05

Primeiro teste com Selenium

Transcrição

Agora que já conhecemos como o projeto funciona, é hora de finalmente escrevermos um teste com o Selenium. Com o projeto aberto, criaremos nossa primeira classe de teste, um "Hello world!", para testarmos as funcionalidades da ferramenta.

Atualmente o projeto não possui a pasta "src/test/java", padrão do Maven. Sendo assim, criaremos com o botão direito no projeto e então em "New > Folder". Na janela que se abrirá, preencheremos o campo "Folder name" com "src/test/java" e clicaremos em "Finish". Você só precisará fazer isso caso seu projeto também não tenha esse diretório.

Clicaremos com o botão direito na nova pasta e então em "New > Other". Na nova janela, selecionaremos "Class" e clicaremos em "Next". Atribuiremos à classe o nome `HelloWorldSelenium` e substituiremos o pacote (*package*) para o mesmo da aplicação, `br.com.alura.leilao` , e clicaremos em "Finish".

Na classe `HelloWorldSelenium` , escrevemos um método `hello()` que por enquanto ficará vazio. Acima dele, adicionaremos a anotação `@Test` para que o JUnit o reconheça como um método de teste, e faremos a importação do pacote `org.junit.jupiter.api.Test` (algo que, no Eclipse, pode ser facilitado com o atalho "Ctrl + Shift + O".

```
package br.com.alura.leilao;

import org.junit.jupiter.api.Test;

public class HelloWorldSelenium {

    @Test
    public void hello() {

    }
}
```

COPIAR CÓDIGO

Antes de escrevermos nosso teste, precisamos adicionar o Selenium ao projeto. Como estamos utilizando o Maven, podemos baixar e instalar o Selenium na aplicação simplesmente adicionando-o como uma dependência. Para isso, abriremos o arquivo `pom.xml`, buscaremos a área de dependências e adicionaremos, após a dependência `com.h2database`, a do Selenium. Isso requer que saibamos o `groupId` e o `artifactId` desta dependência.

O site do Selenium (https://www.selenium.dev/documentation/en/webdriver/)possui, além da documentação e explicações sobre o projeto, um menu "Downloads" onde encontraremos a seção "Selenium Client & Webdriver". Nela estão descritas as versões para cada linguagem de programação, no nosso caso o Java. Entretanto, se baixarmos o Selenium pelo link disponível na página, receberemos um ZIP com os arquivos `.jar` que devem ser adicionados manualmente ao projeto. Como estamos utilizando o Maven, essa não é a opção que desejamos.

Ao invés disso, buscaremos na página pela seção "Maven infomation". Clicando no link, teremos disponível a tag com a dependência. Entretanto, a dependência padrão baixa os drivers de todos os navegadores - Chrome, Firefox, Safari, IE e

assim por diante. Para simplificarmos o conteúdo do curso, utilizaremos somente o Google Chrome. Sendo assim, mais abaixo na página, copiaremos a versão da tag para apenas um único driver.

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-firefox-driver</artifactId>
  <version>3.X</version>
</dependency>
```

COPIAR CÓDIGO

Note que esta versão está adaptada ao Firefox, mas podemos simplesmente substituir, no arquivo `pom.xml`, a palavra `firefox` por `chrome` de modo a conseguirmos o driver correto. Como estamos utilizando o Springboot no projeto, que dá conta de fazer o download da versão correta das dependências, podemos remover essa informação da tag.

```
//...código omitido

        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
        </dependency>

        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-chrome-driver</artifactId>
        </dependency>

    </dependencies>
//...código omitido
```

COPIAR CÓDIGO

Após salvarmos o arquivo, voltaremos para nossa classe de teste. Agora que já temos a API disponível, podemos escrever nosso primeiro teste automatizado utilizando o Selenium. A ideia desse teste é abrirmos o navegador e entrarmos na página principal do nosso projeto, http://localhost:8080/leiloes (http://localhost:8080/leiloes).

Para abrir o navegador, existe uma interface principal do Selenium chamada WebDriver. Criaremos uma variável `browser` do tipo `WebDriver` e importaremos esta interface do pacote `org.openqa.selenium.WebDriver`. Sendo uma interface, no momento da instanciação precisamos passar qual será a implementação. Já que queremos utilizar o Chrome, instanciaremos um novo `ChromeDriver()` e o importaremos. Se quiséssemos utilizar o Firefox, teríamos `new FirefoxDriver`, e assim por diante para cada navegador específico.

