

# **CURSO DE TESTES AUTOMATIZADOS**

## **TESTES FUNCIONAIS**

### **CASOS DE TESTES E SELENIUM IDE**

Wagner Costa  
wcaquino@gmail.com

# CASOS DE TESTES

- Identificar e comunicar as condições específicas detalhadas que serão validadas para permitir a avaliação de determinados itens do teste alvo

# UTILIZAÇÃO

- Documentar testes manuais.
- Retirar o conhecimento [para testar determinada funcionalidade] da cabeça do analista de testes.
- Servir de insumo para criação de testes automatizados.
- Artefato para auxiliar o cliente na validação do produto.

# CASO DE TESTE TRADICIONAL

Caso de Teste 1		Incluir Serviço	
Seq.	Ação	Dados de Entrada	Resultado Esperado
1	Acionar a opção "Serviço" no menu		O sistema apresenta uma tela para a escolha de tipo de profissional
2	Escolher o tipo de profissional e acionar a opção "Visualizar"	Tipo: Perito Médico	O sistema apresenta uma tela com a lista dos serviços
3	Acionar a opção "Incluir"		O sistema apresenta uma tela para inclusão de serviços
4	Acionar a opção "Confirmar"		O sistema exibe a mensagem: "O campo Nome do Serviço é de preenchimento obrigatório".
5	Informar os dados e acionar opção "Confirmar"	Nome: Serviço 1 Sigla: SRVTST	O sistema exibe a mensagem: "Inclusão realizada com sucesso."

# PONTOS IMPORTANTES

- Linguagem formal
- Cada passo possui um conjunto com “ação” e “reação”
- Pode ser usado como ponto de partida para o roteiro de aceitação do usuário
- Caso tenha alteração no teste, o caso de teste deverá ser alterado também

# LINGUAGEM GHERKIN

## ■ Funcionalidade: Serviço

### ■ Cenário: Incluir Serviço

- Dado que sou um usuário logado no sistema
- Quando clicar no link "serviço"
- Então devo ver a tela para escolha de profissional
- Quando escolher o tipo de profissional Perito Médico
- E clicar no botão "Visualizar"
- Então devo ver a lista dos serviços
- Quando clicar no botão "Incluir"
- Então devo ver o formulário para inclusão de serviços
- Quando clicar no botão "Confirmar"
- Então devo ver a mensagem "O campo Nome do Serviço é de preenchimento obrigatório"
- Quando informar o nome do serviço "Serviço 1"
- E informar a sigla "SRVTST"
- E clicar no botão "Confirmar"
- Então devo ver a mensagem "Inclusão realizada com sucesso"

# PONTOS IMPORTANTES

- Linguagem simples e estruturada
  - Reduz distância entre *stakeholders* e equipe
- Cada linha está relacionada com alguma operação simples
  - Criação, ação ou assertiva
- Documentação viva
  - Especificação executável\*

# EXPRESSÕES-CHAVE EM GHERKINS

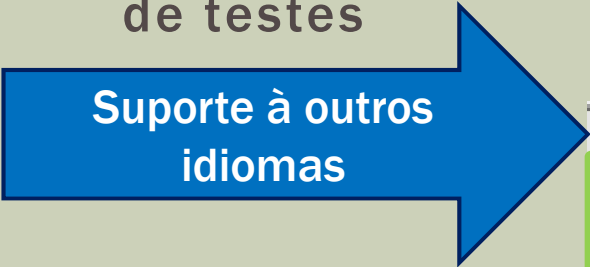
- Funcionalidade
- Cenário
- Dado
- Quando
- Então
- E
- Mas
- Contexto
- Esquema do Cenário
- Exemplos



# FUNCIONALIDADE (FEATURE)

- Descritas e armazenadas em arquivos *.feature*
- Tem por finalidade oferecer um texto descritivo sobre o grupo de testes

Suporte à outros idiomas



```
# language: pt
```

```
Funcionalidade: Deposito
```

```
    O valor da conta deve aumentar a medida que  
    novas quantias são depositadas na conta
```

```
Cenario: Saldo apos depositos
```

```
    Dado que tenho 100 reais na minha conta
```

```
    Quando deposito 50 reais
```

```
    E deposito 70 reais
```

```
    Entao devo ter 220 de saldo
```

