Testes com Selenium

Marco Mangan 26/07/2024

Visão Geral do Curso

Cinco encontros

- Dia 1: Qualidade de software
- Dia 2: Configuração do ambiente
- Dia 3: Escrita e execução de testes
- **Dia 4: Escrita e execução de testes**
- Dia 5: Integração com DevOps

Objetivos

- Apresentar práticas de programação como Page Objects
- Apresentar o uso de sincronização com Wait
- Reforçar as oito etapas de testes
- Apresentar técnicas de geração de casos de testes
- Apresentar diagramas como recurso de comunicação visual

Introdução à Teste de Software

Dia 4 (26/07/2024, sexta-feira)

- 14h00 - 14h30: Wait e Page Object
- 14h30 - 15h30: Seletores XPath, Web Element (CSS Selector) e DOM (JS)
- 15:30 - 15h45: Intervalo
- 15h45 - 17h00: Oito etapas de teste e diagramas

Page Objects e Wait

- Técnicas adicionais de programação de testes, usar quando possível e necessário
- **Page Object** é uma forma de organizar uma classe auxiliar que contém seletores
  - Aplicado em reutilização de código entre testes, necessidade de nível de abstração adicional [Lab08]
- **Wait** espera condição específica antes de continuar a execução
  - Aplicado em conexão de baixa velocidade ou alta latência, tempo de resposta alto, requisito de desempenho e usabilidade, verificação de estrutura (elemento existe?) [Lab09]

Page Objects e Wait

Conclusão

- Ao encontrar dificuldade de automação, execute o teste manualmente com Selenium IDE
    - Ao final, registre manualmente o resultado do teste, como se fosse automatizado. Não atrase a execução do teste!
- Use o código exportado pelo Selenium IDE como base para automação
- Caso ocorram problemas com sincronização, ou se for uma prática da equipe, adicione o ExpectedCondition ou Wait
- Caso ocorra mais de três testes similares, recorra à parametrização, refatoração e Page Objects. Trate sempre de fatias VERTICAIS! Nunca horizontais!

Seletores

- Obter uma referência para ler ou escrever em um elemento da aplicação
- XPath, Web Element, DOM

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Página de Exemplo</title>
</head>
<body>
    <form id="exampleForm">
        <label for="name">Nome:</label>
        <input type="text" id="name" name="name">
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

XPath

- Expressivo: Pode navegar através de elementos e atributos em um documento **XML**
- Travessia: Pode atravessar tanto para cima quanto para baixo no DOM, permitindo consultas mais complexas
- Preciso: Pode localizar elementos mesmo quando há múltiplos elementos com propriedades semelhantes

- Desempenho: A execução pode ser mais lenta, especialmente em árvores grandes
- Complexidade: A sintaxe pode ser complexa e difícil de ler e manter
- Inconsistências: Navegadores podem ter implementações parciais ou inválidas

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

class XPathExample
{
    static void Main()
    {
        IWebDriver driver = new ChromeDriver();
        driver.Navigate().GoToUrl("file:///caminho/para/seu/exemplo.html");

        // Localize o elemento de entrada usando XPath
        IWebElement inputElement = driver.FindElement(By.XPath("//input[@id='name']"));

        // Execute ações no elemento
        inputElement.SendKeys("João Silva");

        // Feche o navegador
        driver.Quit();
    }
}
```

Web Element

CSS Selector

- Velocidade: Geralmente são mais rápidos que seletores XPath na maioria dos navegadores
- Simplicidade: A sintaxe é muitas vezes mais simples e intuitiva para aqueles familiarizados com CSS (***Cascading Style Sheet***)
- Uso Generalizado: Amplamente utilizados no desenvolvimento Web

- Travessia: Seletores CSS só podem atravessar para baixo no DOM, não permitindo a mesma flexibilidade do XPath
- Capacidades Limitadas: Menor capacidades de consulta que o XPath.

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

class CssSelectorExample
{
    static void Main()
    {
        IWebDriver driver = new ChromeDriver();
        driver.Navigate().GoToUrl("file:///caminho/para/seu/exemplo.html");

        // Localize o elemento de entrada usando Seletor CSS
        IWebElement inputElement = driver.FindElement(By.CssSelector("#name"));

        // Execute ações no elemento
        inputElement.SendKeys("João Silva");

        // Feche o navegador
        driver.Quit();
    }
}
```

