

Como criar seu primeiro teste automatizado com Selenium WebDriver?

Bom, falando aqui de iniciante para iniciante eu vou tentar diminuir os gargalos de aprendizagem que tive durante o estudo do Selenium Webdriver em Java. Mas antes disso, vamos esclarecer algumas coisas aqui.

O que eu consigo automatizar com o Selenium Webdriver?

Então, em geral Testes de Interface do usuário, ou seja, você consegue simular a manipulação da página como se fosse um usuário normal.

Tá, mas para que eu vou querer automatizar isso se eu já faço manualmente?

Então, automatizando os principais fluxos de execução da sua aplicação, você traz uma certa tranquilidade para a sua equipe, visto que, toda vez que tiver um build do projeto, a automação vai rodar e garantir que aquilo que já funcionava antes continua funcionando e isso dá mais segurança a toda equipe. Além disso, vale lembrar que, nós como Homo Sapiens, estamos sujeitos a temperamentos, as vezes você pode não estar 100% no dia, ou simplesmente se sente tão confortável com o sistema sempre indo tudo certinho que

você deixa de testar uma parte importante, bom, é aí que entra o problema, isso pode gerar certos prejuízos a equipe.

O que é recomendado que eu automatize com o Selenium Webdriver?

O que o usuário deveria fazer no seu sistema? Criar um questionário? Fazer uma compra? Enviar um arquivo? Enfim, depende muito de sistema para sistema, mas eu vou considerar aqui o seguinte: Abrir a página, realizar o login(ou tentar), clicar em um botão, preencher um campo e capturar um texto que são os comandos mais utilizados e a automação desse tipo de teste gira basicamente nisso, cabendo a você somente saber organizar.

Quais conteúdos eu já devo ter uma noção para poder compreender e desenvolver com facilidade a automação?

- Conhecimento básico na linguagem Java
- Conhecimento básico em HTML e CSS
- Ter a JDK do Java já instalada
- Um pouquinho de inglês também ajuda, mas nada fora do normal

Preparando o ambiente

No seu Google Chrome vá na opção Ajuda e Sobre o Google Chrome, lá vai informar a versão do seu Chrome, o meu por exemplo está na versão 84.0.4147

Sabendo isso, vá agora no site do chromedriver <https://chromedriver.chromium.org/downloads> e realize o download da versão do chromedriver igual à do seu. Irá vir um arquivo .rar, você extrai e terá um executável.

Para facilitar você vai criar uma pasta no seu diretório C: com o nome que você preferir, eu aqui irei chamar de “projeto”, dentro dessa pasta você coloca o executável do chromedriver.

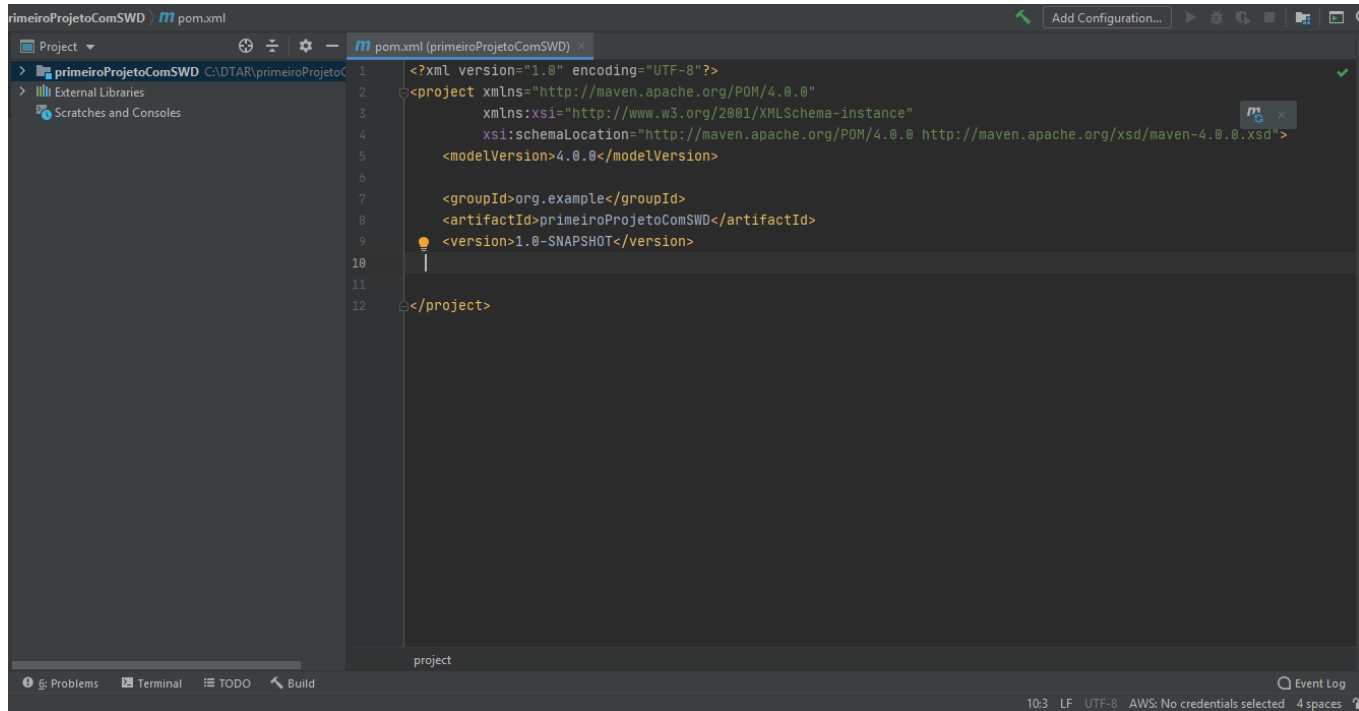
Criando um projeto na sua IDE de escolha