# CENARIO (SCENARIO)

- Descrevem algum comportamento esperado pelo sistema
  - Definidos através de passos (*steps*)
- Uma funcionalidade pode ter mais que um cenário
- Geralmente seguem um padrão:
  - Preparar o ambiente conforme desejado
  - “Pertubar” o sistema com alguma ação
  - Verificar o comportamento do sistema

# STEPS: DADO (GIVEN), QUANDO (WHEN) E ENTÃO (THEN)

- Palavras usadas para definir o cenário

```
# language: pt
Funcionalidade: Deposito







    O valor da conta deve aumentar a medida que
    .....
    novas quantias são depositadas na conta

Cenario: Saldo apos depositos
Dado que tenho 100 reais na minha conta
Quando deposito 50 reais
E deposito 70 reais
Entao devo ter 220 de saldo
```

# E (AND) E MAS (BUT)

- Adiciona fluência na leitura do cenário
- Usados para evitar repetições desnecessárias dos termos.

# RESULTADO

- ▲  Funcionalidade: Deposito (0,177 s)
  - ▲  Cenario: Saldo apos depositos (0,169 s)
    -  Dado que tenho 100 reais na minha conta (0,164 s)
    -  Quando deposito 50 reais (0,001 s)
    -  E deposito 70 reais (0,000 s)
    -  Entao devo ter 220 de saldo (0,004 s)

# CONTEXTO (BACKGROUND)

- Quando os cenários compartilham o mesmo ponto de partida (Given – Dado – Contexto inicial), a expressão “Dado” de cada cenário pode ir para o bloco de Contexto.

# CÓDIGO

```
import org.junit.Assert;

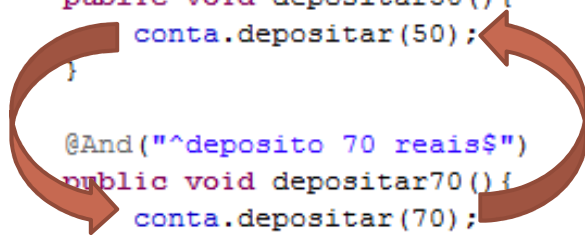
@RunWith(Cucumber.class)
public class ContaTestSteps {
    private Conta conta;

    @Given("^que tenho 100 reais na minha conta$")
    public void iniciarConta() {
        conta = new Conta();
        conta.setSaldo(100);
    }

    @When("^deposito 50 reais$")
    public void depositar50() {
        conta.depositar(50);
    }

    @And("^deposito 70 reais$")
    public void depositar70() {
        conta.depositar(70);
    }

    @Then("^devo ter 220 de saldo$")
    public void checarSaldo220() {
        Assert.assertEquals(220, conta.getSaldo());
    }
}
```



# STEPS PARAMETRIZÁVEIS

```
import org.junit.Assert;
import org.junit.runner.RunWith;

import cucumber.annotation.pt.Dado;
import cucumber.annotation.pt.Entao;
import cucumber.annotation.pt.Quando;
import cucumber.junit.Cucumber;

@RunWith(Cucumber.class)
public class ContaTestSteps {
    private Conta conta;

    @Dado("^que tenho (\\d+) reais na minha conta$")
    public void que_tenho_reais_na_minha_conta(int saldo) {
        conta = new Conta();
        conta.setSaldo(saldo);
    }

    @Quando("^deposito (\\d+) reais$")
    public void deposito_reais(int valor) {
        conta.depositar(valor);
    }

    @Entao("^devo ter (\\d+) de saldo$")
    public void devo_ter_de_saldo(int saldoPrevisto) {
        Assert.assertEquals(saldoPrevisto, conta.getSaldo());
    }
}
```