```
import org.junit.jupiter.api.Test;
import org.openqa.selenium.chrome.ChromeDriver;

public class HelloWorldSelenium {

    @Test
    public void hello() {

        WebDriver browser = new ChromeDriver();
    }
}
```

COPIAR CÓDIGO

O próximo passo será abrirmos o endereço da aplicação. A partir da variável `browser`, chamaremos o método `navigate().to()` passando uma string com o endereço da página que queremos acessar. Por fim, para não mantermos uma janela do navegador aberta, vamos fechá-la como método `browser.quit()`.

```
public class HelloWorldSelenium {

    @Test
    public void hello() {

        WebDriver browser = new ChromeDriver();
        browser.navigate().to("http://localhost:8080/leiloes");
        browser.quit();
    }
}
```

COPIAR CÓDIGO

Escrevemos nosso "hello world" com apenas três linhas: a primeira abre uma janela do Google Chrome, a segunda navega para o endereço da aplicação e a terceira fecha a janela que foi aberta. Entretanto, ao executarmos o teste (clicando com o botão direito e então em "Run as > JUnit Test"), receberemos um erro indicando que o executável do driver não foi encontrado.

Isso acontece porque o Selenium, além de ter uma API que disponibiliza classes para escrevermos nosso teste automatizado, também precisa de um driver - um arquivo que fará a ponte com o navegador que está instalado no computador. Sendo assim, além de baixarmos a dependência do Selenium por meio do `pom.xml`, também precisamos baixar o driver do Google Chrome.

Pesquisando por "chrome webdriver" no buscador, encontrarem o site do projeto Chromium (https://chromedriver.chromium.org/downloads), no qual poderemos baixar o driver desejado. À época da gravação desse curso, a última versão disponível era a **87**, que é justamente a versão do Chrome que tenho instalada em meu computador. É ideal que você baixe a versão respectiva ao navegador que você tem instalado em sua máquina, de preferência atualizando o seu Chrome para ter disponível a versão mais recente.

Clicando no link de download, seremos levados a uma página com as versões do driver para cada sistema operacional, seja Linux, Windows ou Mac. Baixaremos então o arquivo ZIP cujo conteúdo deverá ser extraído para o computador. Em seguida, criaremos no projeto um diretório "drivers" dentro da qual manteremos o driver que acabamos de baixar. Todos os drivers que quisermos utilizar posteriormente serão salvos nessa pasta.

Agora precisamos apontar para o Selenium que o driver do Chrome está no diretório que criamos. Na documentação disponível no site do Selenium encontramos a seção "Driver requirements" (requisitos de driver). Na parte específica do Java, ele indica a linha que informa ao Selenium o caminho do executável.

```
System.setProperty("webdriver.chrome.driver", "/path/to/chromedr
```

COPIAR CÓDIGO

Adicionaremos a linha ao nosso método de teste, substituindo o caminho pelo diretório do nosso webdriver ("/drivers/chromedriver", já que o criamos na raiz do projeto).

```
public class HelloWorldSelenium {

    @Test
    public void hello() {
        System.setProperty("webdriver.chrome.driver", "/drivers/
        WebDriver browser = new ChromeDriver();
        browser.navigate().to("http://localhost:8080/leiloes");
        browser.quit();
    }
}
```

COPIAR CÓDIGO

Ao executarmos, teremos um novo erro indicando que o driver encontrado não é executável. Isso acontece no Linux e no MacOS, pois o arquivo baixado não está com permissão de escrita. Para resolvermos, abriremos o Terminal/Prompt de comando, acessaremos o diretório em que está o driver e substituiremos a sua permissão de execução (no Linux) com `chmod +x chromedriver`. Após executarmos, voltaremos ao teste e tentaremos executá-lo novamente.

Nosso navegador será aberto, mas não conseguiremos entrar no endereço - nesse caso, somente porque nosso projeto não está sendo executado. Resolveremos esse problema acessando a classe `LeilaoApplicationJava` e executando-a com botão direito seguido de "Run as > Java Application".

Ao rodarmos o teste, uma janela do navegador será aberta, acessará o endereço da aplicação e será fechada em seguida. Na aba do JUnit veremos que 1 teste foi executado e que ele passou com sucesso.

Esse foi o nosso "hello world" com Selenium. Perceba que foi um processo um pouco complicado, já que tivemos que fazer todo o passo a passo de baixar as dependências, baixar o driver, alterar suas permissões de execução, setar o caminho do driver e escrever o teste em si com os passos específicos. Não se preocupe, pois essas configurações só precisam ser feitas uma vez, e daqui para frente somente nos preocuparemos em escrever os testes.

Na próxima aula continuaremos explorando a API do Selenium.