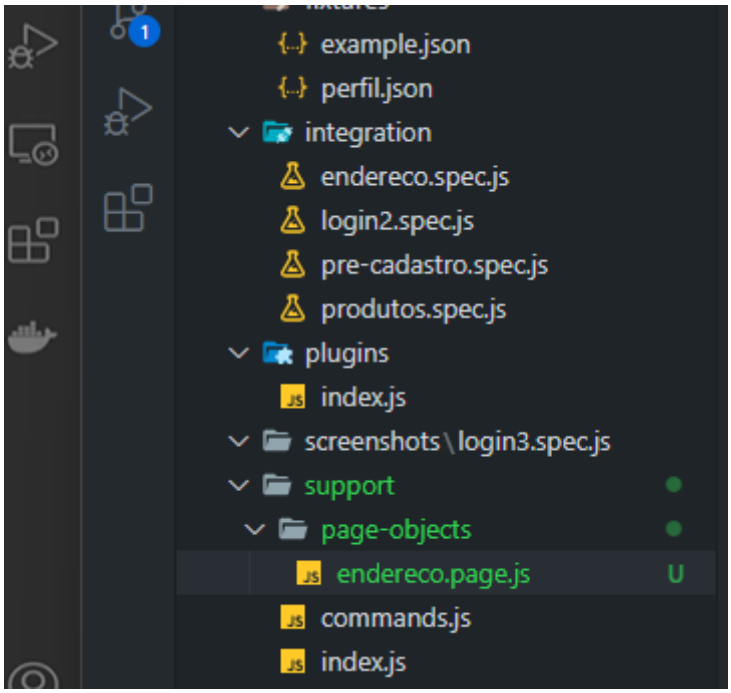
Page Objects

PageObject é um padrão de design que ajuda a aprimorar a manutenção de testes e reduzir a duplicação de código, também pode ser utilizado para descrever e documentar o fluxo de uma aplicação.

Vamos ver na prática. Abra o projeto da aula anterior, vamos mexer no arquivo endereço.spec.js. Na pasta "Suporte" crie uma pasta de nome "page-objects" e dentro dessa pasta um arquivo de nome "endereço.page.js" conforme imagem abaixo:



Vamos criar uma classe dentro do arquivo endereço.page.js assim:

```
class EnderecoPage {

}

export default new EnderecoPage()  // A propriedade exporte serve para
classe ficar exposta a outros arquivos
```

Os {} colchetes significam objeto.

Dentro dos {} colchetes vamos inserir os métodos:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        //serão colocados os elementos + ações
    }

    editarEnderecoEntrega(){
        //serão colocados os elementos + ações
    }
}

export default new EnderecoPage()  // A propriedade exporte serve para
classe ficar exposta a outros arquivos
```

Salva o arquivo e agora no arquivo endereço,spec.js precisamos adicionar o código que chama (importa) a classe criada:

```
import EderecoPage from '../support/page-objects/endereco.page'
```

O código acima tem de estar no começo do arquivo.

Crie o teste usando a classe, assim:

```
it.only('Deve fazer cadastro de faturamento com sucesso', () => {
        enderecoPage.editarenderecoFaturamento()
    });
```

Use o .only para rodar apenas esse teste

O código todo ficará assim:

```
/// <reference types="cypress"/>
import enderecoPage from '../support/page-objects/endereco.page';

describe('Funcionalidade Endereços - Faturamento e Entrega', () => {
    beforeEach(()=>{
        cy.visit('Minha conta') // vai acessar a página de login
        cy.fixture('perfil').then(dados =>{
            cy.login(dados.usuario, dados.senha)
        })

    });

    it.only('Deve fazer cadastro de faturamento com sucesso', () => {
        enderecoPage.editarenderecoFaturamento()
    });
});
```

Execute o cypress:

Npx cypress open

Agora vamos capturar os elementos da página começando com o elemento “Endereço”

Copia o código do cypress e cole no código do arquivo endereço.page.js assim:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address > a').click()
    }
```