# MAPEAMENTO COM FEATURE

- O *.feature* não necessita de alteração

```
# language: pt
Funcionalidade: Deposito
```

```
    O valor da conta deve aumentar a medida que
    novas quantias são depositadas na conta
```

```
Cenário: Saldo apos depositos
```

```
    Dado que tenho 100 reais na minha conta
```

```
    Quando deposito 50 reais
```

```
    E deposito 70 reais
```

```
    Entao devo ter 220 de saldo
```

```
@RunWith(Cucumber.class)
public class ContaTestSteps {
    private Conta conta;

    @Dado("^que tenho (\\d+) reais na minha conta$")
    public void que_tenho_reais_na_minha_conta(int saldo) {
        conta = new Conta();
        conta.setSaldo(saldo);
    }

    @Quando("^deposito (\\d+) reais$")
    public void deposito_reais(int valor) {
        conta.depositar(valor);
    }

    @Entao("^devo ter (\\d+) de saldo$")
    public void devo_ter_de_saldo(int saldoPrevisto) {
        Assert.assertEquals(saldoPrevisto, conta.getSaldo());
    }
}
```

# ESQUEMA DO CENÁRIO E EXEMPLOS

- Alternativa para testar diferentes resultados com cenários semelhantes

Esquema do Cenário: Depósitos variados

Dado que tenho 100 reais na minha conta

Quando deposito <deposito1> reais









E deposito <deposito2> reais

Entao devo ter <saldo> de saldo

Exemplos:

deposito1	deposito2	saldo
100	0	200
300	50	450

# RESULTADO DO ESQUEMA

- ▲  Esquema do Cenário: Depósitos variados (0,006 s)
  - ▲  Exemplos: (0,006 s)
    - ▶  | 100 | 0 | 200 | (0,002 s)
    - ▲  | 300 | 50 | 450 | (0,002 s)
      -  Dado que tenho 100 reais na minha conta (0,000 s)
      -  Quando deposito 300 reais (0,000 s)
      -  E deposito 50 reais (0,000 s)
      -  Entao devo ter 450 de saldo (0,002 s)

- Os *step definitions* não foram alterados



**SELENIUM**

# PARA QUE SERVE O SELENIUM?



# UTILIDADES DO SELENIUM

- Auto falante
- Anticoncepcional
- Óleo de carro
- Fabricar barbante cheiroso
- Retificador
- Elemento da tabela periódica
- Shampoo anti-caspa
- Complexo vitamínico
- Calmante
- Ferramenta para Testes
  - Origem do nome?

# FERRAMENTA PARA TESTES

- Selenium é uma ferramenta para automatizar os testes de aplicações web em várias plataformas (browsers) e sistemas operacionais diferentes.



# SABORES

- Selenium IDE



- Selenium WebDriver

- Sucessor do Selenium RC



- Selenium Grid





# SELENIUM IDE

- Extensão do Firefox
- **Record:** grava todos os cliques, escritas e várias outras ações executadas na página para fazer o teste.
- **Playback:** Após o teste gravado, basta apertar o play para executá-lo sempre que necessário



# SELENIUM WEBDRIVER

- Roda os testes em múltiplos browsers e plataformas.
- Permite Escrever seus testes na sua linguagem de programação preferida
- Sucessor do Selenium Remote Control



# SELENIUM GRID

- Distribui os seus testes entre vários servidores
- Economia de tempo rodando testes em paralelo
- Algumas empresas utilizam para testes de carga / desempenho

