**Automação de Testes com Selenium**

A automação de testes vem como um meio de manter a eficiência da QA sem. necessariamente, aumentar os custos.

Forma

Descrição gerada automaticamenteEm um cenário real, há diversas reestruturações de escopo de projetos, o que, muitas vezes, aumenta o volume de testes necessários. Com a automação, podem-se definir testes a serem automatizados e, assim, melhorar a eficiência da equipe e evitar o aumento de custos com testes.

**Pirâmide Escopo - Tempo - Custo, com a Qualidade ao centro,** que é interferida por essas três variáveis. A automação de testes quebra esse paradigma e melhora a velocidade e qualidade de entrega de produtos.

**5W2H – 7 perguntas a serem feitas sobre testes:**

- **What**? – O que deve ser automatizado;

- **Who**? – Quem da equipe possui o perfil para a atividade – conhecimentos lógicos, linguagens de programação, etc.;

- **Where**? – Decidir se será em um projeto à parte ou em um ambiente compartilhado com o desenvolvimento;

- **When**? – Saber o momento certo de automatizar;

- **Why**? – Descobrir os reais motivos para a implementação de automação nos testes – questão de prazos? Questões de qualidade?

- **How**? – Definir as técnicas e ferramentas a serem utilizadas no processo de automação – quais serão as técnicas utilizadas para a implementação (testes de regressão, performance. Quais ferramentas serão utilizadas?

- **How** **Much**? – Calcular os custos para realizar a automatização e verificar se compensa aos resultados propostos – Verificar os custos *versus* os benefícios da automação dos testes;

**Princípios da Automação de Testes:**

- **Projetar os casos de testes primeiramente** para, depois, automatizar – Definir e desenhar os casos de testes para, depois, fazer a automatização;

- **Focar nas prioridades** – Não pensar em automatizar 100% da funcionalidade e priorizar as validações principais, dentro das regras do negócio; **Obs.: Na minha opinião, devemos focar no principal para a automação e aplicar no restante organicamente, quando houver disponibilidade para tal.**

- **Projete a automação de testes de forma que facilite revisões** – Facilitar a interpretação dos testes automatizados. Por exemplo: comentar na linha de código da automação, explicar o que fazem os comandos. Tentar deixar o mais fácil possível para futuros profissionais conseguirem interpretar corretamente o funcionamento do teste;

- **Inicie a automação de testes o mais cedo possível** em um projeto – Automatizar a partir da primeira sprint, se possível.

**SELENIUM**

Selenium é uma **suíte** criada em 2004, composta por 4 **ferramentas** de automação para aplicações web: **IDE**, **Remote Control**, **WebDriver** e **GRID**, que serão expostas mais à frente neste documento.

O Selenium permite:

- Criar e executar scripts de testes independentemente do browser ou sistema operacional utilizado;

- Realizar Testes de Estresse/Carga através da execução de testes em paralelo: utilização de diversas Treads em diversos dispositivos e plataformas virtuais. Exemplo: Cadastro simultâneo de diversos usuários ao mesmo tempo, cada um em plataformas diferentes;

- Adicionar plug-ins (Selenium IDE), permitindo scripts de testes mais robustos e que atendem às necessidades dos negócios: Laços, Estruturas de Condições (condicionais), exportação de relatórios, evidências testes em erros;

- Integrar os scripts de teste a um projeto de teste em diversas linguagens: possibilidade de converter scripts do Selenium em me arquivos na linguagem em que a o sistema está sendo desenvolvido, como Java, C#, PHP, Python ou Ruby.

**Selenium IDE**

O Selenium IDE (Integrated Development Environment) é uma ferramenta de automação de testes de software, sendo um dos componentes do conjunto de ferramentas Selenium, que é amplamente utilizado para testes de aplicativos web. É uma extensão do navegador que permite gravar, editar e reproduzir testes de forma interativa em navegadores da web, especialmente utilizada para testes de regressão, onde podem ser gravadas diversas ações interativas (cliques em botões, preenchimento de campos diversos, preenchimento de formulários, etc.) e reproduzi-las posteriormente para garantir que o aplicativo continue funcionando corretamente após mudanças ou atualizações.

A interface gráfica Selenium IDE possui os seguintes elementos principais:

1. Barra de Ferramentas – possui as funcionalidades que gerenciam os testes, tais como gravar, executar, pausar, etc.;
2. Lista de Casos de Teste – lista dos casos de teste que compõem a suíte de teste em uso.
3. Editor de Script – espaço para editar o script do caso de teste selecionado, podendo definir o step em que o teste irá iniciar ou parar;
4. Rodapé – contém informações de log da execução do caso de teste selecionado.

Podemos criar uma lista de métodos para executar um bloco de comandos no Selenium, tornando a codificação de testes mais enxuta.

- Reference – Documentação dos comandos do Selenium – documentação com o detalhamento de cada comando;

- UI-Element – Identificação do Elemento. Referência de nome dos elementos da tela, que podem ser customizáveis;

- Rollup – Forma de listar todos os métodos criados, que são chamados no script;

- Informações – Exibe as informações do log;

- Limpar – Limpar o Log que está sendo gerado.

**Comandos Usuais:**

- Click – Clique do mouse (pode ser clickAndWait, que clica e aguarda a página carregar, dependendo do que está sendo testado);

- Type – Preenchimento de campos diversos (nome, e-mail, texto, etc.).

Campo **Alvo**: Campo que possibilita indicar um “campo-alvo” para a interação com a ação específica utilizada. Por exemplo, um click em um botão específico, onde o “campo-alvo” seja o nome desse botão, ou um “type” em um “campo-alvo” específico (os nomes dos “**Alvos**” correspondem aos nomes específicos dados a esses campos específicos na programação do sistema a ser automatizado.

