01

Preenchendo e testando formulários

Transcrição

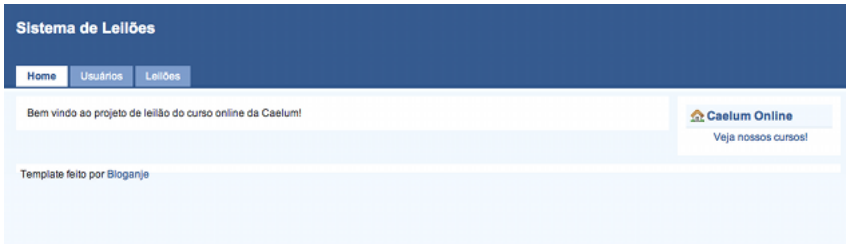
Antes de começarmos a testar, precisamos fazer o download do projeto que utilizaremos ao longo do curso. Como já dito, testaremos um sistema de leilões. Mas não se preocupe, você não precisa conhecer de desenvolvimento web. A aplicação está pronta e apenas a executaremos!

Faça o download do projeto aqui (http://s3.amazonaws.com/caelum-online-public/PM-74/leiloes.zip). Dezipe-o em alguma pasta e, no terminal, digite a seguinte instrução dentro do diretório do projeto:

```
ant jetty.run
```

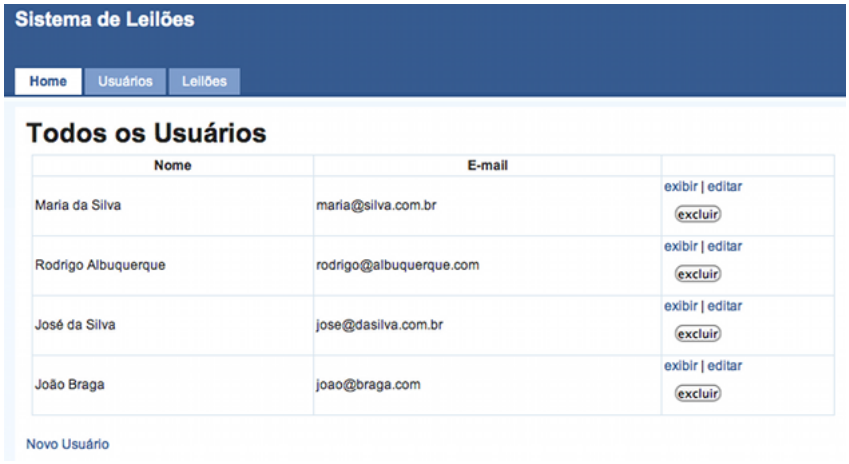
Na primeira vez, essa operação pode levar alguns minutos, pois o ant está baixando todas as dependências necessárias para rodarmos o projeto Java. Mas assim que ele terminar, nosso projeto estará rodando!

Abra o browser no seguinte endereço: `http://localhost:8080 (http://localhost:8080)` . Você deve ver uma aplicação web parecida com essa:



Vamos começar a testar pela funcionalidade de cadastro de usuários! Clique em "Usuários" no menu superior. Cadastre um novo usuário, clicando no link "Novo Usuário". Preencha um nome e um e-mail qualquer:

Clique em "Salvar". O sistema devolve o usuário para a listagem com o novo usuário cadastrado:



Ótimo, aparentemente a funcionalidade está funcionando. Mas nossa experiência nos diz que futuras alterações no sistema podem fazer com que a funcionalidade pare! Vamos então automatizar um teste para o cadastro de um novo usuário. O cenário é o mesmo que acabamos de testar de maneira manual.

Vamos lá. Crie a classe "UsuariosSystemTest". Nela crie o método `main()`:

```
public static void main(String[] args) {

}
```

A primeira parte do nosso teste manual foi entrar na página de cadastro de usuários. Sabemos que a URL é a seguinte: http://localhost:8080/usuarios/new (http://localhost:8080/usuarios/new) . Vamos então fazer o Selenium abrir o Firefox nessa página:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://localhost:8080/usuarios/new");
```

Nessa página, precisamos cadastrar algum usuário. Vamos supor o usuário "Ronaldo Luiz de Albuquerque" com o e-mail "ronaldo2009@terra.com.br". Para preencher esses valores de maneira automatizada, precisamos saber o nome dos campos de texto para que o Selenium saiba aonde colocar essa informação!

Aperte CTRL + U (no Firefox e Chrome) ou Ctrl+F12 (no Internet Explorer) para exibir o código-fonte da página. Veja que o nome dos campos de texto são "usuario.nome" e "usuario.email". Com essa informação em mãos, precisamos 1) encontrar esses elementos na página e 2) preencher com os valores que queremos:

```
// encontrando ambos elementos na pagina
WebElement nome = driver.findElement(By.name("usuario.nome"));
WebElement email = driver.findElement(By.name("usuario.email"));

