Como iniciar a criação de Testes Funcionais

Pedro de Alcântara dos Santos Neto (pasn@ufpi.edu.br)

1º. Passo: Instalar o Selenium IDE

Para instalar o Selenium IDE, você deve acessar a página http://selenium-ide.openqa.org/download.jsp. Nessa página existe o link para instalar o selenium-ide 0.8.7. Você deve executar o arquivo pelo firefox, que o instala automaticamente como um novo plugin.

Após instalar o Selenium, deve aparecer no menu Ferramentas, do Firefox, o item Selenium IDE. Acessando esse item será aberta uma janela do Selenium, podendo assim iniciar a captura de ações executadas no seu navegador.

2º. Passo: Gravar um script com os passos que você deseja automatizar na forma de um teste.

Para criar um teste na forma de um programa Java, você deve inicialmente capturar as ações relacionadas ao uso do trecho do sistema que você quer testar, para depois exportar para o formato Java. Por exemplo, para criar um Procedimento de Teste para o login no quantum, devo gravar as ações de login nesse sistema. A Figura 1 exibe um exemplo da captura de ações relacionadas ao login no Quantum. A linguagem utilizada pelo Selenium é simples e intuitiva. Ao clicar no campo “comando” podemos ver a lista de comandos disponíveis.

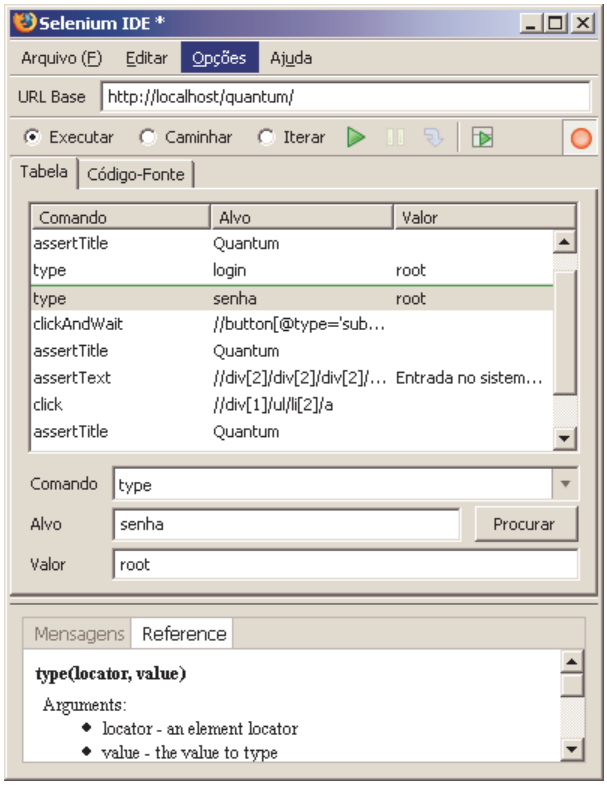

Figura 1: Tela do Quantum com as ações do Login.

3º. Passo: Exportar o teste para Java

Após criar o teste no Selenium, é necessário exporta-lo para Java. Com isso teremos as

```
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class loginExemplo extends SeleneseTestCase {
	public void testLoginExemplo() throws Exception {
		selenium.open("http://localhost/quantum/getRoles.do");
		assertEquals("Quantum", selenium.getTitle());
		selenium.type("login", "root");
		selenium.type("senha", "root");
		selenium.click("//button[@type='submit']");
		selenium.waitForPageToLoad("30000");
		assertEquals("Quantum", selenium.getTitle());
		assertEquals("Entrada no sistema efetuada com sucesso.",
		    selenium.getText("//div[2]/div[2]/div[2]/span"));
		selenium.click("//div[1]/ul/li[2]/a");
		assertEquals("Quantum", selenium.getTitle());
		assertEquals("Saída do sistema efetuada com sucesso.",
		    selenium.getText("//div[2]/div[2]/div[2]/span"));
	}
}
```

Figura 2: Teste exportado para o formato Java.

mesmas ações capturadas anteriormente, porém utilizando uma API Java para execução de comandos.

O teste exibido na Figura 1, exportado para Java, ficará conforme exibido na Figura 2.

```
  public void login(String login, String senha, String role,
String nomeUsuario, String erro) throws Exception
  {
    selenium.open("http://localhost/quantum/ ");
    assertEquals("Quantum", selenium.getTitle());
    selenium.type("login", login);
    selenium.type("senha", senha);
    selenium.click("//button[@type='submit']");
    selenium.waitForPageToLoad("30000");
    assertEquals("Quantum", selenium.getTitle());

    if (erro.length() > 0)
    {
      //Verifica a mensagem de erro
      assertEquals(erro,
       selenium.getText("//div[2]/div[2]/div[2]/span"));
      return;
    }

    //Espera até que um dos resultados da página apareçam,
para
    //que sejam feitas as demais verificações.
    TesteUtil.esperaPorResultado(selenium, "Entrada no
sistema efetuada com sucesso.",
"//div[2]/div[2]/div[2]/span");

    assertEquals(nomeUsuario, selenium.getText("//strong"));
    assertEquals("Sair", selenium.getText("link=Sair"));
    assertEquals("Principal",
selenium.getText("//a[contains(text(),'Principal')]"));

    if (role.equals(RolesEnum.ROOT.getValor()))
    {
      verificaRoot();
    } else if (role.equals(RolesEnum.COORDENADOR.getValor()))
    {
      verificaCoordenador();
    } else
    {
      verificaColaborador();
    }
  }
```

Figura 3: Exemplo de Procedimento de Teste.

