bash-api-test

HTML: https://paulojeronimo.com/posts/bash-api-test

1. Introdução

O Bash se torna super poderoso para testes de APIs quando aliado a ferramentas como curl, httpie e jq. Ao escrever código com essa *stack*, você verá que ele é sucinto e eficiente para trabalhar com requisições e respostas na chamada a APIs. Conheça o bash-api-test, um *framework* que te ajudará nisso.

Frameworks para teste de APIs são numerosos nas linguagens de programação mais apropriadas para este finalidade. Eu confesso que, Para programadores JavaScript (ou Typescript), alguns *frameworks* ainda podem ser mais simples de serem usados do que esse que estou desenvolvendo. Porém, em se tratando de opções para teste de APIs via linha de comando, e Para quem gosta de Shell, essas ferramentas são incríveis!

Quando surgiu-me a ideia de desenvolver um *framework* que automatiza a execução de testes de APIs, o que eu queria mesmo era responder a pergunta: **será que eu consigo aprimorar meu conhecimento em Bash enquanto desenvolvo scripts mais simples criar testes usando essas ferramentas que gosto?** Nessa época faltava-me conhecimento em uma peça fundamental para escrever esse *framework*: o jq. Mas, desenvolvi minhas habilidades nele e, hoje, isso tem sido peça chave na construção do bash-api-test.

1.1. Minhas motivações na escrita deste framework

1. **Testes leves e rápidos**.
    a. Até mesmo utilizando o Termux em meu celular.

1. **Aprendizado**.
   Eu posso te garantir que, mesmo desenvolvendo soluções em Bash há vários anos (desde 2007), todo vez que eu crio algo novo com ele eu aprendo, ainda, muito mais. Portanto, meu objetivo na construção desse framework é meu aperfeiçoamento tanto no Bash quanto nessas ferramentas Além disso, eu também espero conseguir te ajudar a entender um pouco mais sobre elas.

1.2. O surgimento da ideia

A ideia de criação deste framework surgiu em Junho de 2020 quando eu criei um exemplo de como utilizar essas ferramentas na utilização de testes automatizados. Como escrevi, nesse tempo eu não dominava o jq. Mas, comecei a trabalhar bastante com o ele para fazer testes de integração de APIs com o Keycloak, uma ferramenta que já me deu muito trabalho (no bom sentido dessa expressão).

Em fevereiro de 2021, trabalhei em um projeto na OSO DevOps no qual aprimorei bastante meu conhecimento no jq. Para expressar um pouco desse conhecimento de forma pública, e me aprimorar um pouco mais, também desenvolvi alguns brinquedos.

Hoje eu sei o enorme poder que a junção dessas ferramentas oferece e, dessa forma, acho importante compartilhar esse aprendizado.

2. Tutorial: como testar uma API com o bash-api-test?

Neste primeiro tutorial (de uma série) eu te mostrarei como testar uma API utilizando o Bash para excutar o bash-api-test de duas formas diferentes. A primeira será no seu próprio celular, se ele for um Android. Você fará isso No Termux.

Outra forma que você poderá executar o bash-api-test, neste tutorial, será Via Docker. Em Próximos Tutoriais eu te explicarei, em detalhes, como escrever seus próprios testes para que eles sejam executados pelo bash-api-test.

2.1. No Termux

Copie e cole os comandos abaixo em seu shell. Se tiver dúvidas sobre como instalar o Termux ou sobre como executar os comandos abaixo dentro dele, veja esses vídeos.

```
source <(curl -sSL https://raw.githubusercontent.com/paulojeronimo/bash-api-test/main/termux-setup.sh) v0.5.0
```

Inicie o script `api-server.sh` em uma nova sessão tmux com esse comando:

```
./api-server.sh start --with-tmux
```

Esse comando abrirá o tmux com dois (2) paineis. O de baixo executará o JSON Server e uma API que testaremos.

No painel de cima você testará essa API através do seguinte comando:

```
./sample-tests.runner.sh
```

2.1.1. Vídeos

1. Instalação do Termux no Android. (← TODO)
2. Configuração e execução do bash-api-test no Termux. (← TODO)

2.2. Via Docker

Você pode testar o bash-api-test se tiver o Docker instalado em sua máquina. Particularmente eu, até o momento, só realizei testes utilizando o Docker que tenho instalado em um sistema operacional Ubuntu e em um macOS. Para utilizar o Windows, você deverá ter o WSL 2 instalado executando um Linux (além do Docker) e executar os procedimentos a seguir.

Até mesmo para iniciar este projeto via Docker eu utilizo um script que desenvolvi, em Bash, e que chamei de docker-termux.

O procedimento abaixo é ligeiramente diferente do que eu descrevo no README.adoc do projeto (em inglês) e que está publicado também (em HTML) em https://paulojeronimo.com/bash-api-test/#through-docker.

Na prática, a forma que eu apresento a execução do projeto a seguir é interessante pois ela não te obriga a ter nenhuma outra ferramenta, além de um Bash (na versão igual ou superior a que vem instalada por padrão no macOS) e do Docker. Porém, quem já possui um ambiente de desenvolvimento instalado é beneficiado com o procedimento que descrevi no README.adoc do projeto bash-api-test.

De qualquer forma, vamos lá:

Baixe o script docker-termux em sua máquina e coloque-o em execução (ele criará um contêiner Docker que utilizo para testar, localmente em meu notebook e fora de um Android, minhas aplicações no Termux):

```
$ curl -sSL
https://raw.githubusercontent.com/paulojeronimo/dotfiles/master/.scripts/docker/docker
-termux
-o docker-termux

$ ./docker-termux
```

Após obter acesso ao shell do contêiner Docker você pode testar os mesmos passos que descrevo para a execução No Termux.

2.2.1. Vídeos

1. Desenvolvimento Mobile NÃO CONVENCIONAL utilizando o Termux.
2. Executando o bash-api-test via docker-termux. (← TODO)

3. Frameworks para teste de APIs

3.1. Para programadores JavaScript

Jest e SuperTest

Se você programa em JavaScript (ou TypeScript) e deseja utilizar o Jest unido ao SuperTest, essa é uma ótima opção!

Com eles você terá ainda mais facilidade e rapidez na codificação de testes para APIs.

3.2. Para quem gosta de Shell

- https://stackoverflow.com/questions/29042593/rest-api-testing-from-commandline
  - https://github.com/melezhik/swat
  - https://github.com/subeshb1/api-test

4. Próximos Tutoriais

O objetivo dessa série de tutoriais é ajudá-lo a conhecer um pouco mais sobre a junção que cola todas as ferramentas no framework bash-api-test (o Bash) além de, obviamente, melhorar seus *skills* em todas elas.

Caso deseje ser notificado da criação ou da atualiação desse conteúdo, por favor, assine meu canal no Telegram: https://t.me/paulojeronimo_com.