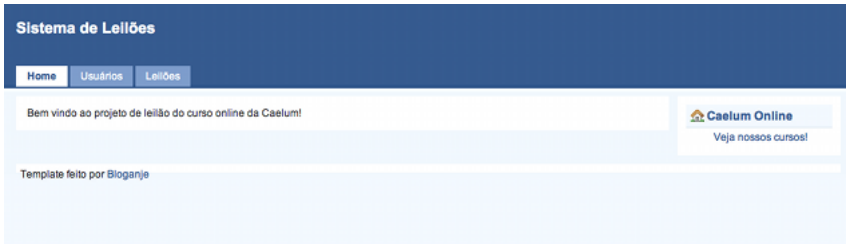
01

Testes manuais ou automatizados?

Transcrição

Estamos muito acostumados a testar nossas aplicações de maneira manual. Empresas geralmente possuem imensos roteiros de script, e fazem com que seus analistas de teste executem esses scripts incansavelmente. Mas quais os problemas com essa abordagem?

Ao longo deste curso, testaremos uma aplicação que administra uma empresa de leilões. Nela, o usuário da empresa pode adicionar novos usuários, cadastrar leilões e efetuar lances nos mesmos. Essa aplicação foi desenvolvida em Java e está pronta para ser executada. A imagem abaixo mostra a tela inicial do sistema:



A aplicação não é muito grande: possui cadastro de usuários, leilões e lances. Mas, apesar da pequena quantidade de funcionalidades, pense na quantidade de cenários que você precisa testar para garantir seu funcionamento:

- Cadastrar um usuário com sucesso
- Editar um usuário
- Validar o cadastro de usuário com nome ou e-mail inválido
- Excluir um usuário
- Exibir ficha completa de um usuário
- ...

Se pensarmos em todos os cenários que devemos testar, percebemos que teremos uma quantidade enorme de cenários! Quanto tempo uma pessoa leva para executar todos esses cenários? Agora imagine o mesmo problema em uma aplicação grande. Testar aplicações grandes de maneira manual leva muito tempo! E por consequência, custa muito caro!

Na prática o que acontece é que, como testar sai caro, as empresas optam por não testar! No fim, entregamos software com defeito para nosso cliente! Precisamos mudar isso!

Se removermos a parte humana do processo e fizermos com que a máquina execute o teste, resolvemos o problema: a máquina vai executar o teste rápido, repetidas vezes, e de graça!

A grande questão é: como ensinar a máquina a fazer o trabalho de um ser humano? Como fazê-la abrir o browser, digitar valores nos campos, preencher formulários, clicar em links e etc?

Para isso, faremos uso do Selenium! O Selenium é um framework que facilita e muito a vida do desenvolvedor que quer escrever um teste automatizado! Para usá-lo, faça o download do Selenium Server Standalone aqui (https://github.com/seleniumhq/selenium).

Basta agora criar um projeto java qualquer (aqui vamos chamá-lo de "testes-de-sistema") e referenciar esse .jar (copie o .jar pra dentro do projeto, clique com o botão direito sobre ele e selecione `Build Path -> Add Library` !

A primeira coisa que uma pessoa faria para testar a aplicação seria abrir o browser. Com Selenium, precisamos apenas da linha abaixo:

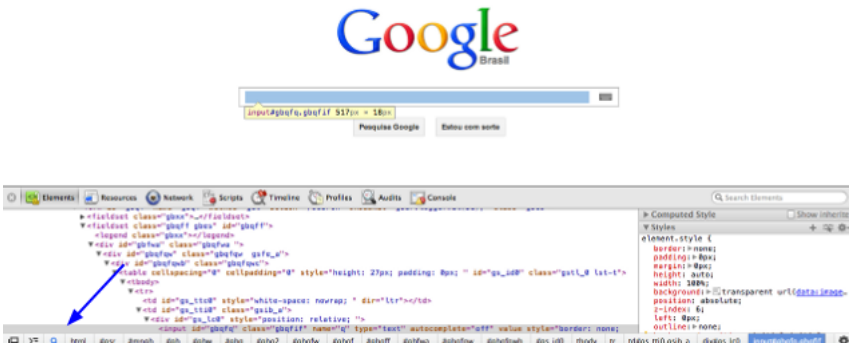
```
WebDriver driver = new FirefoxDriver();
```

Nesse caso, estamos abrindo o Firefox! Em seguida, entraríamos em algum site. Vamos entrar no Google, por exemplo, usando o método `get()` :

```
driver.get("http://www.google.com.br/");
```

Para buscar o termo "Caelum", precisamos digitar no campo de texto. No caso do Google, o nome do campo é "q".

Para descobrir, podemos fazer uso do Inspector, no Chrome (ou do Firebug no Firefox). Basta apertar Ctrl+Shift+I (ou F12), e a janela abrirá. Nela, selecionamos a lupa e clicamos no campo que queremos descobrir o nome. Ele nos levará para o HTML, onde podemos ver `name="q"` .



Com Selenium, basta dizermos o nome do campo de texto, e enviarmos o texto:

```
WebElement campoDeTexto = driver.findElement(By.name("q"));
campoDeTexto.sendKeys("Caelum");
```

Agora, basta submetermos o `form` ! Podemos fazer isso pelo próprio `campoDeTexto` :

```
campoDeTexto.submit();
```

Pronto! Juntando todo esse código em uma classe Java simples, temos:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class TesteAutomatizado {

    public static void main(String[] args) {
        // abre firefox
        WebDriver driver = new FirefoxDriver();

        // acessa o site do google
        driver.get("http://www.google.com.br/");

        // digita no campo com nome "q" do google
        WebElement campoDeTexto = driver.findElement(By.name("q"));
        campoDeTexto.sendKeys("Caelum");

        // submete o form
        campoDeTexto.submit();

    }
}
```

Se você não entendeu algum método invocado, não se preocupe. Estudaremos eles com mais detalhes nos próximos capítulos. Nesse momento, rode a classe! Acabamos de fazer uma busca no Google de maneira automatizada!

Execute o teste novamente. Veja agora como é fácil, rápido e barato! Qual a vantagem? Podemos executá-los a tempo! Ou seja, a cada mudança que fazemos em nosso software, podemos testá-lo por completo, clicando apenas em um botão. Saberemos em poucos minutos se nossa aplicação continua funcionando!

Quantas vezes não entregamos software sem tê-lo testado por completo? Vamos acabar com isso!

Observação: Caso esteja usando a versão nova do Selenium, será necessário configurar o *GeckoDriver* antes de executá-lo: `System.setProperty("webdriver.gecko.driver", "ENDEREÇO DO EXECUTÁVEL DO GECKODRIVER");` O endereço do executável será algo como `C:\GeckoDriver\geckodriver.exe` . Caso não o tenha instalado na sua máquina, você pode baixa-lo aqui: https://github.com/mozilla/geckodriver/releases (https://github.com/mozilla/geckodriver/releases)