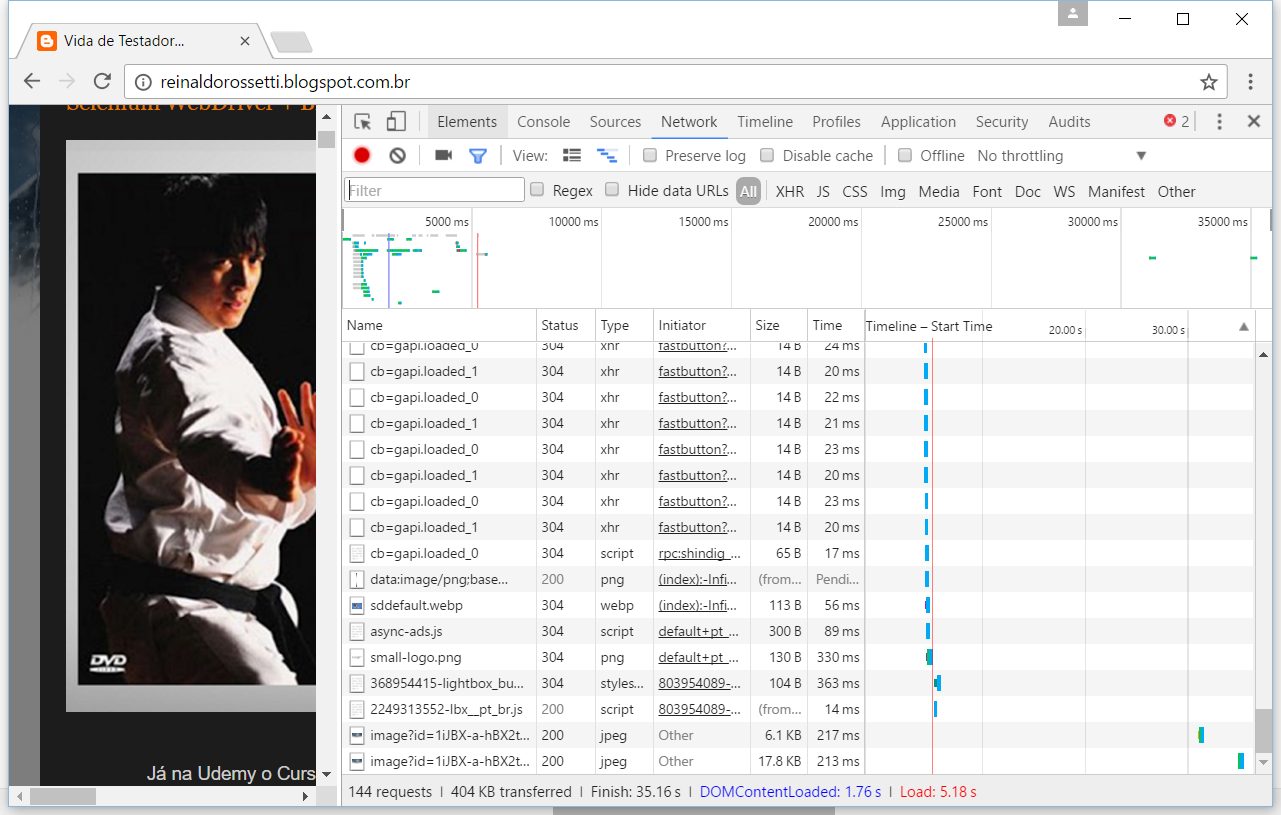
Capturando os dados HTTP das Páginas.

Muita das vezes pra entender determinados problemas, precisamos dos dados gerados via HTTP, algumas ferramentas de análise de tráfico Web trabalham com o HAR file e os browsers também.

O formato HTTP Archive ou HAR, é um formato de arquivo JSON formatado para registrar a interação de um navegador com um site. A extensão comum para esses arquivos é .har. O formato HAR é suportado por vários softwares, incluindo:

* [Fiddler](https://en.wikipedia.org/wiki/Fiddler_(software))
* [Firebug](https://en.wikipedia.org/wiki/Firebug_(software))
* [HttpWatch](http://www.httpwatch.com/)
* Charles Web Proxy (software)
* [Firefox](https://en.wikipedia.org/wiki/Firefox)
* [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome)



*No Chrome podemos gravar o .HAR file e exportar também.*

Podemos pegar os arquivos de três formas, a primeira solução é usar alguns plugins que exportam o .har file, simplesmente você precisa baixar o plugin e configurar o profile adicionando o plugin e os comandos:

**Passo a Passo:**

1. Baixar o plugin e colocar numa pasta em nosso projeto;

https://addons.mozilla.org/firefox/downloads/file/449485/har\_export\_trigger-0.5.0-beta.10-fx.xpi

2. Passar o caminho do Plugin file no nosso código, através da função **add\_extension**, depois habilitar o plugin e o token, em seguida colocamos algumas preferências como mudar o nome do arquivo, qual pasta baixar o .har file,  e se ele vem comprimido ou não.

