



Automação de Testes Robot Framework

Prof Msc Randerson Melville



+

•

○

O que é o Robot Framework?

- Um framework de automação de testes open-source.
- Baseado em palavras-chave (keyword-driven testing).
- Suporta integração com várias bibliotecas, como Selenium para automação de testes de interface gráfica (UI).

+

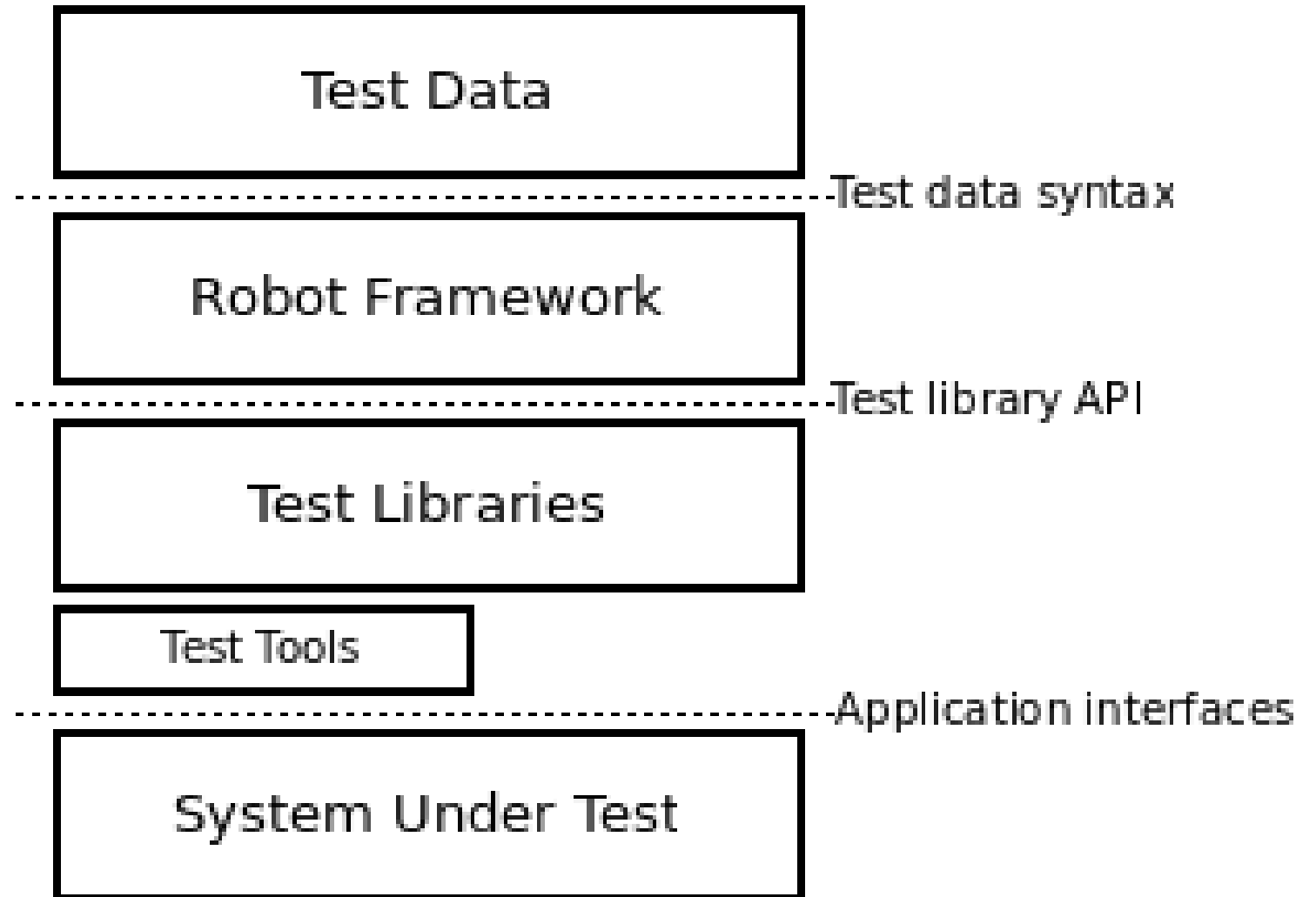
•

○

Benefícios do Robot Framework:

- Simplicidade na escrita de testes.
- Extensível com bibliotecas customizadas.
- Independente da linguagem de programação, pode ser usado com Python, Java, etc.

Arquitetura do Robot Framework



+

•

○

Estrutura dos Testes no Robot Framework

- **Formato dos Arquivos de Teste:**
 - Arquivos de teste em formato .robot ou .txt.
- **Seções principais dos arquivos:**
 - **Settings:** Configurações globais, como importação de bibliotecas.
 - **Variables:** Definição de variáveis reutilizáveis.
 - **Test Cases:** Onde os testes são definidos usando palavras-chave.
 - **Keywords:** Criação de palavras-chave customizadas.
- **Sintaxe básica:**
 - Uso de espaços e tabulação.
 - Palavras-chave e argumentos.

+

•


○

Bibliotecas Comuns do Robot Framework

- **BuiltIn Library:**
 - Biblioteca padrão com palavras-chave úteis (e.g., Log, Run Keyword If).
- **SeleniumLibrary:**
 - Automação de testes de navegadores.
- **OperatingSystem Library:**
 - Execução de comandos do sistema operacional.
- **Outras bibliotecas:** Collections, String, e HTTP.

