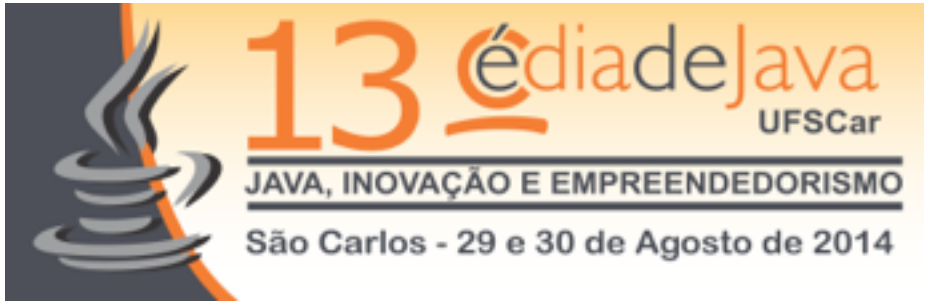
Qualister

Mini-curso

Automação de Testes Funcionais com Selenium Webdriver



contato@qualister.com.br
(48) 3285-5615
twitter.com/qualister
facebook.com/qualister
linkedin.com/company/qualister

Instrutor

Júlio de Lima

Especialista em teste de software com ênfase em automação de testes de software, possui formação em Tecnologia da Informação e certificações internacionais (CTFL e CTAL-TM pelo ISTQB) e nacional (CBTS pela ALATS)


julio.lima@qualister.com.br


twitter.com/juliodelimas


br.linkedin.com/in/juliodelimas

Qualister

- Fundada em 2007
- Mais de 1.000 clientes em todo o Brasil
- Mais de 50 cursos sobre teste de software
- Mais de 3.000 alunos formados
- Áreas de atuação:
  - Consultoria na área de teste qualidade de software
  - Cursos
  - Revenda de ferramentas

Agenda

Automação de Testes Funcionais com Selenium WebDriver

1. Fundamentos em automação de testes
2. Boas práticas
3. Identificando elementos
4. Introdução ao JUnit
5. Sobre o Selenium WebDriver
6. Let's automate!

http://slidesha.re/1B1WXJe

Fundamentos em automação de testes

O que é teste de software?

Manuais e Automatizados

Fundamentos em automação de testes

Fundamentos em automação de testes

Alta velocidade de execução
Alta amplitude e profundidade de testes
Repetitível
Pouco envolvimento humano
Resultados consistentes

Boas práticas

Boas práticas

Concisos: os testes automatizados devem ser tão simples quanto possível, mas não simples demais;

Explícitos: os testes automatizados relatam os desvios por meio de relatórios explicitamente, sem a necessidade de interpretação humana;

Repetíveis: os testes automatizados podem ser executados quantas vezes forem necessárias sem a intervenção humana;

