Configurando o ambiente para executar o Appium

Oscar Tanner Follow
Oct 4, 2017 · 8 min read

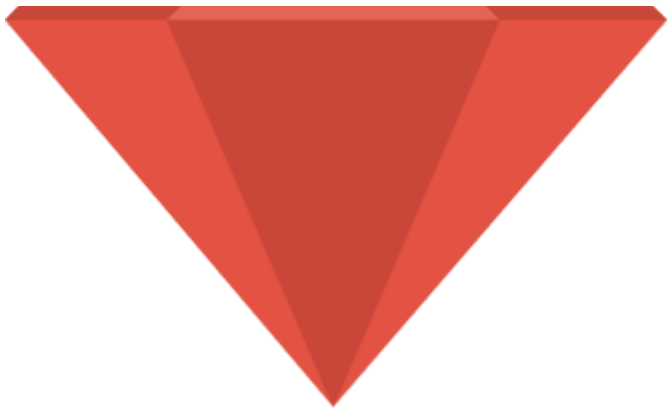
Olá! Nesse post vou ensinar como configurar uma máquina para utilizar o Appium com um cliente Ruby. Esse tutorial foi desenvolvido para abordar a instalação nos SOs Windows, Linux e MacOS.

Para os usuários Windows, uma dica legal é instalar o *Cmder* (http://cmder.net/). Os passos são:

1. Baixar a versão full;
2. Descompactar (pode demorar um pouco);
3. (Opcional) Incluir os executáveis que desejar na pasta *bin* para que eles sejam incluídos automaticamente na variável PATH;
4. Executar o *Cmder.exe*.

Mais informações sobre a instalação, atalhos e muito mais, podem ser encontradas nesse link do Github https://github.com/cmderdev/cmder e nesse link do youtube https://www.youtube.com/watch?v=2YCttatKtuU.

Vamos começar instalando o Ruby. Para usuários Windows, o **Papito** fez um excelente Post (https://medium.com/@papito/instalando-ruby-cucumber-e-capybara-no-windows-10-acb1fe833a95) explicando certinho como fazer a instalação.



Para usuários de Linux e MacOS eu sempre recomendo a instalação do Ruby usando um Gerenciador de Versões, como *RVM* ou *rbenv*. Esses programas são responsáveis por permitir que você possua diversas versões de Ruby instaladas ao mesmo tempo, possibilitando que você consiga definir para quais pastas do sistema (projeto) você deseja utilizar cada versão. Com isso, o processo de atualização da linguagem é facilitado, permitindo que você sempre tenha a versão mais atualizada de Ruby em sua máquina.

> *No MacOS o Ruby já vem instalado e no Linux possuímos o Gerenciador de pacotes. Não recomento utilizar essas versões de Ruby, pois normalmente não são as mais atualizadas.*

Não existem grandes diferenças entre as funcionalidades fornecidas pelo RVM e pelo rbenv, somente algumas particularidades. Se você procurar no google por “RVM x rbenv” vai descobrir que existem muitos posts sobre o assunto.

> *Já utilizei RVM por muitos anos e atualmente uso o rbenv. Por que mudei? Por pura curiosidade e porque o rbenv pode ser instalado no MacOS usando o homebrew, então é um pouco mais fácil do que o RVM. Já no Linux, o rbenv deve ser instalado por meio de Github Checkout, então o RVM é uma opção mais fácil.*

Para instalar o RVM, siga as instruções presentes em http://rvm.io/rvm/install. Basicamente você deverá instalar uma chave GPG e o RVM stable com o Ruby. No caso do Ubuntu, você poderá inclusive instalar o RVM utilizando o gerenciador de pacotes.

Para instalar o rbenv, siga as instruções em https://github.com/rbenv/rbenv#installation ou, se for usuário MacOS, os passos descritos abaixo.

Primeiro será necessário ter o Homebrew instalado. Caso não tenha, siga as instruções em https://brew.sh/.

