Robox em Java: Desenvolvimento de Automação

O desenvolvimento de robótica em Java (*Robox*) e sua aplicação em automação de processos, como o RPA (Automação Robótica de Processos), são áreas que estão se expandindo rapidamente.

Este artigo explora a aplicação de Java para automação de tarefas tanto em ambientes web quanto em software de desktop, mostrando como essas técnicas podem ser usadas para melhorar a produtividade em processos repetitivos.

Abordaremos exemplos práticos para RPA em Java, usando diferentes níveis de complexidade, desde tarefas simples até fluxos empresariais avançados. Para isso, utilizaremos versões LTS do Java, como Java 11 e 17.

Além disso, expandiremos a automação para incluir exemplos que utilizam o Java para mapear interações com softwares desktop, aumentando o escopo da automação além do ambiente web.

Robox em RPA – Nível Básico

Começaremos com um exemplo simples de automação em ambiente web, usando Java e a biblioteca Selenium, para simular o login em um site.

Exemplo: Automação de Login em um Website

Versão do Java Utilizada: Java 11 LTS

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.By;

public class AutomacaoLoginBasico {

    public static void main(String[] args) {
        // Configura o caminho do driver do Chrome
        System.setProperty("webdriver.chrome.driver", "/caminho/para/chromedriver");

        // Inicializa o navegador Chrome
        WebDriver driver = new ChromeDriver();

        // Abre a página de login
        driver.get("https://exemplo.com/login");

        // Preenche o campo de login
```

```
        driver.findElement(By.id("usuario")).sendKeys("meuUsuario");

        // Preenche o campo de senha
        driver.findElement(By.id("senha")).sendKeys("minhaSenha");

        // Simula o clique no botão de login
        driver.findElement(By.id("botaoLogin")).click();

        // Finaliza a sessão
        driver.quit();
    }
}
```

- **Processo:** Automatiza uma simples interação com uma página de login usando Selenium, uma biblioteca amplamente utilizada para automação de navegação em navegadores.
- **Aplicação:** Tarefas repetitivas como preenchimento de formulários podem ser automatizadas com facilidade.

Robox em RPA – Nível Intermediário

Agora, expandimos a automação para a coleta de dados de diferentes páginas da web, salvando os resultados em um arquivo local. Este exemplo simula um cenário de *web scraping*.

Exemplo: Extração de Dados de Vários Sites

Versão do Java Utilizada: Java 11 LTS

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.By;
import java.io.FileWriter;
import java.io.IOException;

public class AutomacaoExtracaoDadosIntermediario {

    public static void main(String[] args) throws IOException {
        // Configura o driver do Chrome
        System.setProperty("webdriver.chrome.driver", "/caminho/para/chromedriver");
        WebDriver driver = new ChromeDriver();

        // Array de URLs para extrair dados
        String[] urls = {"https://exemplo.com/page1", "https://exemplo.com/page2",
"https://exemplo.com/page3"};

        // Abre arquivo para salvar os dados
        FileWriter writer = new FileWriter("dadosExtraidos.txt");

        // Loop através das URLs para extrair dados
        for (String url : urls) {
            driver.get(url);

            // Extrai o texto de um elemento da página
            String dados = driver.findElement(By.id("dados")).getText();
```

```
            // Escreve os dados no arquivo
            writer.write("Dados extraídos de " + url + ":\n" + dados + "\n\n");
        }

        // Fecha o arquivo e o navegador
        writer.close();
        driver.quit();
    }
}
```

- **Processo:** Realiza uma navegação por várias URLs, extraindo dados e salvando-os em um arquivo local, útil em operações de scraping e coleta de informações.
- **Aplicação:** Automatização de tarefas de coleta de dados de múltiplas fontes.

Robox em RPA – Nível Avançado

Este exemplo simula um fluxo de trabalho mais complexo em uma aplicação empresarial.

O robô fará login, extrairá dados e tomará decisões com base nos dados extraídos.

Exemplo: Automação de Fluxo de Trabalho Empresarial com Tomada de Decisão

Versão do Java Utilizada: Java 17 LTS

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.By;
import java.util.concurrent.TimeUnit;

public class AutomacaoFluxoAvancado {

    public static void main(String[] args) {
        // Configura o driver do Chrome
        System.setProperty("webdriver.chrome.driver", "/caminho/para/chromedriver");
        WebDriver driver = new ChromeDriver();

        // Timeout para carregar as páginas
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

        try {
            // Login em sistema empresarial
            driver.get("https://empresa.com/login");
            driver.findElement(By.id("usuario")).sendKeys("usuarioEmpresa");
            driver.findElement(By.id("senha")).sendKeys("senhaSegura");
            driver.findElement(By.id("botaoLogin")).click();

            // Extração de dados de um relatório após o login
            driver.get("https://empresa.com/relatorio");
            String relatorio = driver.findElement(By.id("dadosRelatorio")).getText();

            // Tomada de decisão com base nos dados extraídos
```

```
            if (relatorio.contains("Aprovado")) {
                System.out.println("Processo aprovado. Enviando notificação.");
                driver.get("https://empresa.com/enviarNotificacao");
                driver.findElement(By.id("mensagem")).sendKeys("Processo aprovado.");
                driver.findElement(By.id("enviar")).click();
            } else {
                System.out.println("Processo não aprovado. Aguardando revisão.");
            }
        } finally {
            // Finaliza o processo e fecha o navegador
            driver.quit();
        }
    }
}
```

- **Processo:** Este código simula um fluxo de trabalho empresarial, com base na extração de dados e na tomada de decisão automática.
- **Aplicação:** Usado em contextos empresariais para automatizar workflows, como a aprovação de processos e a comunicação automática com sistemas.

Robox em Automação de Software Desktop

Em muitos cenários, a automação de software desktop pode ser necessária, como ao interagir com aplicativos que não têm uma interface web.

O Java, em conjunto com bibliotecas como FlaUI (usada com a ponte Java-C#), pode ser usado para automatizar interações com software de desktop.

Exemplo: Automação de Software Desktop

Usaremos o JNativeHook, uma biblioteca em Java que permite capturar eventos do sistema operacional para mapear e automatizar interações com software desktop.

Versão do Java Utilizada: Java 17 LTS

