

Locators customizados

Transcrição

[0:00] Bom, para terminarmos essa parte da Home, já testamos login, agora precisamos testar a parte do cadastro, para terminar de encobrir essa parte da home. Certo?

[0:09] Então, se formos na parte do sign up, vemos aqui que tem a parte do registro, tem quatro campos de formulário, e-mail, nome completo, nome do usuário e senha.

[0:21] E tem o botão de registrar.

[0:25] O que precisamos fazer, então?

[0:27] Basicamente o que fizemos na tela de login, certo?

[0:31] Então, vou voltar para o nosso Vs Code, aqui, vou abrir o code.

[0:36] E, lembrando, como agora a gente vai testar a parte de criação de usuários seria ideal já, que você já tenha aberto aqui, e executando sua API.

[0:47] Tudo bem?

[0:48] Então tá bom.

[0:50] Executando a API, voltamos aqui ao nosso e2e, e temos essa estrutura: signin.po.ts e signin.e2e.

[1:02] Bom, a signin.e2e, fazia sentido por que a gente só estava testando a parte do signin, não é?

[1:08] Agora vamos deixar com que esse nome fique mais genérico, por exemplo, como a gente esta testando a parte da home, e a home tem por baixo dela tanto a parte de login como a parte de registro, seria ideal que a gente fizesse o que?

[1:23] Mudasse esse nome para: home.

[1:27] Certo?

[1:28] E como vamos começar a criar nossos page object, mais de um, o que podemos fazer?

[1:35] Dentro da pasta search, na pasta hierarquia e2e SRC, vamos criar, botão direito em cima da pasta SRC, nova pasta e digitar page object.

[1:52] Tudo bem?

[1:53] E vamos jogar para dentro do nosso, da nossa pasta page object, nosso signin.po.ts, perfeito?

[2:02] Aqui ele já alterou para nós o nosso import, mas, nada impede de você vir aqui e colocar ./page object / signin.po para atualizar a localização do nosso page object dentro da home.e2e/spec.

[2:24] Tranquilo?

[2:25] Então, vamos criar nosso page object.

[2:26] New File signup.pageobject.ts e vamos fazer a mesma coisa, vamos dar os imports.

[2:37] Opa.

[2:38] Os imports do browser from protractor, mesma coisa que fizemos no signin.po.

[2:48] Exportclass signup, opa, signup page, perfeito.

[2:58] A mesma estrutura que usamos lá, por exemplo, navegar para página tal. Pegar o input tal.

[3:08] É o que nós vamos fazer aqui.

[3:09] Então, navegar para signup.

[3:16] Tranquilo?

[3:17] E vamos fazer o que?

[3:19] Return browse.get e passamos /hashtag/signup.

[3:30] Faltou o home aqui, se não, vai dar erro.

[3:32] Está bom.

[3:35] No signin.po.ts a gente já tem o verificar o url, como a gente vai chamar tanto o singnup.po, signin.po e o signup.po lá para o home.e2e, não tem problema, não vai precisar que a gente utilize, não vai precisar que a gente crie um outro arquivo, a gente pode utilizar o de outro page object. - [4:03] Certo?

[4:04] Então, não vou criar navegar no método de verificar url, vou deixar a responsabilidade para o outro page object.

[4:14] E o que precisamos fazer aqui?

[4:15] Simplesmente, capturar esses inputs que vemos nessa tela de home.signup, né?

[4:23] Que é: e-mail, full namo, user name e password.

[4:28] Então, voltando para o código.

[4:32] Na verdade nem é pegar. A gente só vai mandar registro, então, o que a gente pode fazer?

[4:36] Registrar usuário. Usuário.

[4:43] Certo.

[4:45] E aqui vamos fazer o que?

[4:47] Vamos dar um return elemento.

[4:52] Opa.

[4:53] Return, beleza.

[5:00] By.css.

[5:02] Talvez você esteja se perguntando: mas, eu vou ter que colocar em tudo, isso toda hora.

[5:08] Escrever form control name, toda hora será que não tem um jeito mais fácil de fazer isso? Sabe?

[5:12] O que a gente pode fazer?

[5:15] Ao invés de colocar o input e o css, a gente, o protractor deixa disponível para que nós consigamos criar os nossos elementos, os nossos locator ou os nossos localizadores de maneira customizada.