**Scritps de Testes:**

O Selenium IDE possibilita o desenvolvimento de scripts de testes em diversos formatos, dentre eles:

- Testar uma aplicação que esteja rodando com C#.NET e MySQL Workbench na máquina local;

- Testar quaisquer outros scripts ou URL’s desejados;

- Gravar e replicar os testes projetados – com o Selenium IDE é possível gravar todas as ações efetuadas na url e replicar posteriormente, capturando as ações executadas pelo usuário.

**O Selenium identifica automaticamente as ID’s de todos os componentes na aplicação web, na seguinte ordem de prioridade:**

**1º - ID;**

**2º - Name;**

**3º - CSS Selector;**

**4º - XPE.**

**Validação da Tela de testes:**

A primeira validação, e mais importante, a ser realizada, é a validação sem o preenchimento de nenhum campo (no caso de um teste em um tipo “formulário”).

Em um ambiente de testes, antes de proceder a qualquer tipo de teste, é importante proceder à validação dos campos principais do formulário web. Para fazer tal validação, devemos:

- Clicar com o botão direito do mouse sobre o **Elemento** desejado, como Título, Campo, Namespace, Texbox, ou outro elemento qualquer desejado na interface de teste que está sendo verificada. Assim, abrirá o menu com as opções de validação de campos como, por exemplo:

> **assert: VALIDAÇÃO**;

> **store: ARMAZENAMENTO:** armazena o valor de um elemento em uma variável;

> **assertElementPresent:** valida se o elemento selecionado está presente em tela, independentemente do tipo ou do valor do elemento em questão;

> **verifyValue** – validação de valores de um campo que comporte valores;

> **assertText:** validação da presença de textos em campos.

Scripts de testes são o passo-a-passo dos testes gravados no Selenium. Podemos gravar diversos scripts de diversos testes.

Boas práticas:

- Validação da tela de testes antes de iniciar um teste;

- Testar os fluxos principais básicos primeiramente e, depois, proceder aos fluxos alternativos diversos que estejam no escopo do projeto;

- Nomear os casos de testes seguindo o padrão **CT01**NomeDoCasoDeTeste;

- Comentar ao máximo possível as ações dos scripts de testes, a fim de facilitar a utilização e a manutenção posterior dos desses.

Para inserir comentários no script de testes no Selenium IDE 3.17.0, rodando no navegador Edge, basta inserir uma linha de comando iniciar a descrição com duas barras “//” no campo “Command”.

**DEPRECATED // Plugins:**

O Selenium IDE possui uma variedade de plugins para browsers ou arquivos .js, cujo caminho é apontado no menu “Opções” da IDE. Abaixo, podemos verificar alguns desses plugins:

- Selenium IDE Button: Permite alternar a exibição da IDE do Selenium, seja em um pop-up ou no próprio frame do browser;

- Flow Control: Adiciona comandos de repetição ao script de teste;

- ScreenShot on Fail: Registra um print da tela quando ocorre um erro na execução do teste;

- Pretty Report: Exporta os resultados de testes em um relatório com um visual mais bonito e legível.

**Obs.:** Aula de 2021, portanto, pode ser que alguns plugins não existam ou não sejam mais necessários.

**Estruturas Condicionais:**

O Selenium IDE 3.17.0, rodando em Edge, nos possibilita criar estruturas condicionais com os elementos da tela. Por exemplo, se quisermos verificar se um campo possui ou não informação, podemos fazer uma *if-else* para verificar essa condição. Diversas outras aplicações de condição podem ser efetuadas.

**Exemplo de Estrutura Condicional:**

No exemplo abaixo, podemos verificar uma estrutura condicional onde:

- Comando *store title*: armazena o do título (do formulário, no caso) na variável “title”;

- Target do comando *store title*: ${title}==”Sellers - SalesWebMvc”. A sintaxe compara a variável “title” com o texto entre aspas duplas;

- Comando *echo*: Como em Java, exibe na tela o texto do seu Target;

- Não preciso explicar o *else*.

****

**Estrutura *times*:**

Repete um bloco de comandos *x* vezes.

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

**Instalação de drivers do Selenium usando Node.js:**

- Instalar o Node.js;

- Abrir o prompt de comando:

- Verificar se foi instalado corretamente com o comando ***node -v*** no prompt de comando;

- Instalar o pacote *nmp siderunner* do Selenium com o comando ***npm install -g Selenium-side-runner***;

- Instalar o driver para o browser utilizando os comandos ***npm install -g edgedriver*** para Edge, ***npm install -g chromedriver*** para Chrome e ***npm install -g geckodriver*** para Firefox. Lista disponível no link <https://www.selenium.dev/selenium-ide/docs/en/introduction/command-line-runner>.

- Verificar as instalações na pasta C:\Users\Usuário\AppData\Roaming\npm:

Texto

Descrição gerada automaticamente com confiança média

- Criar variáveis de ambiente.

Comando para rodar testes pelo prompt de comando: ***selenium-side-runner C:\Users\coelh\Pessoal\TI\CURSOS\Selenium\TestSelenium2023\TesteSelenium2023\_2\SalesWebMvcTests.side***;

Comando para rodar testes em background: ***selenium-side-runner --config-file C:\Users\coelh\Pessoal\TI\CURSOS\Selenium\TestSelenium2023\TesteSelenium2023\_2\config.yaml C:\Users\coelh\Pessoal\TI\CURSOS\Selenium\TestSelenium2023\TesteSelenium2023\_2\SalesWebMvcTests.side***.