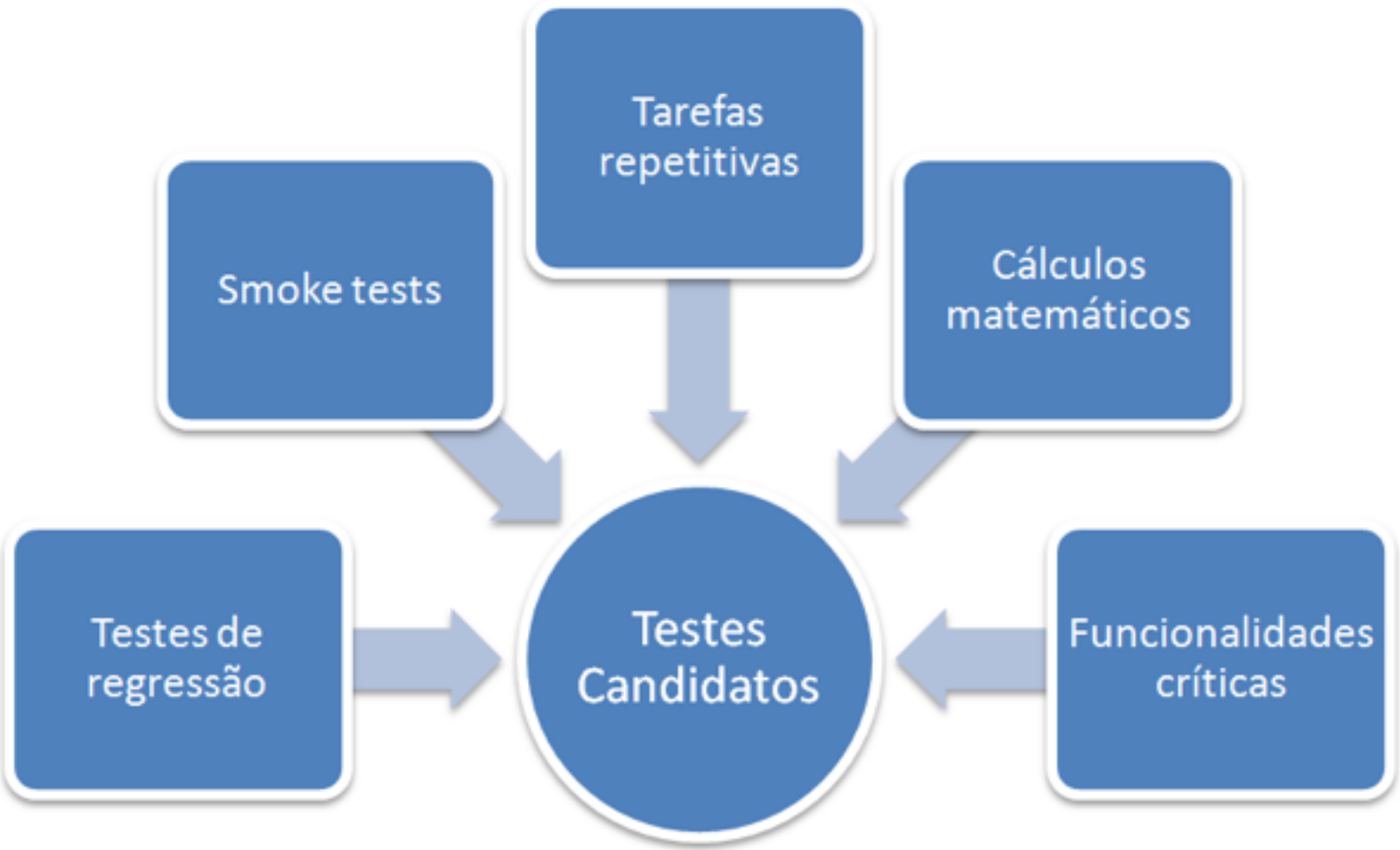
Claros: as instruções codificadas nos testes automatizados devem ser claras e fáceis de entender;

Eficientes: os testes automatizados devem ter um desempenho satisfatório;

Independentes: os testes automatizados devem satisfazer as suas próprias precondições e devem permitir a sua execução em qualquer ordem de maneira independente;

Boas práticas

Boas práticas



Identificando elementos

Identificando elementos

Identificando elementos

```
<form action="http://localhost:8888/quickloja/login/entrar" method="post" class="form-signin">
        <h2 class="form-signin-heading">QuickLoja</h2>
        <div class="control-group">
                <label class="control-label" for="usuariologin">Login</label>
                <div class="controls">
                        <input type="text" name="usuariologin" id="usuariologin" placeholder="Login">
                </div>
        </div>

        <div class="control-group">
                <label class="control-label" for="usuariosenha">Senha</label>
                <div class="controls">
                        <input type="password" name="usuariosenha" id="usuariosenha" placeholder="Senha">
                </div>
        </div>


        <div class="control-group">
                <div class="controls">
                        <button type="submit" class="btn btn-medium btn-primary">Entrar</button>
                </div>
        </div>
        <input type="hidden" name="token" value="f076783a498da86e4fe060bba75dcb87" />
        <input type="hidden" name="ignorarToken" value="true" />
</form>
```