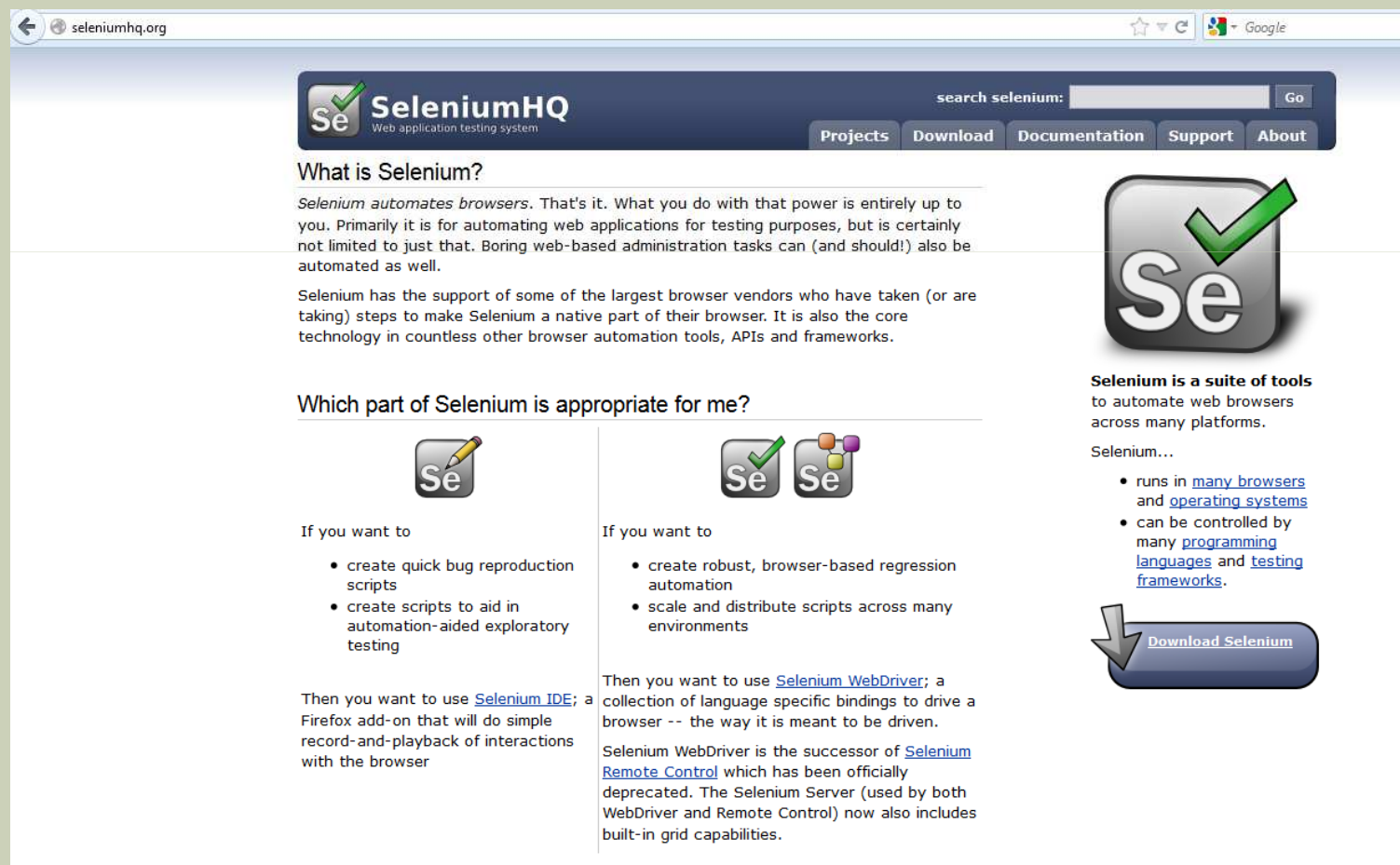


The image features the Selenium IDE logo, which consists of two vertical rectangular blocks. The left block is a large rectangle filled with a solid reddish-brown color. The right block is a narrower vertical rectangle filled with a solid dark grey color. The text 'SELENIUM IDE' is centered horizontally within the reddish-brown block.

**SELENIUM IDE**

# COMO INSTALAR O SELENIUM IDE?

## ■ Basta adicionar o plugin do Firefox



The screenshot shows the SeleniumHQ website in a browser window. The address bar shows 'seleniumhq.org'. The website has a dark blue header with the SeleniumHQ logo and a search bar. Below the header, there are navigation links: Projects, Download, Documentation, Support, and About. The main content area is divided into two columns. The left column is titled 'What is Selenium?' and contains text about Selenium's capabilities. The right column is titled 'Which part of Selenium is appropriate for me?' and contains two sections: 'If you want to' and 'If you want to'. Each section has a list of bullet points and a link to download Selenium. The 'If you want to' section has a link to 'Selenium IDE' and the 'If you want to' section has a link to 'Selenium WebDriver'. There is also a large 'Download Selenium' button on the right side of the page.

**SeleniumHQ**  
Web application testing system

search selenium:  Go


[Projects](#) [Download](#) [Documentation](#) [Support](#) [About](#)

### What is Selenium?

*Selenium automates browsers.* That's it. What you do with that power is entirely up to you. Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.