> *O Ruby possui um console interativo que interpreta e executa comandos. Para poder utilizar caracteres especiais sem problemas nesse console, é necessário instalar também o pacote readline.*

Os comandos necessários para instalar o rbenv no MacOS são:

```
$ brew install readline
$ brew install rbenv
$ rbenv init     ## seguir as instruções fornecidas
$ rbenv versions ## para escolher qual versão de Ruby você deseja
instalar. Atualmente é a versão mais recente é a 2.4.2
$ RUBY_CONFIGURE_OPTS=--with-readline-dir="$(brew --prefix
readline)" rbenv install 2.4.2  ## substituir pelo número da versão
mais recente
rbenv global 2.4.2 ## substituir pela versão mais recente
```

Tanto para o RVM quanto para o rbenv, ao final, garanta:

1. O *.bash_profile* (ou arquivo referente, como *.bashrc* ou *.zrhrc*) foi alterado com as linhas necessárias para executar o gerenciador de versões sempre que uma sessão for iniciada no terminal;
2. Sua sessão no terminal foi reiniciada, inicializando assim o gerenciador de versão;
3. Uma versão global do Ruby foi selecionada.

Depois de instalado o Ruby, devemos instalar as ferramentas necessárias para executarmos os aplicativos Android e iOS em nossas máquinas. Lembrando que somente usuários MacOS podem executar aplicativos iOS.

Para Android, o **Rafael Toledo** fez um post muito bom que descreve o passo a passo da instalação do Android Studio no Windows, Linux e MacOS.
http://www.rafaeltoledo.net/configurando-o-ambiente-de-desenvolvimento-tutorial-android-1/

Em passos gerais, o que deve ser feito é:

1. Instalar a última versão do Java JDK;
2. Instalar o Android Studio;
3. Configurar as variáveis de ambiente.

É importante também instalar um emulador para executar os aplicativos, assim você não precisa ter um aparelho de celular físico. Você pode utilizar os próprios emuladores do SDK do Android se preferir, mas eles exigem um trabalho a mais para configurar tudo certinho e deixá-los funcionando da melhor forma possível. Por isso, eu sempre recomendo o Genymotion. Ele é pago, mas existe uma versão grátis de uso pessoal. O legal é que a instalação dele é super fácil (bom e velho next, next, next…) e a única dependência é que você tenha o Oracle Virtual Box instalado.

Como o Genymotion é pago ou free na versão pessoal, muitas empresas não permitem a sua utilização. Se esse for o seu caso, ou se preferir, você poderá utilizar os emuladores do próprio Android SDK.

Para isso você precisará abrir o Android Studio e, caso não tenha um projeto Android, criar um projeto vazio (next, next, next…). Sim, isso realmente é necessário :(

Com o projeto criado e aberto no Android Studio, você irá visualizar o ícone do AVD Manager (Gerenciador de AVD ou Android Virtual Device)

Ícone do AVD Manager

Ao clicar nesse ícone você verá a opção `+ Create Virtual Device`. Os passos daqui para a frente são bem simples:

1. Selecionar ou criar o tipo de emulador que você deseja;
2. Selecionar qual versão do Android será instalada no emulador (a primeira vez que você utilizar uma versão ela precisará ser baixada)
3. Nesse passo, eu recomendo clicar no botão `Show Advanced Settings` e selecionar `Cold boot` como valor para a opção `Boot option`. Normalmente, para limpar a memória dos meus emuladores, eu costumo reiniciá-los. Se você deixar a opção padrão selecionada (`Quick boot`), toda vez que você fechar seu emulador, o estado atual será salvo e carregado na próxima vez que você abrir esse emulador;
4. Clicar em `Finish`.

Se for um usuário de MacOS, instale o Xcode para poder executar aplicativos iOS. A instalação é bem simples e deve ser feita pela app Store. Depois de instalar o Xcode,

execute-o para aceitar seus termos de uso. Somente depois disso as funcionalidades ficarão totalmente disponíveis.

E, finalmente, vamos instalar o **Appium!**

Usuários Windows, Linux e MacOS, poderão instalar o Appium Desktop, que é um wrapper UI para o servidor do Appium. Atualmente a última versão é a *1.2.3* e pode ser baixada em https://github.com/appium/appium-desktop/releases/tag/v1.2.3

> *Apesar do Appium Desktop ser mais fácil de instalar, minha preferência pessoal é instalar diretamente pelo terminal. Isso também pode ser feito para qualquer um dos 3 sistemas operacionais mencionados.*

Para MacOS é bem simples, você precisará executar somente dois comandos:

```
$ brew install node
$ npm install -g appium
```

Para Linux, a instalação pode ser feita de forma análoga, visto que, assim como existe o Homebrew para MacOS, existe o LinuxBrew para Linux (http://linuxbrew.sh/).