```
import org.jnativehook.GlobalScreen;
import org.jnativehook.keyboard.NativeKeyEvent;
import org.jnativehook.keyboard.NativeKeyListener;

public class AutomacaoDesktop implements NativeKeyListener {

    public static void main(String[] args) {
        try {
            // Registra o hook global para capturar eventos do teclado
            GlobalScreen.registerNativeHook();
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Adiciona o listener de teclas
        GlobalScreen.addNativeKeyListener(new AutomacaoDesktop());
    }
```

```
    @Override
    public void nativeKeyPressed(NativeKeyEvent e) {
        // Captura quando uma tecla é pressionada
        if (e.getKeyCode() == NativeKeyEvent.VC_ENTER) {
            System.out.println("Enter pressionado! Executando ação.");
            // Aqui você pode adicionar a automação de interação com software desktop
            executarAcaoAutomatica();
        }
    }

    @Override
    public void nativeKeyReleased(NativeKeyEvent e) {}

    @Override
    public void nativeKeyTyped(NativeKeyEvent e) {}

    public static void executarAcaoAutomatica() {
        // Exemplo de interação com software de desktop
        System.out.println("Interagindo com software desktop...");
        // Implementar comandos específicos de interação
    }
}
```

- **Processo:** O JNativeHook permite capturar eventos de teclado e mouse, possibilitando a automação de softwares desktop que exigem interações específicas.
- **Aplicação:** Este exemplo pode ser estendido para automatizar o controle de janelas, teclas de atalho e outros elementos de software desktop.

Combinando Java com bibliotecas poderosas, como Selenium para automação web e JNativeHook para automação de software desktop, podemos implementar soluções robustas para automação de processos repetitivos em diversos ambientes.

A capacidade de mapear interações tanto em web quanto em software desktop torna o Java uma escolha ideal para implementar soluções de RPA e Robox em uma ampla gama de aplicações, desde tarefas simples até fluxos de trabalho empresariais complexos.

Dependências Utilizadas

Selenium WebDriver

O Selenium WebDriver é uma biblioteca essencial para automação de navegadores. Ela permite simular interações com páginas web, como clicar em botões e preencher formulários.

- **Documentação Oficial**: https://www.selenium.dev/documentation/
- **Download do WebDriver para Chrome**: https://sites.google.com/a/chromium.org/chromedriver/downloads

- **Dependência Maven**:

xml
```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.4.0</version>
</dependency>
```

JNativeHook

O JNativeHook é uma biblioteca em Java que permite capturar eventos de sistema como interações de teclado e mouse, essencial para automação de software desktop.

- **Documentação Oficial**: https://github.com/kwhat/jnativehook
- **Dependência Maven**:

xml
```
<dependency>
    <groupId>com.github.kwhat</groupId>
    <artifactId>jnativehook</artifactId>
    <version>2.2.1</version>
</dependency>
```

FlaUI (para automação de software desktop via Java-C#)

O FlaUI é uma biblioteca popular para automação de aplicativos desktop baseados em Windows. Embora seja nativa de C#, pode ser usada em Java com integração via ponte JNI.

- **Documentação Oficial**: https://github.com/FlaUI/FlaUI
- **Download da Ponte Java-C#**: https://github.com/java-native-access/jna

JavaFX (opcional para automação de interfaces desktop customizadas)

Caso seja necessário criar ou interagir com GUIs desktop personalizadas, o JavaFX é uma excelente ferramenta.

- **Documentação Oficial**: https://openjfx.io/
- **Download do JavaFX SDK**: https://gluonhq.com/products/javafx/
- **Dependência Maven**:

xml
```
<dependency>
    <groupId>org.openjfx</groupId>
    <artifactId>javafx-controls</artifactId>
    <version>17.0.2</version>
</dependency>
```

Arquivos JAR

Para aqueles que preferem incluir os arquivos JAR diretamente ao invés de usar o Maven, aqui estão os links para download:

1. **Selenium JAR**:
   - Download: https://www.selenium.dev/downloads/
2. **JNativeHook JAR**:
   - Download: https://mvnrepository.com/artifact/com.github.kwhat/jnativehook
3. **FlaUI**:
   - Download: https://github.com/FlaUI/FlaUI/releases
4. **JavaFX SDK**:
   - Download: https://gluonhq.com/products/javafx/

Instalação e Configuração

1. Maven: Se estiver utilizando Maven para gerenciar dependências, basta adicionar as dependências listadas no arquivo pom.xml do seu projeto. Após isso, o Maven fará o download automaticamente das bibliotecas necessárias.

2. Manualmente: Para adicionar os arquivos JAR manualmente, faça o download dos arquivos pelos links fornecidos e adicione-os ao classpath do seu projeto no IDE que estiver utilizando (Eclipse, IntelliJ, etc.).

Essas referências e links cobrem tudo que você precisa para rodar os exemplos apresentados neste artigo. Dessa forma, tanto a automação web quanto a automação de softwares desktop podem ser implementadas com as ferramentas adequadas, integradas ao ambiente Java.

EducaCiência FastCode para comunidade