File > New > Project

Na tela que abrir você vai em Maven, lá em Project SDK precisa já estar selecionado a SDK, se não tiver você vai lá e seleciona. Caso não apareça você vai precisar instalar. Aperta em Next, em Name você põe o nome do seu projeto, em Location você localiza a pasta onde você quer que o projeto seja criado, aqui eu recomendo em C:\...Nome do seu projeto e aperte Finish.

Feito isso seu primeiro projeto será criado, provavelmente você já irá cair na tela do “pom.xml”, esse arquivo ajuda

você a configurar e importar o que você vai precisar nos testes.



Dentro do pom.xml você precisará incluir as dependências do Junit e selenium-java que se encontram no link abaixo, é só procurar pelo nome da extensão que quer usar Ex.JUnit ou Selenium.

<https://mvnrepository.com/>

Obs. Versões alpha e beta são versões mais instáveis então procure sempre versões mais estáveis e que tenham mais pessoas usando como no exemplo abaixo:

Category	Version	Repository	Artifacts	Date
Cache Implementations	4.0.0-rc-1	Central	10	Sep, 2021
Cloud Computing	4.0.0-beta-4	Central	10	Jun, 2021
Code Analyzers	4.0.0-beta-3	Central	9	Apr, 2021
Collections	4.0.0-beta-2	Central	3	Mar, 2021
Configuration Libraries	4.0.0-beta-1	Central	4	Feb, 2021
Core Utilities	4.0.0-alpha-7	Central	4	Nov, 2020
Date and Time Utilities	4.0.0-alpha-6	Central	7	May, 2020
Dependency Injection	4.0.0-alpha-5	Central	5	Mar, 2020
Embedded SQL Databases	4.0.0-alpha-4	Central	7	Jan, 2020
HTML Parsers	4.0.0-alpha-3	Central	12	Sep, 2019
I/O Utilities	4.0.0-alpha-2	Central	7	Jul, 2019
JDBC Extensions	4.0.0-alpha-1	Central	8	Apr, 2019
JDBC Pools	3.141.59	Central	391	Nov, 2018
JPA Implementations	3.141.x	Central	13	Nov, 2018
JSON Libraries	3.141.5	Central	9	Oct, 2018
JVM Languages	3.141.0	Central	9	Oct, 2018
Logging Frameworks	3.14.x	Central	100	Aug, 2018
Logging Bridges	3.14.0	Central	41	Jun, 2018
Mail Clients	3.13.x	Central	41	Jun, 2018
Maven Plugins	3.13.0	Central	47	May, 2018
Mocking	3.12.x	Central	47	May, 2018
Object/Relational Mapping	3.12.0	Central	64	Mar, 2018
	3.11.x	Central	64	Mar, 2018
	3.11.0	Central	64	Mar, 2018

Selenium Java » 3.141.59

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

License: Apache 2.0

Categories: Web Testing

HomePage: <http://www.seleniumhq.org/>

Date: (Nov 14, 2018)