Web Element

CSS Selector

- Acesso Direto: Trabalhar diretamente com o DOM oferece acesso total à estrutura e propriedades do documento
- Integração com **JavaScript**: A manipulação direta do DOM se integra perfeitamente com JavaScript, permitindo mudanças dinâmicas no conteúdo da página
- Atualizações Imediatas: Mudanças no DOM são refletidas imediatamente no navegador

- Verbosidade: A manipulação direta do DOM pode ser mais verbosa e exigir mais código
- Complexidade: Trabalhar diretamente com o DOM pode ser complexo e propenso a erros
- Desempenho: A manipulação direta pode levar a problemas de desempenho se não for tratada de maneira eficiente

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

class DomManipulationExample
{
    static void Main()
    {
        IWebDriver driver = new ChromeDriver();
        driver.Navigate().GoToUrl("file:///caminho/para/seu/exemplo.html");

        // Localize o elemento de entrada usando JavaScript
        IJavaScriptExecutor jsExecutor = (IJavaScriptExecutor)driver;
        IWebElement inputElement = (IWebElement)jsExecutor.ExecuteScript("return document.getElementById('name');");

        // Execute ações no elemento
        inputElement.SendKeys("João Silva");

        // Feche o navegador
        driver.Quit();
    }
}
```

Seletores

Conclusão

- Nem sempre o selector indicado no código exportado pelo Selenium IDE é o mais conveniente
- Em algumas situações, melhor seguir “receitas" de outros testadores:
  - https://stackoverflow.com/questions/tagged/selenium-webdriver?tab=Newest
  - https://www.guru99.com/selenium-tutorial.html
  - https://www.seleniumeasy.com/selenium-tutorials?page=1
  - https://github.com/SeleniumHQ/selenium/tree/trunk/dotnet/test/common
- Cuidado: algumas receitas podem ser incompatíveis com a arquitetura da sua aplicação!

Oito etapas de um teste

Antes da execução do teste

1. Planejamento do Teste
• Defina o escopo e os objetivos do teste
• Identifique os recursos necessários (ferramentas, ambientes, pessoal)
• Desenvolva um plano de teste e cronograma
2. Design do Teste
• Crie casos de teste detalhados com base nos requisitos e especificações
• Defina os dados de teste e os resultados esperados
• Determine a configuração do ambiente de teste
3.Configuração do Ambiente de Teste
• Prepare o ambiente de teste
• Instale e configure o software e hardware necessários
• Certifique-se de que o ambiente de teste corresponda ao ambiente de produção
4. Preparação dos Dados de Teste
• Crie e configure os dados de teste necessários para a execução dos casos de teste
• Garanta que os dados sejam precisos, relevantes e suficientes para o teste

Oito etapas de um teste

Durante a execução do teste

5. Execução do Teste

- Execute os casos de teste conforme o plano de teste
- Registre os resultados dos testes e quaisquer anomalias ou defeitos
- Utilize ferramentas de automação de testes, se aplicável

6. Relatório e Rastreamento de Defeitos

- Registre quaisquer defeitos encontrados durante a execução dos testes
- Acompanhe os defeitos da notificação à resolução
- Comunique-se com os desenvolvedores e partes interessadas

Oito etapas de um teste

Após a execução do teste

7. Relatório de Teste
- Compile os resultados dos testes e forneça um resumo
- Inclua métricas como cobertura, taxas de sucesso e densidade de defeitos
- Destaque problemas ou observações

8. Encerramento do Teste
- Avalie o processo e os resultados do teste
- Garanta que todos os testes planejados foram executados ou adiados
- Arquive os artefatos de teste (casos de teste, scripts, dados)
- Realize uma retrospectiva para melhorar os esforços de teste

Oito etapas de um teste

Após a execução do teste

7. Relatório de Teste
- Compile os resultados dos testes e forneça um resumo
- Inclua métricas como cobertura, taxas de sucesso e densidade de defeitos
- Destaque problemas ou observações

8. Encerramento do Teste