1. from selenium import webdriver
2. from selenium.webdriver.common.by import By
3. from selenium.webdriver.support.wait import WebDriverWait
4. from selenium.webdriver.support import expected\_conditions as EC
6. fp = webdriver.FirefoxProfile()
7. fp.add\_extension(extension='C:\\temp\\har\_export\_trigger-0.5.0-beta.10-fx.xpi')
8. # Configuracoes para habilitar o plugin.
9. fp.set\_preference("extensions.netmonitor.har.enableAutomation", True)
10. fp.set\_preference("extensions.netmonitor.har.contentAPIToken", "test1234")
11. fp.set\_preference("extensions.netmonitor.har.autoConnect", True)
12. fp.set\_preference("devtools.netmonitor.har.enableAutoExportToFile", True)
13. fp.set\_preference("devtools.netmonitor.har.forceExport", True)
14. # Configuracoes para mudar o nome do arquivo e local, como comprimir o arquivo.
15. fp.set\_preference("devtools.netmonitor.har.compress", False)
16. fp.set\_preference("devtools.netmonitor.har.defaultFileName", "HarOutputExport\_%y%m%d\_%H%M%S")
17. fp.set\_preference("devtools.netmonitor.har.defaultLogDir", "C:\\temp")
19. def \_waitForAlert(driver):
20. return WebDriverWait(driver, 5).until(EC.alert\_is\_present())
22. browser = webdriver.Firefox(firefox\_profile=fp)
23. browser.get("http://www.w3schools.com/js/tryit.asp?filename=tryjs\_confirm")
24. browser.switch\_to.frame("iframeResult")
25. browser.find\_element(by=By.CSS\_SELECTOR, value="body > button").click()
26. alert = \_waitForAlert(browser)
27. alert.accept()

Problema dessa solução, em cada browser terá que baixar um plugin diferente, olhei as reclamações no github, quando o browser atualizar pode deixar de funcionar nas versões novas dos browsers.

2. Temos uma segunda solução melhor usar um servidor de proxy, podemos criar um servidor local e configurar o browser, já vi algumas pessoas usando o fiddler pra isso, mas sei se é um ótima solução usar o fiddler, ele pode interferir nos testes. Eu achei a melhor solução usar uma API do python,  existe uma API chamada browsermob-proxy, a solução baseia em cliente servidor, ele inicia o servidor .jar e configuramos o cliente.

**Passo a Passo:**

1. Baixar o servidor no site abaixo e copiar o mesmo para o c: do sistema;

https://bmp.lightbody.net/

2. Instale a API no Python, abra o cmd/shell e digite o comando: pip install browsermob-proxy;

3. Passar o caminho do servidor no nosso código, sem a extensão:

1. "C:\\browsermob-proxy-2.1.2\\bin\\browsermob-proxy"

**Código Fonte:**

1. # Importa as APIs necessaria.
2. from browsermobproxy import Server
3. from selenium import webdriver
5. # Passa o caminho do servidor browsermob-proxy, nao adicione extensao.
6. server = Server("C:\\browsermob-proxy-2.1.2\\bin\\browsermob-proxy")
7. # Comando pra Iniciar o servidor
8. server.start()
9. # Cria o proxy, e retorna o objeto client com todas as funcoes.
10. proxy = server.create\_proxy()
12. # Configura o Selenium com o proxy.
13. profile = webdriver.FirefoxProfile()
14. profile.set\_proxy(proxy.selenium\_proxy())
15. driver = webdriver.Firefox(firefox\_profile=profile)
17. # Ativa o HAR file para ser gravado.
18. proxy.new\_har("google")
19. driver.get("http://www.google.com")
21. # Retorna o HAR file em formato json.
22. print(proxy.har)
24. # Para o servidor.
25. server.stop()
26. driver.quit()

A vantagem da API, que podemos usar outras funções da classe, como limpar o cache **clear\_dns\_cache**, você pode usar a função **new\_har**e passar como parâmetro a opção que você deseja. Você pode criar uma função genérica no **behave**usando o [behave.userdata] para ativar e desativar os logs.

1. """
2. This sets a new HAR to be recorded
4. :param ref: A reference for the HAR. Defaults to None
5. :param options: A dictionary that will be passed to BrowserMob Proxy \
6. with specific keywords. Keywords are: \
7. captureHeaders - Boolean, capture headers \
8. captureContent - Boolean, capture content bodies \
9. captureBinaryContent - Boolean, capture binary content
10. """

**Mais detalhes acesse os links abaixo:**

http://www.softwareishard.com/blog/har-export-trigger/

https://www.youtube.com/watch?v=FmsLJHikRf8

https://browsermob-proxy-py.readthedocs.io/en/stable/