Para Windows, ou se não quiser usar o Linuxbrew, você precisará instalar o *Node.js* baixando um pacote em https://nodejs.org/en/download/. Depois você precisará abrir o terminal ou prompt e executar o comando:

```
$ npm install -g appium
```

Com isso finalizamos a instalação do servidor Appium! Para executá-lo, é só digitar no terminal ou prompt:

```
$ appium
```

Você verá que o servidor é iniciado e passa a escutar requisições no endereço *0.0.0.0:4723*

Se tiver dúvidas para instalar o Appium no Windows, um link interessante para procurar por mais informações é https://appium.readthedocs.io/en/stable/en/appium-setup/running-on-windows/.

Se tiver dúvidas com a instalação no Linux, mais informações podem ser encontradas em http://appium.io/slate/en/master/?ruby#android-setup.

Para finalizar, gostaria de deixar algumas dicas para facilitar o dia a dia.

Alguns comandos são bastante utilizados e portanto é muito interessante criar "atalhos" para eles. Esses atalhos são *alias* (apelidos) que criamos no terminal, que permitem utilizar os comandos escrevendo muito menos :)

Para criar os alias, você precisará:

- **Linux e Mac com bash:** criar (se já não existir) um arquivo texto, chamado .bash_profile, na sua pasta *home* e incluir os alias;
- **Windows com CMDER:** editar o arquivo *config/user-startup.cmd* e incluir os alias.

O alias que eu utilizo são:

```
# 1. For Ruby
alias be="bundle exec "
alias bi="bundle install"
```

```
# ANDROID Helpers
# 2. Start Pixel emulator
alias startpixel="$ANDROID_HOME/tools/emulator -avd Pixel_2_API_25_1
-netdelay none -netspeed full -dns-server 8.8.8.8,8.8.4.4 & disown"

# 3. Show the package ID of a given APK
showpkg () { "$ANDROID_HOME/build-tools/26.0.2/aapt" dump badging
"$@" | grep "package: name"; }

# 4. Show the current Activity ID of the running app
alias showactivity="adb shell dumpsys window windows | grep
mCurrentFocus"
```

Explicando um pouco cada um dos apelidos:

1. O par de comandos `bundle exec` e `bundle install` são muito importantes para qualquer projeto que utilize Ruby. Com os *alias* criados, ao invés de executar, por exemplo, o comando `bundle exec cucumber` você irá executar `be cucumber` ;

2. Eu não gosto de abrir o Android Studio somente para iniciar os meus emuladores, acho muito demorado, pois o Android Studio é um aplicativo pesado. Então, prefiro iniciá-los por meio de um comando no terminal, neste exemplo, o comando `startpixel` . Vale lembrar que `Pixel_2_API_25_1` é o nome do emulador na minha máquina, você precisará incluir o nome do seu. Não sabe qual é o nome? Não tem problema, é só rodar exatamente o comando que eu postei aqui. Na mensagem de erro será apresentado o comando que você deve utilizar para descobrir o nome dos emuladores disponíveis no seu computador.

3. Sempre que vamos preencher um caps file para Android, precisamos saber qual é o pacote do projeto. Com esse *alias* configurado, você precisará somente digitar o comando `showpkg path_do.apk` para que nome do pacote seja exibido;

4. Muitas vezes, nossos aplicativos tem um fluxo que não permite ao Appium descobrir automaticamente qual é o identificador da tela inicial, que faz com o que o Appium entenda que ele não está iniciando o aplicativo corretamente (por exemplo quando temos uma Splash Screen). Para resolver esse problema, precisamos colocar esse identificador dentro da caps `appWaitActivity` . Mas primeiro, precisamos saber qual é esse identificador. Para isso, inicie normalmente o seu aplicativo clicando no ícone dele dentro do seu celular ou emulador Android. Quando todas as animações terminarem e a tela estiver carregada, digite o comando `showactivity` no terminal e o identificador da tela será exibido

Espero ter ajudado com esse post. Qualquer dúvida é só deixar um comentário :)

#qaninja #loveqa #automação de testes #ruby #cucumber #appium

Ruby Appium QA Testing