[5:33] Como assim?

[5:34] Em vez de colocar o css e o input, eu posso colocar direto form control name.

[5:43] Certo?

[5:44] Mas, onde a gente vai configurar isso?

[5:46] Vão lá.

[5:50] Eu vou dar um ctrl P, para digitar o protractor .config e dentro do protractor.config tem um método chamado onPrepare.

[5:59] Esse onPrepare a gente consegue adicionar os nossos locators, nossos localizadores customizados.

[6:07] Como?

[6:08] Vou digitar um by aqui, opa, ele já trás um método (LOUCO) aqui para gente, mas, eu quero um by.addlocator .

[6:17] Está bom?

[6:18] Se você vier, até aqui.

[6:20] Vamos navegar pela API dele.

[6:22] Aqui, tem um. Deixa eu ver. Addlocator

[6:30] E Tem esse addlocator informando para gente como é que a gente cria um novo locator customizado.

[6:37] Então, voltando para o código, protractor.config.ts, vou passar dois parâmetros.

[6:44] Primeiro: form control name com o nome que a gente quer criar no nosso locator e o método.

[6:54] Perfeito?

[6:55] Vou dar um ctrl P para nossa área útil em desenvolvimento aumentar, e aqui eu vou começar a digitar.

[7:01] Eu quero que, toda vez que o nosso (DOM), ele chamar form control name, eu quero que ele busque todos os elementos que tenham o form control name.

[7:14] Como?

[7:15] Eu dou `document.querySelectorAll`, e dentro desse `query selector all` eu vou abrir parênteses, ou melhor, eu vou abrir o acento craseado, e vou digitar aqui, abro colchetes `form control name` ele vai ser igual.

[7:44] Cifrão, abro chaves, o valor que passarmos.

[7:46] Perfeito?

[7:48] Vou dar um ponto e virgula aqui no final.

[7:50] E esse valor vai ser passado aqui dentro dessa função.

[7:54] Entendeu?

[7:55] Certinho?

[7:56] O que ele está fazendo?

[7:57] Ele acessa o (BANCO) e ele busca todos os elementos que são `form control name`, igual aquele valor que a gente buscar, no caso.

[8:06] Então, eu vou voltar para o `signup.po`.

[8:10] E vou dizer o que?

[8:10] Mudei o...

[8:13] Eu preciso importar o `by` aqui, `by`, se não vai dar erro mesmo.

[8:15] E esse `form control name` agora posso dizer que é o e-mail que está referenciado na página do `signup`,

[8:25] Mas, vamos conferir só, todos os elemento, todos os `form control name` que ele tem declarado na página de `signup`.

[8:30] Vou dar um `control P`.

[8:32] `Signup`. Opa.

[8:33] `Signup.component.html`.

[8:38] Então, aqui eu tenho e-mail, eu tenho `full name`, e `user name` e `password`.

[8:48] Beleza.

[8:49] E-mail, `fullname`, `user name` e `password`, no caso aqui, podemos fazer da mesma maneira que criamos no outro `page object`, que é o que?

[9:01] `Form control name`, que eu faço para ir duas variáveis, dois parâmetros, `form control name`.

[9:07] E, um valor, que esse valor eu vou enviar, depois que ele identificar onde está o nosso `form control name`.

[9:16] Certo?

[9:17] `Form control name`.

[9:24] Aqui, então, como é que ficou?

[9:25] Return, elemento, abro parênteses by form control name a variável form control name, que eu estou passando aqui, aqui você pode colocar qualquer nome, eu só deixei form control name para ficar mais intuitivo, aí, .sendKeys valor.

[9:42] Entendido?

[9:42] Então, salvei esse camarada aqui.

[9:48] Para terminar essa parte do page object, eu vou fazer o que?

[9:51] só vou fazer chamada daquele botão que tem no signup.component.html.

[9:58] Está certo?

[9:59] Então, gente, vamos só voltar para o html, vamos verificar aqui, tem um botão e esse botão tem o button (TEXT) com o texto de Register.

[10:08] Então, vamos fazer o que?

[10:09] Voltar parra o signup.po.ts, então, eu tenho que pegar botão register.

[10:20] Perfeito?

[10:21] E aqui, return elemento by button text register.

[10:35] Tá bom?

[10:36] Então, aqui, essa parte do page object, signup.page object.po está tudo certo.