Files: [pom \(3 KB\)](#) [jar \(355 bytes\)](#) [View All](#)

Repositories: Central

Used By: 1,405 artifacts

Note: There is a new version for this artifact

New Version: 4.0.0-rc-2

Maven: [Gradle](#) [Gradle \(Short\)](#) [Gradle \(Kotlin\)](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>
```

[Copiar daqui e colar no pom.xml](#)

☒ Include comment with link to declaration

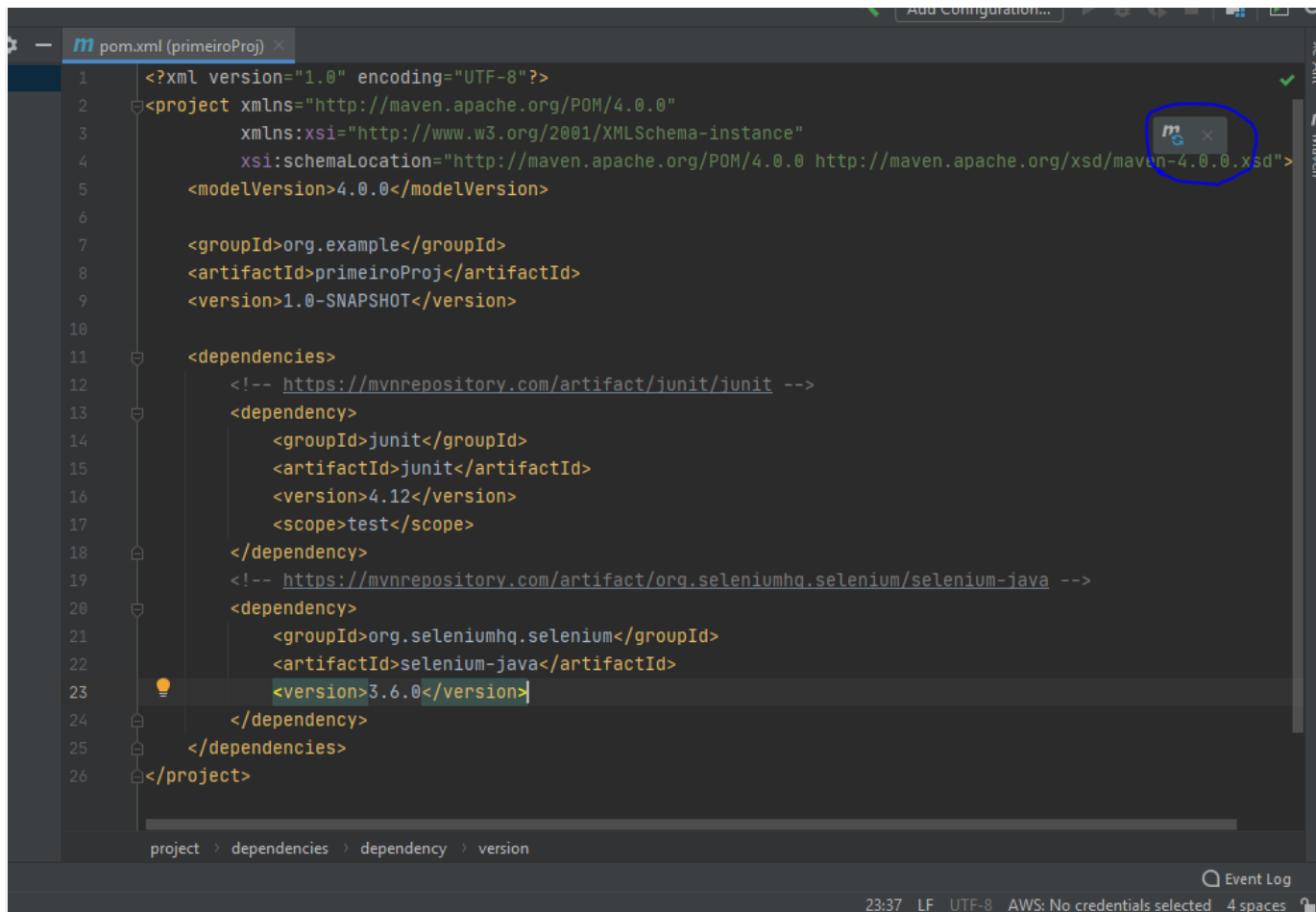


Para facilitar, irei pôr o código aqui já, mas caso tenha atualização de versões é bom vocês buscarem no link acima citado as versões mais atualizadas.

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/junit/junit -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
  </dependency>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.6.0</version>
  </dependency>
</dependencies>
```

Feito isso vai aparecer na tela um botão do maven que é o Load Maven Changes, você precisa rodar esse load para ele carregar as dependências.