### Which part of Selenium is appropriate for me?



If you want to

- create quick bug reproduction scripts
- create scripts to aid in automation-aided exploratory testing

Then you want to use [Selenium IDE](#); a Firefox add-on that will do simple record-and-playback of interactions with the browser




If you want to

- create robust, browser-based regression automation
- scale and distribute scripts across many environments

Then you want to use [Selenium WebDriver](#); a collection of language specific bindings to drive a browser -- the way it is meant to be driven.

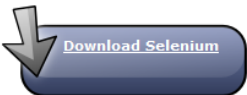
Selenium WebDriver is the successor of [Selenium Remote Control](#) which has been officially deprecated. The Selenium Server (used by both WebDriver and Remote Control) now also includes built-in grid capabilities.



**Selenium is a suite of tools** to automate web browsers across many platforms.

Selenium...

- runs in [many browsers](#) and [operating systems](#)
- can be controlled by many [programming languages](#) and [testing frameworks](#).

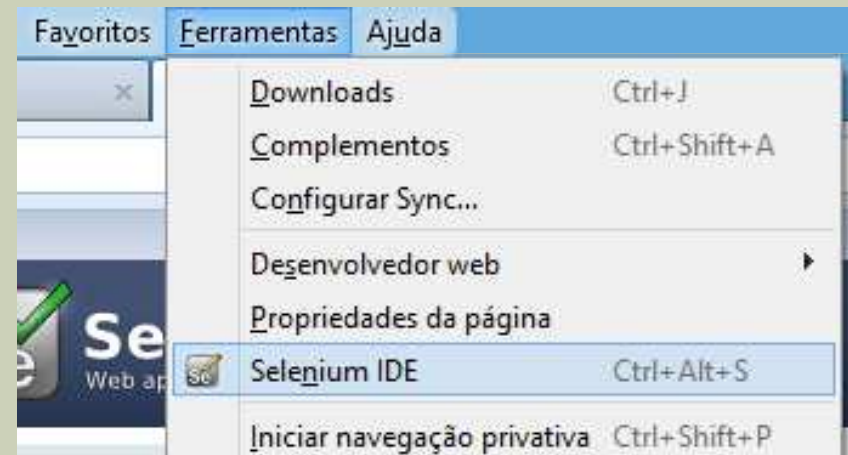
 [Download Selenium](#)