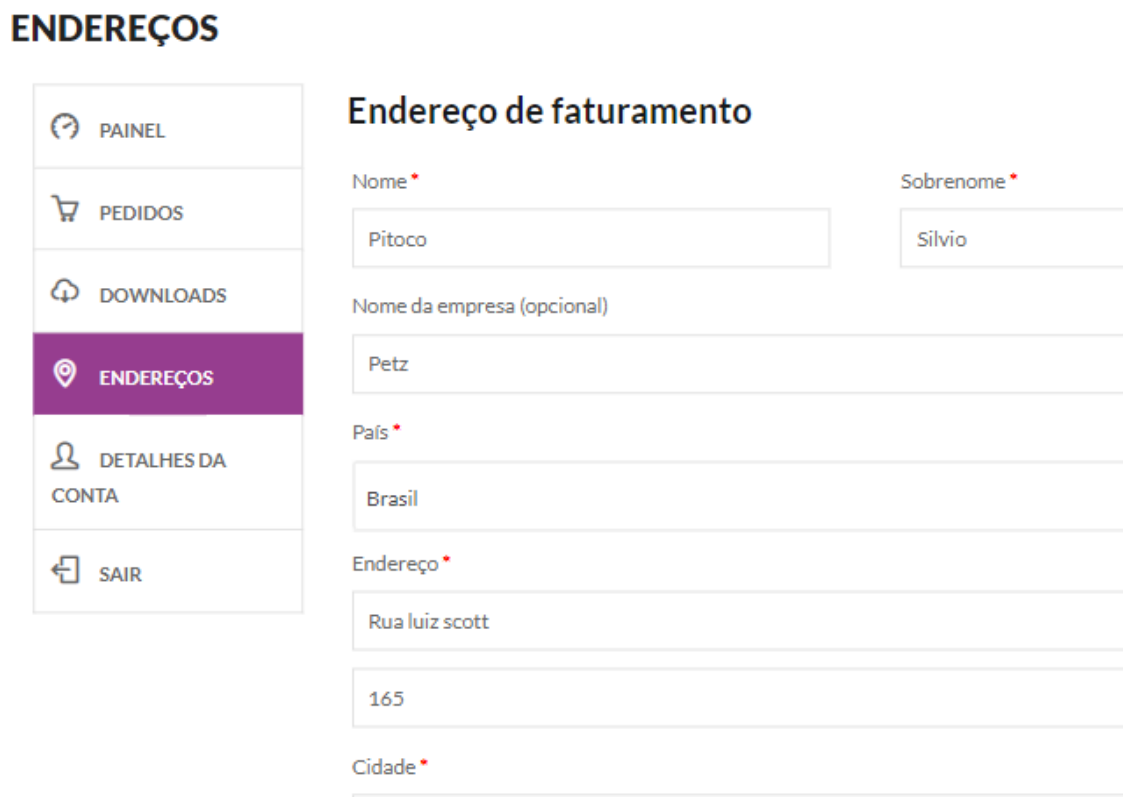
Não esquece de acrescentar o .click() pois temos que clicar para acessar a página de endereços.