- Avalie o processo e os resultados do teste
- Garanta que todos os testes planejados foram executados ou adiados
- Arquive os artefatos de teste (casos de teste, scripts, dados)
- Realize uma retrospectiva para melhorar os esforços de teste

Oito etapas de um teste

Conclusão

- Para aumentar a qualidade dos testes, organize uma rotina com todas as etapas
  - Combine antes para não ter dúvida na hora de fazer
- O primeiro teste é teste mais difícil do projeto! Não ignore testes e automação de testes!
- Revise a rotina, tome decisões em conjunto, não abandone os testes!
- Descubra quem usa os testes e quem conhece casos de testes!
- Testado manualmente é melhor do que não testado!
- Se não há evidência, não foi feito!

Técnicas de geração de casos de testes

- As técnicas visam diversificar os casos de testes e estimular a geração de casos de testes
- Os resultados esperados devem ser confirmados por um oráculo:
  - Especialista
  - Literatura
  - Sistema equivalente

Teste Exploratório

- O teste exploratório envolve o design e a execução simultâneos de testes. Os testadores exploram a aplicação e desenham testes com base em suas descobertas.

- Exemplo:
- Ao testar um novo site de e-commerce, o testador pode:

- Navegar por diferentes categorias.
- Adicionar itens ao carrinho.
- Aplicar vários filtros e opções de classificação.
- Tentar finalizar a compra com diferentes métodos de pagamento.
- Procurar por comportamentos inesperados, erros ou inconsistências.

Partição de Equivalência

- O particionamento de equivalência envolve dividir os dados de entrada em partições de dados equivalentes a partir das quais os casos de teste podem ser derivados. A ideia é que se uma condição em uma partição funcionar, todas as outras condições nessa partição também funcionarão.

- Exemplo:

Suponha que você tenha um campo de texto que aceita uma entrada de idade variando de 1 a 100.

- Partições:

- Entrada válida: 1 a 100
- Entrada inválida: Menor que 1 (por exemplo, 0, -1)
- Entrada inválida: Maior que 100 (por exemplo, 101, 150)

- Casos de Teste:

- Testar com um valor dentro do intervalo válido: 25 (deve passar)
- Testar com um valor abaixo do intervalo válido: 0 (deve falhar)
- Testar com um valor acima do intervalo válido: 150 (deve falhar)

Análise de Valor Limite

- A análise de valor limite foca nos limites entre as partições. É frequentemente usada junto com o particionamento de equivalência.

- Exemplo:

Para o mesmo campo de entrada de idade (1 a 100), os limites são 1 e 100.

- Casos de Teste:

- Testar com o valor mínimo do limite: 1 (deve passar)
- Testar logo abaixo do limite mínimo: 0 (deve falhar)
- Testar logo acima do limite mínimo: 2 (deve passar)
- Testar com o valor máximo do limite: 100 (deve passar)
- Testar logo abaixo do limite máximo: 99 (deve passar)
- Testar logo acima do limite máximo: 101 (deve falhar)

Adivinhação de Erros

- A adivinhação de erros envolve usar a experiência para adivinhar as áreas problemáticas da aplicação onde os defeitos provavelmente ocorrerão.

- Exemplo:

Considere um formulário de login com campos de nome de usuário e senha.

- Casos de Teste:

- Testar com campos de nome de usuário e senha vazios.
- Testar com um nome de usuário muito longo.
- Testar com caracteres especiais no nome de usuário.
- Testar com tentativas de SQL injection

  ex.: username = ' OR ‘1’='1'

Teste de Tabela de Decisão

- O teste de tabela de decisão é usado para sistemas onde a saída depende de combinações de entradas. Envolve criar uma tabela de decisão para representar diferentes condições de entrada e suas ações correspondentes.

- Exemplo:

Considere um calculador de prêmio de seguro que possui diferentes taxas baseadas na idade e no histórico de direção.

Grupo de Idade	Histórico de Direção	Taxa do Prêmio
<25	Bom	Alta
<25	Ruim	Muito Alta
≥25	Bom	Média
≥25	Ruim	Alta

- Casos de Teste:

- Idade < 25, Histórico de Direção Bom: Alta