// digitando em cada um deles
nome.sendKeys("Ronaldo Luiz de Albuquerque");
email.sendKeys("ronaldo2009@terra.com.br");
```

Veja o código acima. Para encontrarmos um elemento, utilizamos o método `driver.findElement` . Como existem muitas maneiras diferentes para encontrar um elemento na página (pelo id, nome, classe CSS, etc), o Selenium nos provê uma classe chamada `By` que tem um conjunto de métodos que nos ajudam a achar o elemento. Nesse caso, como queremos encontrar o elemento pelo seu nome, usamos `By.name("nome-aqui")` .

Tudo preenchido! Precisamos submeter o formulário! Podemos fazer isso de duas maneiras. A primeira delas é clicando no botão que temos na página. Ao olhar o código-fonte da página novamente, é possível perceber que o id do botão de Salvar é "btnSalvar". Basta então pegarmos esse elemento e clicar nele:

```
WebElement botaoSalvar = driver.findElement(By.id("btnSalvar"));
botaoSalvar.click();
```

Uma outra alternativa mais simples ainda é "submeter" qualquer uma das caixas de texto! O Selenium automaticamente procurará o form na qual a caixa de texto está contida e o submeterá! Ou seja:

```
nome.submit();
// email.submit(); daria no mesmo!
```

Se tudo der certo, voltamos a listagem de usuários. Mas, dessa vez, esperamos que o usuário Ronaldo esteja lá. Para terminar nosso teste, precisamos garantir de maneira automática que o usuário adicionado está lá. Para fazer esses tipos de verificação, utilizaremos um framework muito conhecido do mundo Java, que é o JUnit. O JUnit nos provê um conjunto de instruções para fazer esses tipos de comparação e ainda conta com um plugin que nos diz se os testes estão passando ou, caso contrário, quais testes estão falhando!

Para configurar o Eclipse no nosso projeto, clique com o botão direito do mouse sobre o projeto, e vá em `Build Path -> Add Libraries`. Adicione JUnit em sua versão 4.

Para garantir o usuário na listagem, precisamos procurar pelos textos "Ronaldo Luiz de Albuquerque" e "ronaldo2009@terra.com.br" na página atual. O Selenium nos dá o código-fonte HTML inteiro da página atual, através do método `driver.getPageSource()`. Basta então verificarmos se existe o nome e e-mail do usuário lá:

```
boolean achouNome = driver.getPageSource().contains("Ronaldo Luiz de Albuquerque");
boolean achouEmail = driver.getPageSource().contains("ronaldo2009@terra.com.br");
```

Sabemos que essas duas variáveis devem ser iguais a `true`. Vamos avisar isso ao JUnit através do método `assertTrue()` e, dessa forma, caso essas variáveis fiquem com `false`, o JUnit nos avisará:

```
assertTrue(achouNome);
assertTrue(achouEmail);
```

Lembre-se que para o método `assertTrue` funcionar, precisamos fazer o import estático do método: `import static org.junit.Assert.assertTrue;`.

Precisamos agora encerrar o Selenium:

```
driver.close();
```

Por fim, para que o JUnit entenda que isso é um método de teste, precisamos mudar sua assinatura. Todo método do JUnit deve ser público, não retornar nada, e deve ser anotado com `@Test`. Veja:

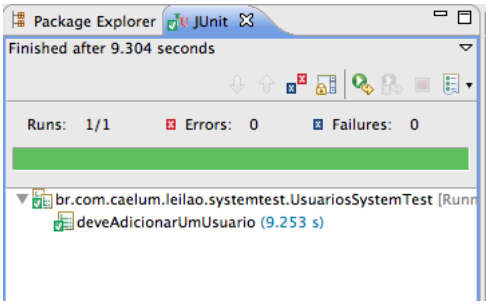
```
@Test
public void deveAdicionarUmUsuario() {
  // ...
}
```

Veja que usamos o nome do método para explicar o que ele testa. Essa é uma boa prática.

Nosso método agora ficou assim:

```
import static org.junit.Assert.assertTrue;

import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

public class UsuariosSystemTest {
    @Test
    public void deveAdicionarUmUsuario() {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://localhost:8080/usuarios/new");

        WebElement nome = driver.findElement(By.name("usuario.nome"));
        WebElement email = driver.findElement(By.name("usuario.email"));

        nome.sendKeys("Ronaldo Luiz de Albuquerque");
        email.sendKeys("ronaldo2009@terra.com.br");
        nome.submit();

        boolean achouNome = driver.getPageSource()
            .contains("Ronaldo Luiz de Albuquerque");
        boolean achouEmail = driver.getPageSource()
            .contains("ronaldo2009@terra.com.br");

        assertTrue(achouNome);
        assertTrue(achouEmail);

        driver.close();
    }
}
```

Repare na anotação `@Test` antes do nome do método. Isso é obrigatório caso queiramos fazer uso do JUnit. Vamos agora executar o teste. Para isso, clique com o botão direito do mouse em cima do código-fonte da classe de teste e selecione `Run As -> Junit Test` , e espere o Selenium executar o teste! Ao final, você deve ver uma tela de confirmação do JUnit:



Pronto! Nosso primeiro teste para a aplicação de leilão está escrito! Mãos à obra!