Agora adicione o código do elemento “edit” que ficará assim dentro da classe:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address > a').click()
        cy.get(':nth-child(1) > .title > .edit').click()
    }
```

Também tem que adicionar o .click() Salva e roda para testar.

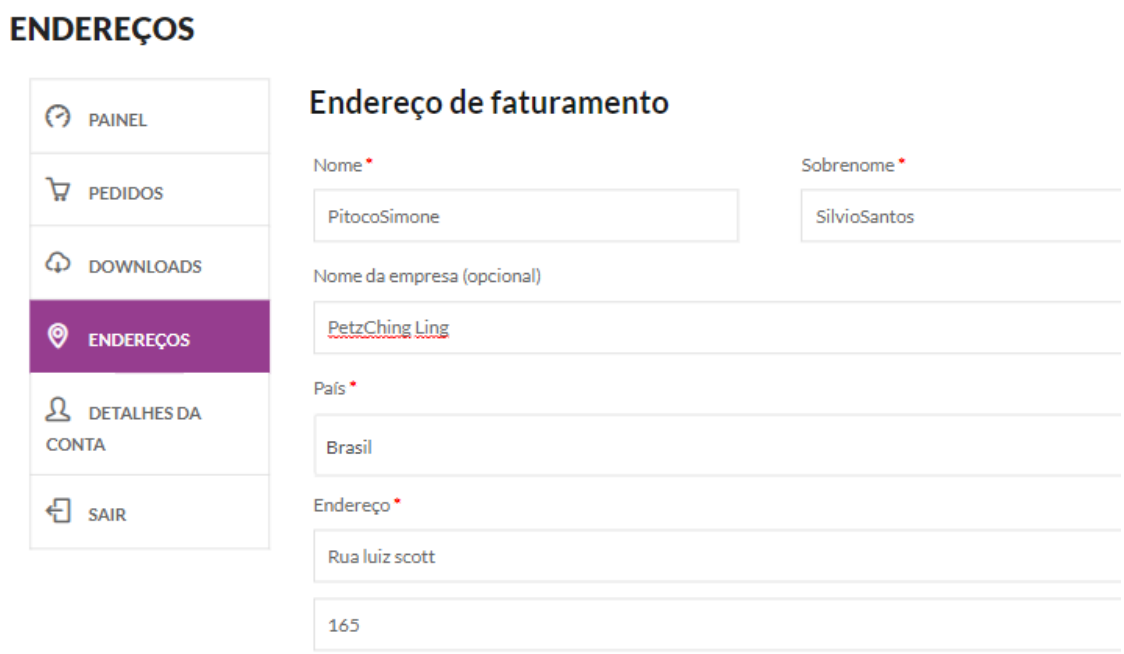
Funcionando, agora vamos capturar os elementos correspondentes ao endereço, pegando o código de cada campo:



Começando pelo “Nome” depois “Sobrenome”, “Nome da empresa”, assim:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address > 
a').click()// link endereço
        cy.get(':nth-child(1) > .title > .edit').click()// botão edit

        cy.get('#billing_first_name').type('Simone')// campo nome
        cy.get('#billing_last_name').type('Santos')// sobrenome
        cy.get('#billing_company'),type('Ching Ling')// nome da empresa

    }
```

Roda o teste até aqui pra ver se está funcionando.



Repare que os campos já continham dados e ele adicionou os dados novos junto. Adicione o .clear() antes do type em todos os elementos, assim ele vai limpar o campo antes de inserir os dados:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address > a').click()// link endereço
        cy.get(':nth-child(1) > .title > .edit').click()// botão edit

        cy.get('#billing_first_name').clear().type('Simone')// campo nome
        cy.get('#billing_last_name').clear().type('Santos')// sobrenome
        cy.get('#billing_company').clear().type('Ching Ling')// nome da empresa

    }
```

Agora vamos inserir o código referente ao campo “Pais” que tem uma particularidade.

O campo onde se digita o pais precisa ser clicado e depois selecionado o pais, portanto temos que usar o código abaixo:

```
 cy.get('#select2-billing_country-container').click().type('Brasil')
```

Se rodar o teste até aqui vai funcionar, porém vai ficar faltando a parte de selecionar o páis. Se utilizarmos o código que o cypress informa não vai ser muito funcional, pois ele vai pegar o final BR do pais Brasil, mas e quando for selecionar outro pais, por exemplo Canadá que é CA vai dar erro no teste.

Nesse caso devemos inspecionar o código. Abra uma nova aba com o site, clique com o botão direito e escolha a opção inspecionar e copie o código correspondente as seleções dos países. Copiei o código conforme o vídeo da aula 3, ficou assim:

```
  cy.get('#select2-billing_country-
container').click().type('Brasil').get('[aria-selected="true"]').click()
```

Foi mais complicado com esse elemento, mas é necessário estar familiarizado com o front end antes da criação dos testes (palavras do professor)

Agora adicione os outros campos conforme o código abaixo:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address >
a').click()// link endereço
        cy.get(':nth-child(1) > .title > .edit').click()// botão edit

        cy.get('#billing_first_name').clear().type('Simone')// campo nome
```

```
        cy.get('#billing_last_name').clear().type('Santos')// sobrenome
        cy.get('#billing_company').clear().type('Ching Ling')// nome da
empresa
        cy.get('#select2-billing_country-
container').click().type('Brasil').get('[aria-
selected="true"]').click()//pais
        cy.get('#billing_address_1').clear().type('Rua dos
Carajás')//endereço
        cy.get('#billing_address_2').clear().type('15') //número
        cy.get('#billing_city').clear().type('Itapevi') // cidade

    }
```

O próximo campo também é diferente para se adicionar, que é o campo "Estado"

Primeiro copie o código do cypress:

```
cy.get('#select2-billing_state-container').click().type('São Paulo')
```

Adicione o .click() e o .type('São Paulo'). Rode para testar até aqui.

O teste funcionou e agora falta a parte que se clica no nome do estado, acontece que a página também aceita ações do teclado. Nesse caso basta adicionar a palavra {enter} entre chaves assim:

```
 cy.get('#select2-billing_state-container').click().type('São
Paulo{enter}')
```

Salva e roda o teste que irá funcionar!!

Lembre-se que você pode se deparar com um front end que não aceita ações do teclado, este foi apenas um exemplo de formas diferentes de fazer o mesmo teste.

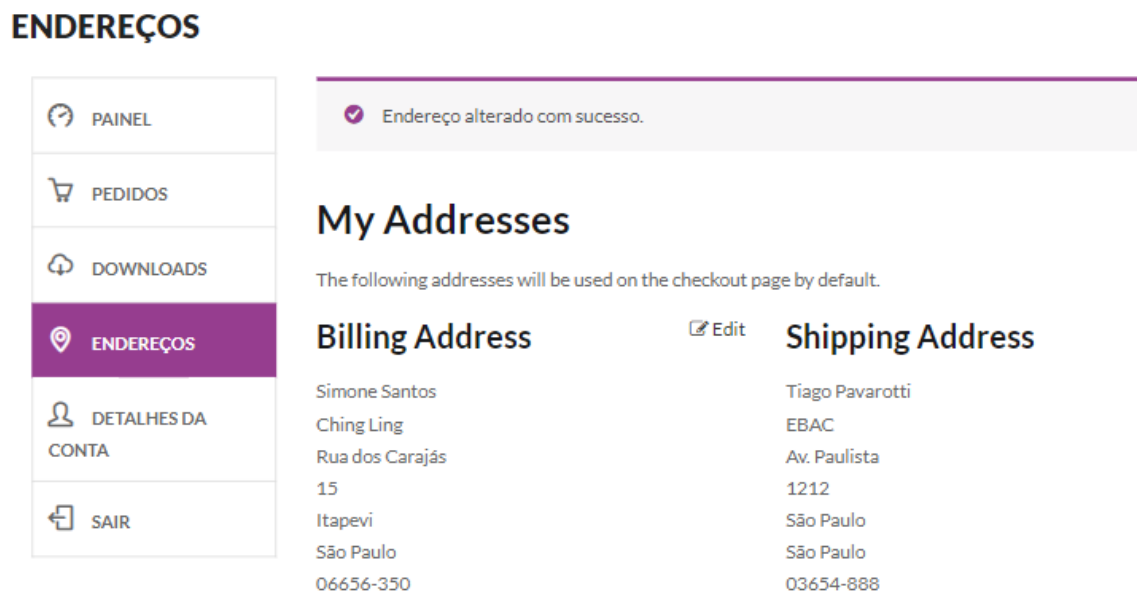
Agora termine de adicionar os outros elementos, o código ficará assim:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address >
a').click()// link endereço
        cy.get(':nth-child(1) > .title > .edit').click()// botão edit

        cy.get('#billing_first_name').clear().type('Simone')// campo nome
        cy.get('#billing_last_name').clear().type('Santos')// sobrenome
        cy.get('#billing_company').clear().type('Ching Ling')// nome da
empresa
        cy.get('#select2-billing_country-
container').click().type('Brasil').get('[aria-
selected="true"]').click()//pais
        cy.get('#billing_address_1').clear().type('Rua dos
Carajás')//endereço
        cy.get('#billing_address_2').clear().type('15') //número
```