Site Robot

<https://robotframework.org/>

ROBOT
FRAME
WORK 

Robot Framework is an open source automation framework for test automation and [robotic process automation \(RPA\)](#). It is supported by the [Robot Framework Foundation](#) and [widely used](#) in the industry.

Its [human-friendly and versatile syntax](#) uses keywords and supports [extending through libraries](#) in Python, Java, and other languages.

It integrates with other tools for comprehensive automation without licensing fees, bolstered by a rich community with hundreds of [3rd party libraries](#).

Prática



Instalação e Configuração

- **Passo 1: Instalar Python e pip**
 - Verifique se o **Python** está instalado executando `python --version` no terminal.
 - Se o Python não estiver instalado, baixe e instale a versão mais recente do [site oficial do Python](#).
 - O `pip` geralmente vem com o Python. Confirme a instalação executando `pip --version`.

Instalação e Configuração

- **Passo 2: Instalar o Robot Framework**

- Com o Python configurado, abra o terminal no **VSCode**.

- Execute o comando:

```
bash
```

```
pip install robotframework
```

- **Passo 3: Instalar o SeleniumLibrary**

- No mesmo terminal, instale a biblioteca **SeleniumLibrary**:

```
bash
```

```
pip install robotframework-seleniumlibrary
```

Instalação e Configuração

- **Passo 4: Instalar o WebDriver (ChromeDriver)**
 - Para automação com o navegador Chrome, baixe o **ChromeDriver** compatível com a versão do Chrome no site oficial do ChromeDriver.
 - Extraia o arquivo e mova o executável para uma pasta acessível pelo sistema (e.g., /usr/local/bin no macOS/Linux ou adicione ao PATH no Windows).
- **Passo 5: Configurar o VSCode**
 - No **VSCode**, instale a extensão **Robot Framework Language Server** para facilitar a criação e execução de testes no editor.
 - Vá em **Extensões** e pesquise por "Robot Framework Language Server".
 - Clique em **Instalar**.
 - Crie uma pasta de projeto no VSCode onde os arquivos de teste serão armazenados.

Primeiro Teste

- Agora vamos criar um script de teste para automatizar a pesquisa de um produto no site de exemplo **Amazon**.
- **Passo 1: Criar o arquivo de teste**
 - No VSCode, crie um novo arquivo chamado `teste_pesquisa_amazon.robot` dentro da pasta do projeto.
- **Passo 2: Estruturar o arquivo de teste**
 - Adicione o código abaixo, que vai automatizar a pesquisa de um produto no site da Amazon:

Primeiro Teste

robot

*** Settings ***

Library SeleniumLibrary

*** Variables ***

\${URL} https://www.amazon.com.br

\${BROWSER} chrome

\${PRODUTO} notebook

*** Test Cases ***

Pesquisa de Produto na Amazon

Open Browser \${URL} \${BROWSER}

Input Text id=twotabsearchtextbox \${PRODUTO}

Click Button id=nav-search-submit-button

Page Should Contain \${PRODUTO}

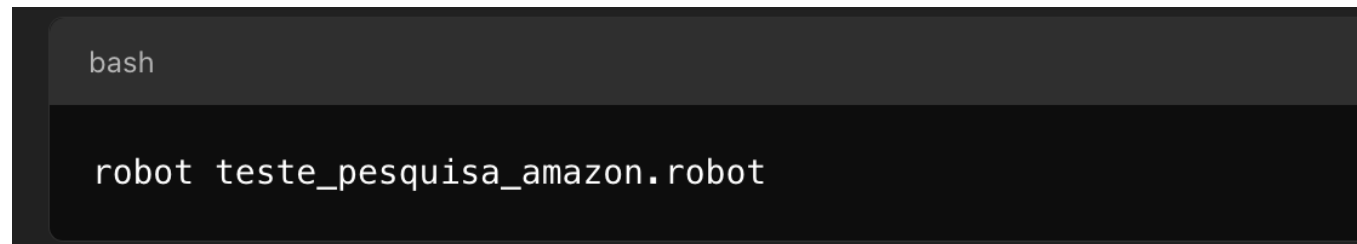
Close Browser

Primeiro Teste

- **Settings:** Importa a biblioteca SeleniumLibrary, que permite interagir com navegadores.
- **Variables:** Define variáveis como a URL do site, o nome do produto a ser pesquisado e o navegador utilizado.
- **Test Cases:** Descreve o caso de teste que:
 - Abre o navegador Chrome e acessa a Amazon.
 - Insere o nome do produto no campo de busca.
 - Clica no botão de busca.
 - Verifica se o nome do produto aparece na página de resultados.
 - Fecha o navegador.

Executar o teste

- No terminal do VSCode, execute o seguinte comando para rodar o teste:

A screenshot of a terminal window with a dark background. The prompt 'bash' is visible on the first line. The command 'robot teste_pesquisa_amazon.robot' is entered on the second line.

```
bash
robot teste_pesquisa_amazon.robot
```

- Isso abrirá o navegador Chrome, realizará a busca e gerará um relatório HTML com os resultados do teste.