Ou caso esteja no Eclipse deverá ir em seu projeto mandar executar um maven install clicando com o botão direito sobre o mesmo.



Carregando as alterações do Maven

Pronto, o ambiente já está configurado.

Abrindo o navegador pela primeira vez

Para criar a primeira classe de testes você vai em “src > test > java” crie um package chamado testes e dentro dele uma nova classe. ATENÇÃO, ESSA NOVA CLASSE DEVE TERMINAR COM O TEXTO “Test”, exemplo “primeiroTesteTest”. Isso vale também para qualquer outra classe que você vá rodar os testes, por exemplo “DashboardTest”, “LoginTest” etc. Você também deve por @Test acima do método de teste que você criou.

Agora dentro da classe você vai criar o método de teste que aqui eu nomeei como entrar no Facebook, mas você pode criar como quiser. Vou pôr o código aqui comentado explicando cada linha. Sugiro que você remova os comentários!

```
@Test
public void entrarNoFacebook(){
    //Aqui você informa onde o chromedriver está na sua máquina,
    // caso você não tenha posto o chrome driver na pasta que eu indiquei,
    // será necessário setar a pasta no segundo parâmetro abaixo.
    System.setProperty("webdriver.chrome.driver", "C:\\projeto\\chromedriver.exe");
    //Aqui você está atribuindo ao navegador para abrir um novo navegador
    WebDriver navegador = new ChromeDriver();
    //Aqui você está informando ao SWD que o tempo máximo de resposta que deve esperar
    // para tentar realizar o teste é de 15 segundos, isso ajuda a quebrar menos
    // o código, porque em alguns casos pode ser que a página ou o elemento demore de aparecer.
    navegador.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
    //Aqui você indica o endereço web que você quer abrir.
    navegador.get("https://facebook.com");
}
```

Abrindo o navegador

Caso você aperte no Run, aquele botão verde que fica ao lado do código ele já irá executar a o teste abrindo o navegador.

Se você receber esse erro quando tente executar o teste siga os seguintes passos:

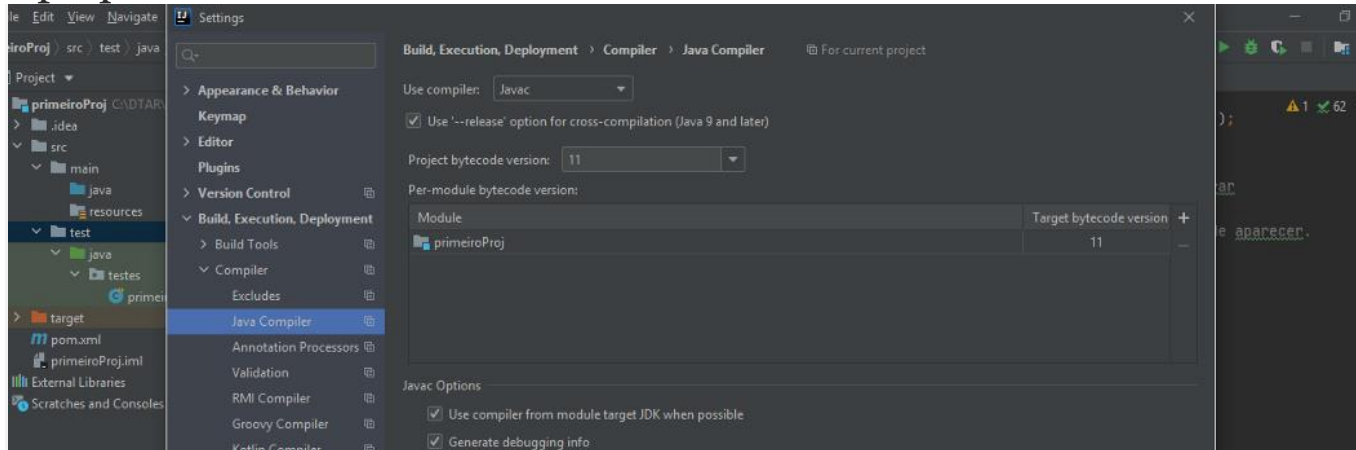
```
ms java: error: release version 5 not supported
```

Erro da versão

File > Setting > Build, Execution, Deployment > Compiler > Java Compiler

Em “Project bytecode version” selecione o “11”. Mais abaixo em “Target bytecode version” selecione “11” também.