```
        cy.get('#billing_city').clear().type('Itapevi') // cidade
        cy.get('#select2-billing_state-container').click().type('São
Paulo{enter}') //estado
        cy.get('#billing_postcode').clear().type('06656350')// cep
        cy.get('#billing_phone').clear().type('11973571191')// telefone
        cy.get('#billing_email').clear().type('email@dominio.com')//
email
        cy.get(':nth-child(2) > .button').click()// botão salvar

    }
```

Salve e rode o teste, todo esse código acima se refere ao método criado dentro da classe que está no arquivo endereço.page,js e que é chamado no arquivo endereço.spec.js.

Segue o resultado:



O teste é feito com apenas uma linha de comando no arquivo endereço.spec.js

```
it.only('Deve fazer cadastro de faturamento com sucesso', () => {
        enderecoPage.editarenderecoFaturamento()
    });
```

Para completar adiciona a comprovação de que o teste passou, o assert no arquivo endereço.spec.js ficará assim:

```
 it.only('Deve fazer cadastro de faturamento com sucesso', () => {
        enderecoPage.editarenderecoFaturamento()
        cy.get('.woocommerce-message').should('contain', 'Endereço
alterado com sucesso') //assert
    });
```

Salve e rode o teste.

Agora para os dados não ficarem fixos, vamos usar as boas práticas de teste (no arquivo endereço.page.js), usando parâmetros, mude os dados no código adicionando parâmetros, ficará assim:

```
class EnderecoPage {
    editarenderecoFaturamento(){
        cy.get('.woocommerce-MyAccount-navigation-link--edit-address > a').click()// link endereço
        cy.get(':nth-child(1) > .title > .edit').click()// botão edit
        cy.get('#billing_first_name').clear().type(nome)// campo nome
        cy.get('#billing_last_name').clear().type(sobrenome)// sobrenome
        cy.get('#billing_company').clear().type(empresa)// nome da empresa
        cy.get('#select2-billing_country-container').click().type('Brasil').get('[aria-selected="true"]').click()//pais
        cy.get('#billing_address_1').clear().type(endereco)//endereço
        cy.get('#billing_address_2').clear().type(numero) //número
        cy.get('#billing_city').clear().type(cidade) // cidade
        cy.get('#select2-billing_state-container').click().type(estado+'{enter}') //estado
        cy.get('#billing_postcode').clear().type(cep)// cep
        cy.get('#billing_phone').clear().type(telfone)// telefone
        cy.get('#billing_email').clear().type(email)// email
        cy.get(':nth-child(2) > .button').click()// botão salvar

    }
```

Agora passe os parâmetros dentro do parênteses depois do nome do método, assim:

```
class EnderecoPage {
    editarenderecoFaturamento(nome, sobrenome, empresa, endereco, numero, cidade,estado, cep, telefone, email )
```

Agora no arquivo de teste endereço.spec.js adicione os nomes correspondentes aos parâmetros:

```
 it.only('Deve fazer cadastro de faturamento com sucesso', () => {
        enderecoPage.editarenderecoFaturamento('Simone', 'Santos', 'Google', 'Carajas', '15', 'Itapevi', 'São Paulo', '06656350', '11973571191', 'wirus1@gmail.com')
        cy.get('.woocommerce-message').should('contain', 'Endereço alterado com sucesso') //assert
    });
```

Salve e rode o teste.

Pergunta do fórum:

Page Object e Comandos Customizados possuem a mesma função?

Olá Oséias, tudo bem?

São conceitos diferentes, explicados em aula. O page Objects é focado em páginas, tipo login, produtos, checkout, etc. O comando customizado esta focado em métodos, funções, exclusivo do cypress para encapsular uma função, como login, adicionar Produtos, realizar pagamento, etc.

Sempre bom ter a estratégia em mente para escolher qual usar.

Abs,

Fábio Araújo