- Idade < 25, Histórico de Direção Ruim: Muito Alta
- Idade ≥ 25, Histórico de Direção Bom: Média
- Idade ≥ 25, Histórico de Direção Ruim: Alta
-

Teste de Transição de Estado

- O teste de transição de estado é usado quando o sistema muda seu comportamento com base no seu estado atual e em um evento ou entrada. É útil para testar máquinas de estado.

- Exemplo:

Considere uma catraca simples que tem dois estados: Travada e Destravada.

Estado | Evento | Próximo Estado | Ação
Travada | Inserir Moeda | Destravada | Destravar Catraca
Travada | Empurrar | Travada | Nenhuma Ação
Destravada| Empurrar | Travada | Travar Catraca
Destravada| Inserir Moeda | Destravada | Nenhuma Ação

- Casos de Teste:

- No estado Travada, inserir uma moeda: Deve mudar para o estado Destravada.
- No estado Travada, empurrar a catraca: Deve permanecer no estado Travada.
- No estado Destravada, empurrar a catraca: Deve mudar para o estado Travada.
- No estado Destravada, inserir uma moeda: Deve permanecer no estado Destravada.
-

Teste de História de Usuário

- O teste de história de usuário envolve criar casos de teste com base em histórias de usuários, que são descrições curtas e simples de um recurso contadas da perspectiva do usuário final. As histórias de usuário capturam o "quem", "o quê" e "por quê" de um recurso, ajudando a garantir que o software atenda às necessidades do usuário.

- Exemplo de História de Usuário:

Como usuário registrado, quero redefinir minha senha para que eu possa recuperar o acesso à minha conta se eu esquecer minha senha.

- Critérios de Aceitação:

1. O usuário deve receber um email de redefinição de senha ao solicitar a redefinição de senha.
2. O email deve conter um link único para uma página de redefinição de senha.
3. O usuário deve poder inserir uma nova senha na página de redefinição.
4. A nova senha deve atender aos requisitos de segurança definidos (por exemplo, comprimento mínimo, pelo menos um caractere especial).
5. O usuário deve ser notificado da redefinição de senha bem-sucedida.

Teste de História de Usuário

- Caso de Teste 1: Solicitar Email de Redefinição de Senha

Pré-condição: O usuário está na página de login e clica no link "Esqueci minha senha".

Passos:

1. Inserir o endereço de email registrado.

2. Clicar no botão "Enviar".

Resultado Esperado:

- O usuário recebe um email de redefinição de senha contendo um link único para a página de redefinição de senha.

Teste de História de Usuário

- Caso de Teste 2: Conteúdo do Email de Redefinição de Senha

Pré-condição: O usuário solicitou uma redefinição de senha.

Passos:

1. Abrir o email de redefinição de senha.

Resultado Esperado:

- O email contém um link único para a página de redefinição de senha.

Teste de História de Usuário

- Caso de Teste 3: Inserir Nova Senha

Pré-condição: O usuário clicou no link no email de redefinição de senha e está na página de redefinição de senha.

Passos:

1. Inserir uma nova senha que atenda aos requisitos de segurança.

2. Confirmar a nova senha.

3. Clicar no botão "Redefinir Senha".

Resultado Esperado:

- O usuário é notificado de que a redefinição de senha foi bem-sucedida.

- O usuário pode fazer login com a nova senha.

Diagramas

- Linguagem visual para sistematizar informações
- Baixo custo
- Exercitam comunicação e reflexão
- Mapas Mentais, Mapas Conceituais, Protótipos de Telas, Arquitetura de Software, Processos…

https://plantuml.com/stdlib

Conclusão

- Acrescente complexidade a medida que os ciclos se repetem
- Utilize receitas e auxílio de ferramentas de captura e reprodução (ex.: Selenium IDE) para obter um esboço de automação
- Mantenha uma arquitetura uniforme
- Mantenha um padrão para cada atividade e crie atividades padronizadas para reduzir dúvidas e ganhar produtividade
- A prioridade é melhorar a qualidade, todo o restante pode ser feito nos próximos ciclos
- Coloque um teste em cada história e evolua quando for possível