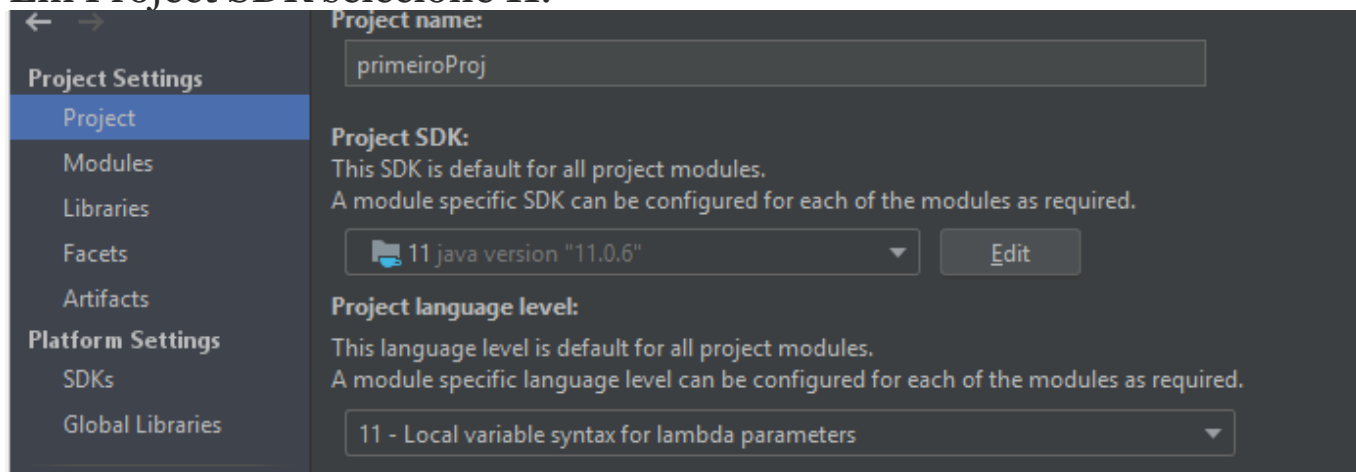
Aplique e salve.



Corrigindo o erro 1

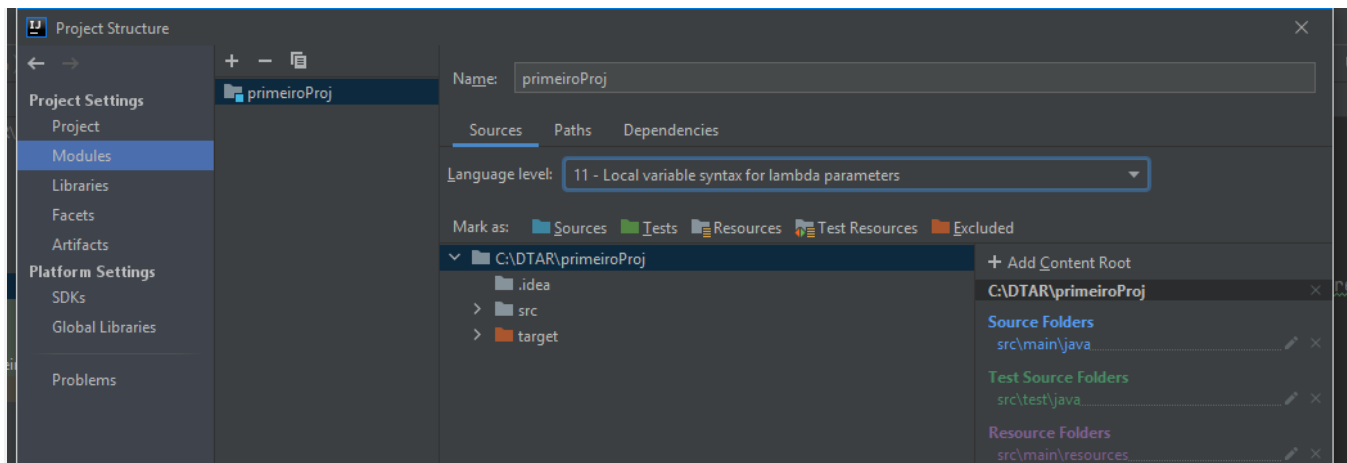
Agora em File > Project Structure > Project

Em Project SDK selecione 11.