Identificando elementos

- Identificação por ID

  Ex. <input type="text" id="nome" />

- Identificação por NAME

  Ex. <textarea name="nome"></textarea>

- Identificação por CSS

  Ex. <button class="btn btn-medium btn-save" />

- Identificação por XPath

  Ex. //input[@name="cpf"]

Introdução ao JUnit

Introdução ao JUnit

- É um framework de testes para Java
- Vai nos ajudar principalmente a validar os resultados esperados
- Já traz mecanismos visuais para informar se o teste passou ou qual problema de validação foi encontrado

Introdução ao JUnit

- No JUnit, um teste é identificado pela anotação **@Test**
- Com ela o JUnit saberá controlar se um teste deve ser executado
- A anotação **@Test** sempre deve estar acima de um método público

Introdução ao JUnit

- Assertions (asserções) são formas de garantir algum tipo de informação.
- Podemos compará-las como a validação do resultado esperado de um teste
- **Duas asserções muito utilizadas são:**
  - assertTrue
  - assertEquals

Introdução ao JUnit

```
public class ClienteTestes{
    @Test
    public void testeUm(){
        Assert.assertEquals("Jose Silva", mostrarNome());
    }

    public string mostrarNome(){
          return "Jose Silva";
    }
}
```

Sobre o Selenium WebDriver

Sobre o Selenium WebDriver

- É uma API (Application Programming Interface)
- Executa ações em browsers web simulando um usuário
- Como se trata de uma API, é necessário programar/desenvolver os scripts de teste
- Pode ser desenvolvido nas seguintes linguagens nativamente:

Sobre o Selenium WebDriver

- Usaremos a API em Java
- Usaremos o Netbeans para desenvolver os testes em Java
- Usaremos o Junit para suporte aos testes
- Observação:
  Uma API como o WebDriver não tem "integração" com qualquer ferramenta de teste unitário. O que fazemos é usar a API do WebDriver em um código/script juntamente com código de uma ferramenta de teste unitário, como o Junit.

Let`s Automate!

Comandos utilizados no Hands-On

```
FirefoxBinary binary = new FirefoxBinary(new File("C:\
    \Users\\ediadejava\\AppData\\Local\\Mozilla Firefox\
    \firefox.exe"));
FirefoxProfile profile = new FirefoxProfile();
WebDriver driver = new FirefoxDriver(binary, profile);
```

Comandos utilizados no Hands-On

.get("URL") *// Acessar página*

.quit() *// Fechar o browser*

.getTitle() *// Retorna o Title da página*

.findElement(By.estratégia) *// Encontra um WebElement*

.click() *// Clica em um WebElement*

.sendKeys("Texto") *// Digita em um WebElement*

.clear() *// Limpa o conteúdo de um WebElement*

.submit() *// Submete um formulário*

Comandos utilizados no Hands-On

Selecionando valores contidos em Combos

```
Select combo = new
    Select(driver.findElement(By.name("estado")));
combo.selectByVisibleText("SP");
```

Interagindo com Janelas Javascript

```
driver.switchTo().alert();
```

driver.accept() ou driver.dismiss()

Voltando à página

```
driver.switchTo().defaultContent()
```

Comandos utilizados no Hands-On

Tirando um print da tela

```
File scrFile =
    ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File("C:\\temp\\screenshot.png"));
```

Links úteis

- **http://seleniumhq.org**
  Site oficial do Selenium. A documentação e simples e fácil de entender e traz exemplos nas linguagens suportadas

- **http://selenium.googlecode.com**
  Página de desenvolvimento do Selenium. É possível encontrar dados mais técnicos e exemplos mais apurados

Qualister

contato@qualister.com.br

(48) 3285-5615

twitter.com/qualister

facebook.com/qualister

linkedin.com/company/qualister