WebParrot introduction and walk-thru.

WebParrot acts as a caching proxy to allow for repeatable client testing. This tutorial will get you started using WebParrot.

Installation:

WebParrot requires nodejs version 0.6.6 or above.
Nodejs can be dowloaded and installed from
http://nodejs.org/#download
WebParrot itself does not need to be installed, it is run using nodejs

---

Running:

To run WebParrot type: 'node <location of WebParrot.js>'.
If nodejs isn't in your path then you will have to run the node executable from wherever you installed it.
WebParrot has several optional command line arguments, use the argument -h to show more information about these.

Using:

To direct your traffic through WebParrot set it as a http proxy in your browser. Once you have done so then will start caching any traffic passing through it.

Firefox:

Several addons exist, however I suggest FoxyProxy.
Using FoxyProxy add a new proxy using the manual proxy configuration. Enter the hostname or ip of the machine you are running WebParrot on (likely localhost) and enter the port it will use for proxy traffic (default 8080).

Next go to URL Patterns and add a new pattern for the website you want to test. I suggest adding a wildcard to the front of any patterns you input (your address bar will not display the http:// part, but FoxyProxy will see it as part of the pattern). Make sure it's set to whitelist and enabled. Set it to a new color so you can see when your direct connection is being use (the icon will be blue) and when WebParrot is being used.

You're ready to go!

Chrome:

Several extensions exist, I suggest 'Proxy Switchy!'

Using Proxy Switchy add a new profile, set it to manual configuration and enter the hostname or ip of the machine you're running WebParrot on (likely localhost) and enter the port it will use for proxy traffic (default 8080). Make sure the if the site you're testing is being hosted on your machine that you

empty 'No Proxy for' at the bottom of the profiles page.

Next go to switch rules and add a name and pattern for the site to test. Set the Proxy Profile to the one you made for WebParrot

You're ready to go!

IE:
Why are you using IE?!

System Proxy:
I recommend against using WebParrot as your system proxy. If you really want to, then you're on your own.

Admin Page:
WebParrot also runs its own web server for administration. By default this is on port 8081, however it can be changed with a command line argument. Go to <hostname>:<webPort> to access the admin page (e.g. localhost:8081).
Note that WebParrot will never cache pages on its web port (so don't use the same as your site, or 80)

Demo Page:
WebParrot can run another web server to demonstrate its abilities. Add the command line argument '-d <demoPort>' to host the demo page. This page will be shown at <hostname>:<demo port>/demo (e.g. localhost:8082/demo).

Modes:

Translucent: In this mode any unknown request will be sent to the server while any known request will be parroted.
Transparent: In this mode all requests are sent to the server and the parrot acts like a normal proxy, however all responses will be recorded.
Opaque: In this mode no requests will be sent to the server, any unknown requests will receive a blank page. Do not start the parrot in opaque mode.

Cache issues:
If you are trying to view a page you have viewed before your browser may send a cache check instead of a normal GET. If the page has no changed then the server will response by saying that the cached page is correct. This response will be recorded by the parrot. Thus you may try to reload from the parrot only to get a cache response. To avoid this issue either clear your browser's cache before using or set the parrot to ignore that particular request (via the admin page).

Locking and ignoring:
Requests can be set to be locked and/or ignored. Locking a request makes it so that the response will not change, even if a new response is seen. (only makes sense in transparent mode or response is set to

ignored). Ignoring a request makes it so that the request is always treated as unknown. Thus in translucent mode the server will always receive the request and in opaque mode a blank page will always be delivered (thus locking is only useful in translucent mode).