Corrigindo o erro 2

Saindo de Project passe para a aba baixo “Modules” em Language level selecione o 11.



Corrigindo o erro 3

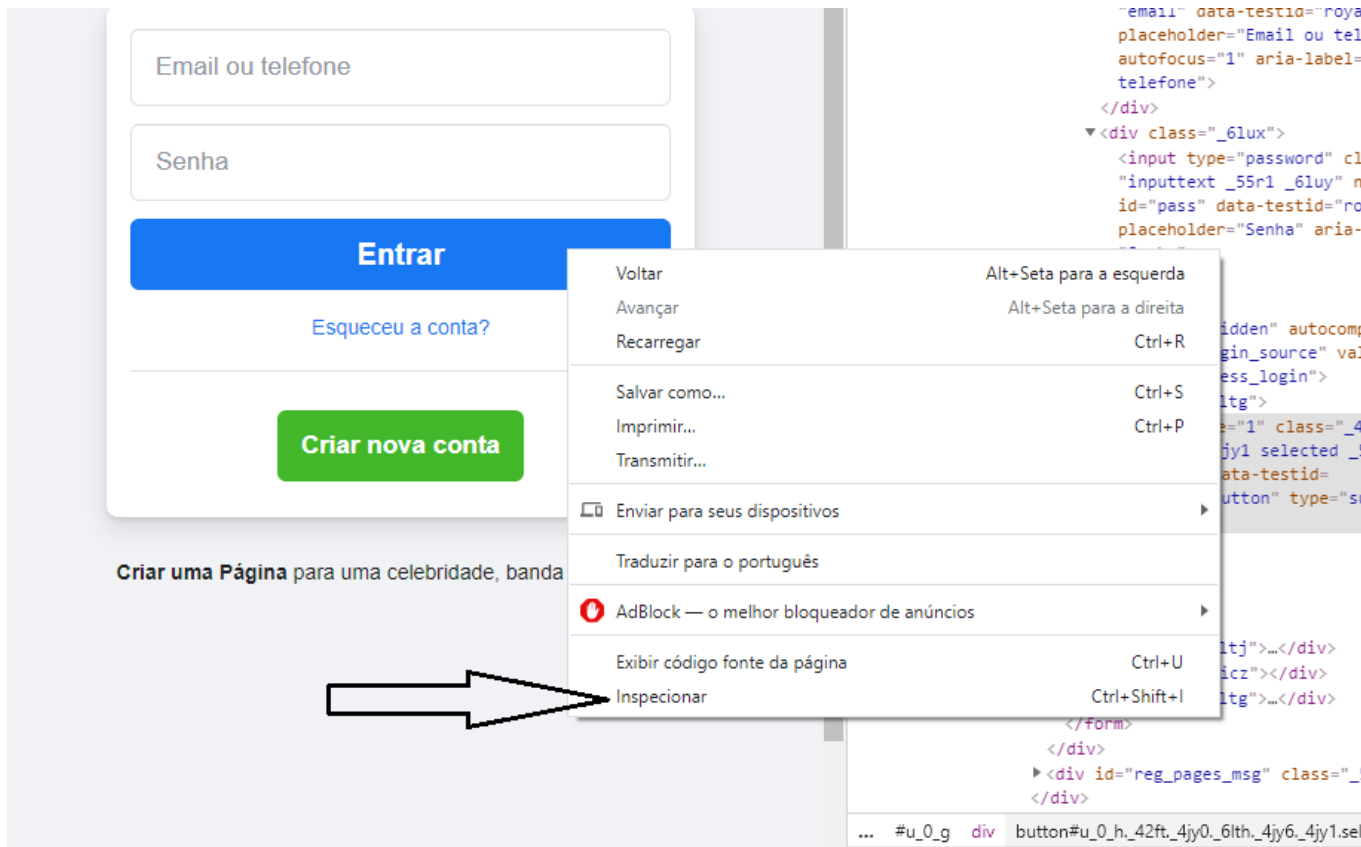
Pronto, é só rodar novamente no Run que agora irá executar.

