma pasta aninhada.

plugins - Esta contém o arquivo index.js. Se tivermos de manipular o comportamento interno do Cypress, essa é a pasta. Onde plugins são declarados para serem executados em um servidor Node.

4. Características



support - Esta pasta contém index.js e commands.js. O arquivo index.js é executado antes de cada arquivo de teste. E o arquivo commands.js, nele podemos armazenar o código reutilizável e usar em todos os arquivos de teste.

cypress.json - Este arquivo é o arquivo de configuração do Cypress, nele temos algumas configurações padrão, como tempo padrão de espera, chromeWebSecurity e assim por diante, que podem ser substituídas.

package.json - Este é um arquivo que contém o nome e a versão de todas as dependências exigidas em nosso projeto.

5. Principais comandos

Como em qualquer elaboração de casos de testes automatizados devemos ter como base os seguintes passos:

1. Acessar um site ✅
2. Procurar por elementos na tela ✅
3. Realizar algumas ações com esses elementos ✅
4. E fazer algumas assertivas ✅

5. Principais comandos

Utilizando Cypress devemos criar uma estrutura como essa:

```
/// <reference types="cypress" />
describe('PLANO DE TESTE OU GRUPO DE TESTE', () => {
//Todos os testes estarão aqui!
        it('NOME DO CASO DE TESTE', () => {
                //Código do teste aqui!
        })
})
```

5. Principais comandos

Seguindo os passos anteriores, e usando Cypress, temos:

1. Acessar um site ✅

    cy.visit(‘*URL OU URI DO SITE***’)**

2. Procurar por elementos na tela ✅

    cy.get(‘*ELEMENTO NO HTML/CSS DO SITE***’)**

    cy.contains(‘*TEXTO/CONTEÚDO DENTRO DO SITE***’)**

5. Principais comandos

Seguindo os passos anteriores, e usando Cypress, temos:

3. Realizar algumas ações com esses elementos ✅
    - cy.get().**click()**
    - cy.get().**type(**'*STRING*'**)**
4. E fazer algumas assertivas ✅
    - Implícitas:
        - cy.get().**should(**'*CONDIÇÃO*', '*VALOR*'**).and(**'*CONDIÇÃO*', '*VALOR*'**)**
    - Explícitas:
        - **BDD: expect(**'*ELEMENTO*'**).to.equal(**'*VALOR*'**)** ou **expect(**'*ELEMENTO*'**).to.exist()**
        - **TDD: assert.equal(**'*ATUAL*', '*ESPERADO*', '*MENSAGEM*'**)** ou **assert.isTrue(**'*VALOR*', '*MENSAGEM*'**)**

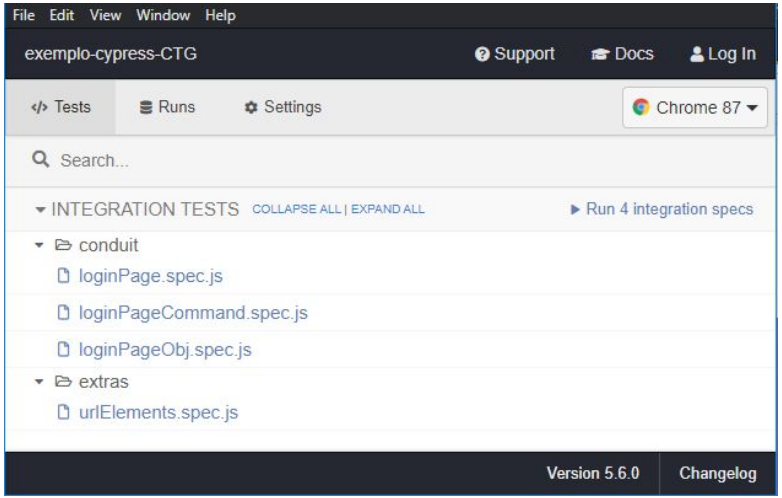
5. Executando os testes

Existem duas formas de executar seus testes usando o Cypress:

1. Usando o *Cypress Desktop App*

    Interface gráfica, acessada através do comando:

    npx cypress open



5. Executando os testes

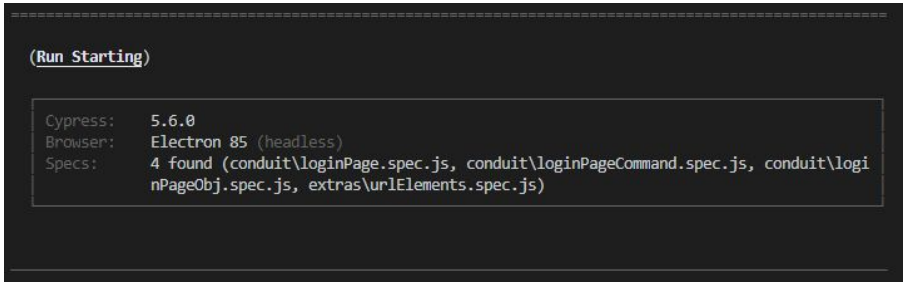
2. Usando o *Cypress CLI*

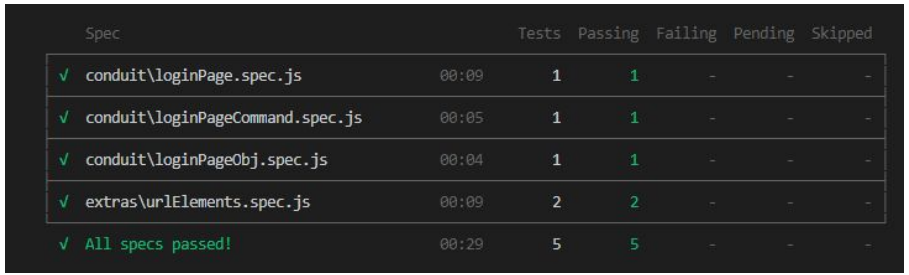
Através do terminal usando linhas de comando:

npx cypress run

Opções:

--browser *<BROWSER>*
--spec *<ARQUIVO ESPECÍFICO>*
--reporter *<PLUGIN DE RELATÓRIO>*





Referências

https://docs.cypress.io/guides/overview/why-cypress.html

https://example.cypress.io/

https://github.com/leandrowcs/exemplo-cypress-CTG

https://medium.com/@leandrowcs1985/vamos-falar-de-automa%C3%A7%C3%A3o-de-testes-web-cypress-6dfddfcb8d7d

Cypress

Automação E2E com JavaScript

Obrigado!

CTG-Brasil 2020
Leandro Wanderley