# ABRINDO A IDE

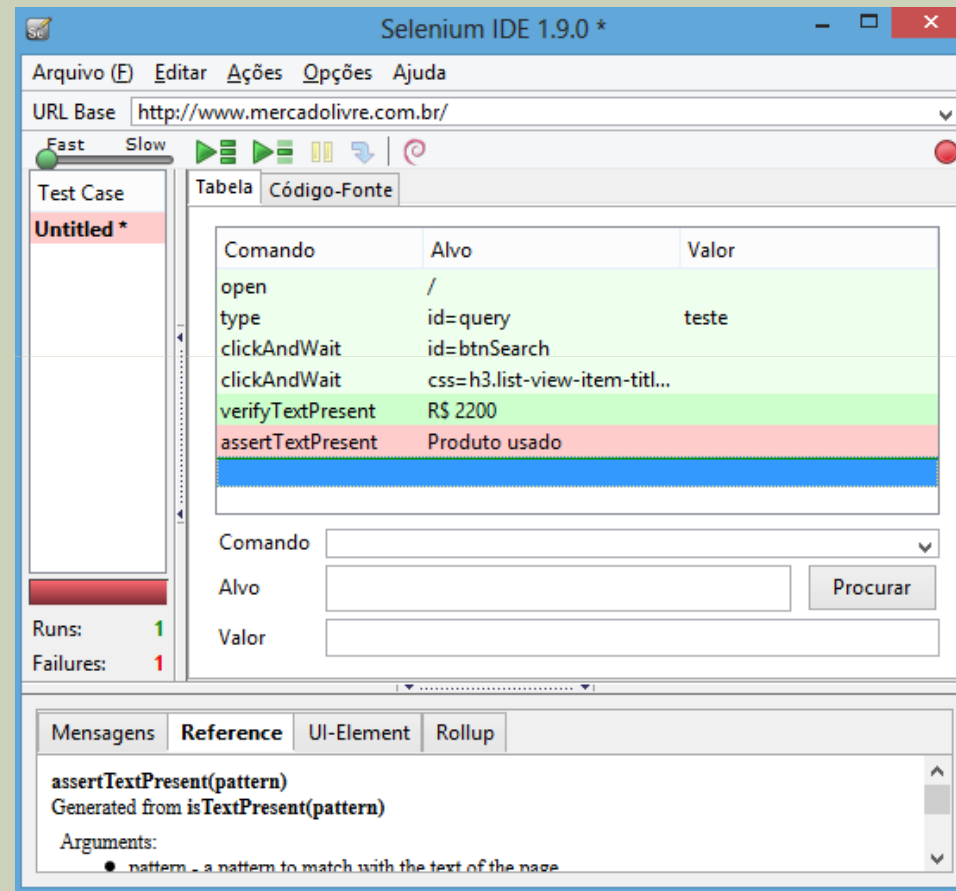
## Aba Lateral



## Janela



# IDE



# PRINCIPAIS COMANDOS SELENIUM - SELENESE

- **Actions:** Manipulam o estado da aplicação
  - Click...
  - Select...
  - ...AndWait
  
- **Acessors:** Examinam o estado e armazenam em variáveis
  - Store...
  
- **Assertions:** Parecidos com os acessors, com a adição de assertivas
  - Assert...
  - Verify...
  - WaitFor...



# VERIFY E ASSERT

- **Verify:** Os comandos de verificação localizam um elemento existente na página SEM INTERROMPER a execução do script de erro.
  - `VerifyText(locator, texto)`
  - `VerifyTextPresent(texto)` - *Deprecated*
  - `AssertElementPresent(locator)`
- **Assert:** Os comandos de afirmação localizam um elemento existente na página e INTERROMPE a execução do script após algum erro.
- Expressões regulares podem ser usadas.

# LOCATOR

- **Forma para localizar unicamente um elemento na página.**
  - **Id**
  - **Name**
  - **Xpath**
  - **DOM**
  - **Links**
  - **CSS**

# Localizar por ID

```
1 <html>
2   <body>
3     <form id="loginForm">
4       <input name="username" type="text" />
5       <input name="password" type="password" />
6       <input name="continue" type="submit" value="Login" />
7     </form>
8   </body>
9 </html>
```

- `id=loginForm (3)`

# Localizar por name

```
1  <html>
2  <body>
3    <form id="loginForm">
4      <input name="username" type="text" />
5      <input name="password" type="password" />
6      <input name="continue" type="submit" value="Login" />
7      <input name="continue" type="button" value="Clear" />
8    </form>
9  </body>
10 </html>
```

- name=username (4)
- name=continue value=Clear (7)
- name=continue Clear (7)
- name=continue type=button (7)

# Localizar por XPath

```
1 <html>
2 <body>
3   <form id="loginForm">
4     <input name="username" type="text" />
5     <input name="password" type="password" />
6     <input name="continue" type="submit" value="Login" />
7     <input name="continue" type="button" value="Clear" />
8   </form>
9 </body>
10 </html>
```

- `xpath=/html/body/form[1]` (3) - Absolute path (would break if the HTML was changed only slightly)
- `//form[1]` (3) - First form element in the HTML
- `xpath=//form[@id='loginForm']` (3) - The form element with @id of 'loginForm'
- `xpath=//form[input/@name='username']` (4) - First form element with an input child element with @name of 'username'
- `//input[@name='username']` (4) - First input element with @name of 'username'
- `//form[@id='loginForm']/input[1]` (4) - First input child element of the form element with @id of 'loginForm'
- `//input[@name='continue'][@type='button']` (7) - Input with @name 'continue' and @type of 'button'
- `//form[@id='loginForm']/input[4]` (7) - Fourth input child element of the form element with @id of 'loginForm'

# Localizar links

```
1 <html>
2   <body>
3     <p>Are you sure you want to do this?</p>
4     <a href="continue.html">Continue</a>
5     <a href="cancel.html">Cancel</a>
6   </body>
7 </html>
```

- link=Continue (4)
- link=Cancel (5)