Localizando elementos

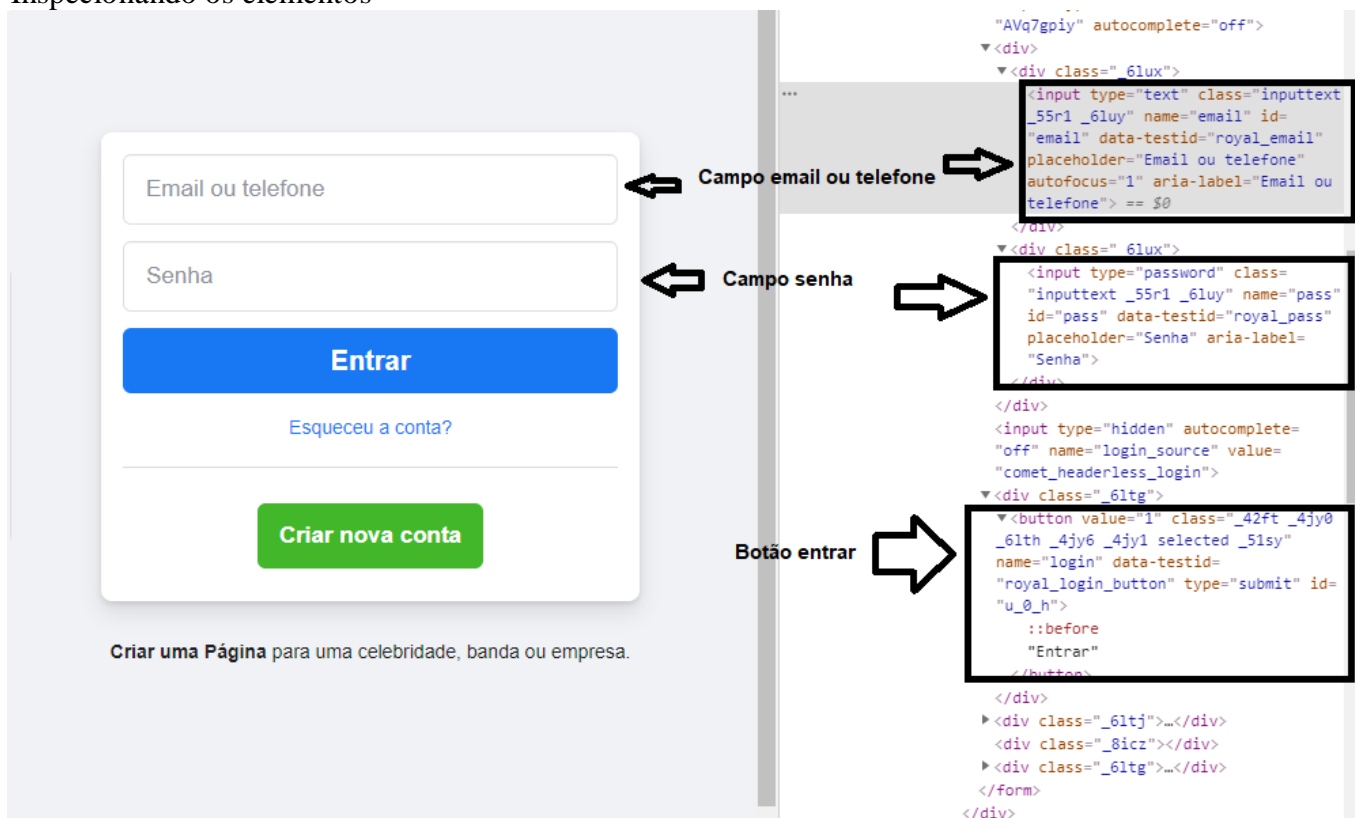
Existem 8 formas de encontrar elementos no Selenium WebDriver. Pode ser pelo nome da classe, css selector, id, name, link text, tag name e xpath. Aqui eu vou utilizar pelo id, xpath, link text e nome da classe. Caso queira saber mais sobre a localização de elementos

acesse https://www.selenium.dev/documentation/legacy/selenium_id/#locating-elements

Para achar os elementos na sua página você deve clicar com o botão direito em cima do elemento que deseja capturar e clicar em Inspeccionar. Observe as imagens:



Inspeccionando os elementos



Inspeccionando os elementos

Aqui estão alguns dos principais métodos que você irá utilizar para localizar os elementos e realizar ações durante a escrita dos scripts:

`findElement()` — Você indica pelo que deve ser buscado, dentro dos () você informa os parâmetros que podem ser algum dos 8 que informei acima.