4º. Passo: Criar um Procedimento de Teste Genérico

Após termos o arquivo Java, gerado a partir do script com a captura das ações do teste, precisamos criar um Procedimento de Teste que possa ser utilizado para testar todas as situações envolvidas nessa parte do sistema. Isso incluir testar o “caminho feliz”, assim como todas as possíveis exceções que possam ser geradas a partir da execução da funcionalidade. Uma reorganização do programa da Figura 2, para criarmos o procedimento de teste Login, é exibida na Figura 3.

5º. Passo: Criar os casos de teste automatizados

A partir desse procedimento de teste é possível fazer uma série de verificações. Conforme já mencionado, as verificações estão associadas ao “caminho feliz” da funcionalidade, assim como os prováveis casos excepcionais, e que normalmente causam a emissão de alguma mensagem ao usuário. Assim, faz parte do processo de teste pensar nesses casos, para em seguida automatizar sua execução. Na Figura 4 apresentamos os casos de teste gerados para verificar a funcionalidade Login no quantum. Observe que foram criados vários logins corretos, com o intuito de verificar o nível de acesso de cada usuário, assim como tentativas de login utilizando algum valor inválido, que causa a exibição de mensagens de erro.

```
package teste.casoDeTeste;
import org.junit.*;
import com.thoughtworks.selenium.*;
import teste.procedimentoDeTeste.LoginProcedimentosDeTeste;

public class LoginTeste {
  private static LoginProcedimentosDeTeste login = null;
  private static Selenium browser = null;

  @BeforeClass
  public static void iniciaBrowser() throws Exception {
      browser = new DefaultSelenium("localhost",
          4444, "*firefox", "http://localhost/quantum/");
      browser.start();
      login = new LoginProcedimentosDeTeste(browser);
  }
   @Test
  public void testaLoginRoot() throws Exception {
    login.login("root", "root", RolesEnum.ROOT.getValor(), "Administrador do Sistema", "");
    login.sair();
  }
  @Test
  public void testaLoginCoordenador() throws Exception {
    login.login("coord1", "coord1", RolesEnum.COORDENADOR.getValor(), "Coordenador 1 Pereira das Neves", "");
    login.sair();
  }
  @Test
  public void testaLoginDesenvolvedor() throws Exception {
    login.login("desenv1", "desenv1", RolesEnum.DESENVOLVEDOR.getValor(), "Desenvolvedor 1 da Silva Souza", "");
    login.sair();
  }
  @Test
  public void testaLoginNaoCadastrado() throws Exception {
    login.login("abcdef", "123456", RolesEnum.DESENVOLVEDOR.getValor(), "", "Não foi possível realizar a entrada no
sistema.");
  }
  @Test
  public void testaLoginComSenhaErrada() throws Exception {
    login.login("coord1", "coord2", RolesEnum.DESENVOLVEDOR.getValor(), "", "Não foi possível realizar a entrada no
sistema.");
  }
  @Test
  public void testaLoginSemLogin() throws Exception {
    login.login("", "coord1", RolesEnum.DESENVOLVEDOR.getValor(), "", "Não foi possível realizar a entrada no
sistema.");
  }
  @Test
  public void testaLoginSemSenha() throws Exception {
    login.login("coord1", "", RolesEnum.DESENVOLVEDOR.getValor(), "", "Não foi possível realizar a entrada no
sistema.");
  }
  @Test
  public void testaLoginSemLoginESenha() throws Exception {
    login.login("", "", RolesEnum.DESENVOLVEDOR.getValor(), "", "Não foi possível realizar a entrada no sistema.");
  }
  @AfterClass
  public static void finalizaBrowser() {
      browser.stop();
  }
}
```

Figura 4: Exemplo de arquivo com diversos testes para o Login do Quantum.

É interessante salientar que a execução dos testes via selenium exige que uma instância da classe DefaultSelenium seja criada. Isso é feito no método com anotação @BeforeTest no exemplo apresentado na Figura 4.

Além disso, podemos observar que nos testes que envolvem um login com sucesso temos uma chamada a função “sair” logo em seguida. Essa função executa a saída no sistema (logout). Temos que fazer isso por que a realização de um login, apenas acontece com sucesso se não houver usuário logado. Devemos lembrar que esse teste executa ações da mesma forma que um usuário do sistema.

6º. Passo: Execução do Teste

Os testes criados com o Selenium são testes que utilizam o JUnit, de forma similar aos testes de unidade. Sua execução acontece da mesma forma que os testes de unidade, clicando com o botão direito do mouse em cima da classe e solicitando sua execução via JUnit. No entanto, existe uma diferença: para que o teste execute, é necessário que exista uma instância do Selenium RC executando na mesma máquina que contém o sistema a ser testado.

O Selenium RC é a parte do selenium responsável por executar cada um dos comandos existentes nos procedimentos de teste. Assim, quando se solicita a digitação de um texto em um comando de uma tela, na verdade emitimos essa solicitação ao Selenium RC, que por sua vez se comunica com o navegador, via funções JavaScript, e executa o comando solicitado.

O Selenium RC é uma extensão do selenium e precisa ser iniciado antes da execução do teste. A iniciação dele pode ser feita a partir da execução do jar com o servidor: java –jar selenium-server.jar. Caso o selenium rc não esteja ativo, a execução dos comandos não será executada com sucesso. O Selenium Rc pode ser obtido a partir do seguinte endereço: http://selenium-rc.openqa.org/download.jsp.