# Localizar por DOM

```
1 <html>
2   <body>
3     <form id="loginForm">
4       <input name="username" type="text" />
5       <input name="password" type="password" />
6       <input name="continue" type="submit" value="Login" />
7       <input name="continue" type="button" value="Clear" />
8     </form>
9   </body>
10 </html>
```

- `dom=document.getElementById('loginForm')` (3)
- `dom=document.forms['loginForm']` (3)
- `dom=document.forms[0]` (3)
- `document.forms[0].username` (4)
- `document.forms[0].elements['username']` (4)
- `document.forms[0].elements[0]` (4)
- `document.forms[0].elements[3]` (7)

# Localizar por CSS

```
1 <html>
2   <body>
3     <form id="loginForm">
4       <input name="username" type="text" />
5       <input name="password" type="password" />
6       <input name="continue" type="submit" value="Login" />
7     </form>
8   </body>
9 </html>
```

- `css=form#loginForm` (3)
- `css=input[name="username"]` (4)
- `css=input.required[type="text"]` (4)
- `css=input.passfield` (5)
- `css=#loginForm input[type="button"]` (4)
- `css=#loginForm input:nth-child(2)` (5)



# CLICK

- Os comandos click executam a ação de um click em botões ou links existentes na página
  - Click(locator)
  - ClickAt(x,y)

## ...ANDWAIT E WAITFOR...

- Os comandos wait são comandos de espera. Essa espera pode ser por uma ação ou elemento da página e após a identificação do elemento o script volta a ser executado.
- **andWait:** Utilizado quando a ação causa uma navegação para outra página ou “recarrega” a atual
  - `clickAndWait`
  - `selectAndWait`
  - `doubleClickAndWait`
- **waitFor:** Aguarda até que alguma condição desejada ocorra. Utilizado principalmente em aplicações com AJAX.
  - `waitForTextPresent`
  - `waitForElementPresent`
  - `waitForConfirmation`

# STORE

- Permite armazenar valores ou constantes para serem utilizadas durante o teste
  - `storeValue(value, variableName)`
  - `storeTitle(variableName)`
  - `storeElementPresent(locator, variableName)`
- Para recuperar o valor da variável, utiliza-se a seguinte notação:
  - `${variável}`