`click()` — Você informa para ser realizada ação de click

`sendKeys()` — Você informa que deve ser enviado um texto

`getText()` — Você informa que deve ser capturado o texto

`assertEquals()` — Você faz uma comparação de igualdade entre os parâmetros informados.

Escrevendo os primeiros comandos

```
//Encontrando o campo email/telefone pelo id e preenchendo
navegador.findElement(By.id("email")).sendKeys( ...charSequences: "marcinho_boladão@test.com");
//Pelo name
navegador.findElement(By.name("email")).sendKeys( ...charSequences: "marcinho_boladão@test.com");
//Pelo xpath
navegador.findElement(By.xpath("//input[@type='text']")).sendKeys( ...charSequences: "marcinho_boladão@test.com");

//Encontrando o campo senha e preenchendo pelo id
navegador.findElement(By.id("pass")).sendKeys( ...charSequences: "123456");
//Pelo name
navegador.findElement(By.name("pass")).sendKeys( ...charSequences: "123456");
//Pelo xpath
navegador.findElement(By.xpath("//input[@type='password']")).sendKeys( ...charSequences: "123456");
```

Preenchendo campo login e senha

No seu caso você só irá utilizar uma forma de encontrar o elemento, eu coloquei três para ficar um pouco mais claro.

Não é em todos os casos que o elemento vai ter um Id ou um Name para encontrar, nesses casos você pode utilizar o Xpath que parece ser um pouco mais difícil, mas ajuda bastante nessas situações. Como se pode observar dentro dos parâmetros você indica ali o Id, Name ou o caminho do Xpath. Dentro do sendKeys() você indica o texto que você quer inserir ou, você pode por uma variável que já contenha o texto, aí você é quem decide.

Agora vamos encontrar o botão Entrar

```
//Encontrando o botão Entrar pelo name
navegador.findElement(By.name("login")).click();
//Pelo id
navegador.findElement(By.id("u_0_h")).click();
//Pelo xpath
navegador.findElement(By.xpath("//button[@type='submit']")).click();
//Pelo texto que ele possui
navegador.findElement(By.linkText("Entrar")).click();
```

Clicando no botão entrar

Como se pode observar neste caso escrevi 4 formas diferentes de encontrar o botão Entrar. Escolha a que melhor se aplica ao seu projeto.

Para finalizar vamos utilizar o getText() e o assertEquals(). Aqui eu vou comparar um texto com a captura do erro quando você não insere nenhuma informação para login.

```
navegador.findElement(By.linkText("Entrar")).click();
navegador.findElement(By.name("login")).click();

String erro = navegador.findElement(By.xpath("//div[@role='alert']")).getText();
assertEquals( expected: "O email ou o número de telefone inserido não " +
    "corresponde a nenhuma conta. Cadastre-se para abrir uma conta.", erro);
```

Capturando erro de campos obrigatórios

Repare que, depois de tentar entrar com um login e senha errado, eu informei para clicar novamente em entrar, porque nesse caso a página de login e senha estarão em branco na segunda tentativa, assim surgir o erro.

Extra — Executando o teste sem abrir o Navegador

Algo que será muito utilizado mais a frente é executar o código sem abrir o navegador, isso será necessário para implementar o código em uma possível integração contínua caso queira. É muito simples, mas deu um trabalho encontrar, então vamos lá.

```
ChromeOptions options = new ChromeOptions();
options.setHeadless(true);
System.setProperty("webdriver.chrome.driver", "C:\\projeto\\chromedriver.exe");
WebDriver navegador = new ChromeDriver();
navegador.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
navegador.get("https://facebook.com");
```

Executando o projeto em Headless

Pronto, inserindo essas duas primeiras linhas de código, o navegador não irá mais abrir, sendo assim, o teste será

executado em modo Headless. Caso você queira voltar para o navegador aparecer, mude de true para false.

Para aprender mais sobre o Selenium WebDriver recomendo que vocês acessem a documentação e também tem um curso ótimo do Júlio de Lima, que além de aprender a criar os scripts de automação ele ensina como organizar seu código com boas práticas, afinal, mais importante que escrever o código, é escrevê-lo bem!

Link da documentação.

<https://www.selenium.dev/documentation/>

Curso de Júlio de lima:

<https://www.youtube.com/watch?v=IELENTQ-PWc>