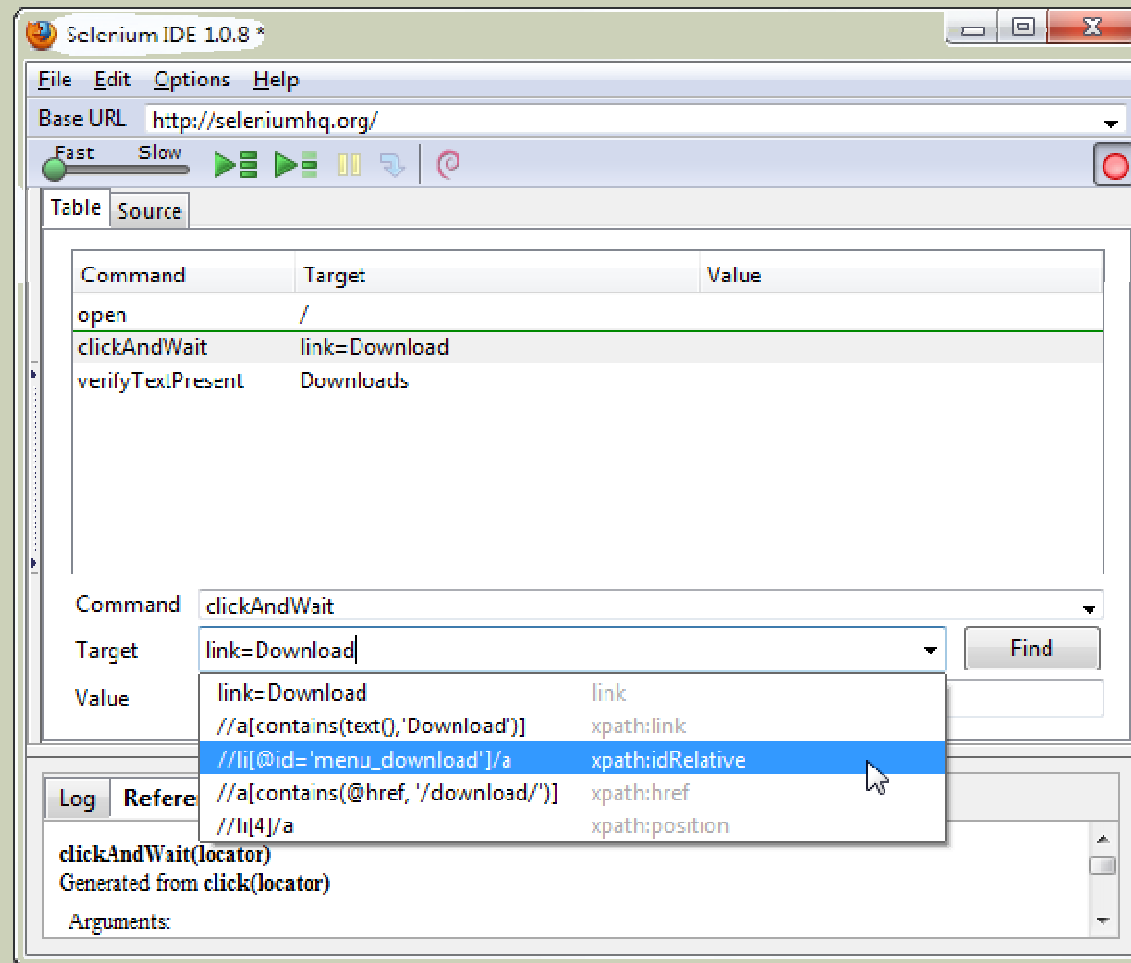
# ALERTS

- Se for informado ao selenium que terá um alerta, ele irá passar por ele normalmente
  - `AssertAlert(texto)`
  - `VerifyAlert(texto)`
  - `verifyAlertPresent()`
- Confirmation: Por ter mais que uma opção, o Selenium precisa saber o que fazer
  - `chooseCancelOnNextConfirmation`
  - `chooseOkOnNextConfirmation`
  - `assertConfirmation(texto)`

# PONTOS FORTES

- Facilidade para gravação e execução dos testes;
- Uso da ferramenta sem conhecimento de linguagem de programação;
- Identificação automática dos campos da tela por seus IDs, nomes ou Xpaths.
- Possibilidade de ajustes nos testes durante e após o processo de gravação;
- Auto preenchimento para os comandos mais comuns;
- Possibilidade de salvar os testes em várias linguagens\* (execução no RC ou WebDriver);
- Possibilidade de execução passo-a-passo dos testes, break points e debug;
- Ferramenta Open Source com comunidade forte;

# LOCATOR ASSISTANCE



# PLUGINS

- Gerais: Firebug / Firepath
- Highlight
- Implicit Wait
- Page Coverage
- Screenshot on fail
- <http://docs.seleniumhq.org/download/>

# LIMITAÇÕES

- Só é possível testar sistemas Web.
- Só existe IDE de gravação para o Mozilla Firefox.
- Não é possível interagir com os documentos que são exibidos dentro de um plug in do browser, como PDF, DOC, AVI etc.
- Não pode ser utilizado em integração contínua.



# MAIS CONTEÚDO...

- Documentação oficial:  
[http://docs.seleniumhq.org/docs/02\\_selenium\\_ide.jsp](http://docs.seleniumhq.org/docs/02_selenium_ide.jsp)
- Blog de um dos mantenedores (e criador de plugins do Selenium IDE): <http://blog.reallysimplethoughts.com/>
- Blog do Elias Nogueira: <http://www.eliasnogueira.com/>
- The cucumber book
- Testes funcionais de Software – Leonardo Molinari
- Inovação e automação de testes de software - Leonardo Molinari

# MÃOS À OBRA

- Criar 3 cenários de testes
  - Escolher um sistema Web público
  - Criar um roteiro contendo 3 cenários de testes com, pelo menos, 10 ações (pares *When* e *Then*) executados no sistema ao todo.
    - Roteiro pode ser tradicional ou Gherkin.
- Instalar o Selenium IDE no Firefox.
  - O plugin pode ser baixado em [seleniumhq.org](http://seleniumhq.org)
- Automatizar os cenários no Selenium IDE
  - Testar no modo rápido e no modo lento