**Automating Chrome Browser from C#**

Chrome Browser supports less known API called Remote Debugging. It is a JSON/WebSockets based protocol, which is basically used by Chrome Developer Tools to attach to a remote web page.

**Note:**

You can read more about this protocol at official page <http://blog.chromium.org/2011/05/remote-debugging-with-chrome-developer.html>

**Setting up environment**

To enable Remote Debugging, you need to start chrome with a parameter, enabling Chrome to listen on a specified Tcp port:

chrome.exe --remote-debugging-port=9222

Also, please ensure, that you dont have firewall blocking traffic to that port.

Once, you have Remote debugging enabled in Chrome, to check whether everything is working fine, we might want to start another Chrome session. This time, it has to be a session for a **different user.**To start chrome under different user, please start another chrome with these parameters:

chrome.exe --user-data-dir=C:\myChromeUser

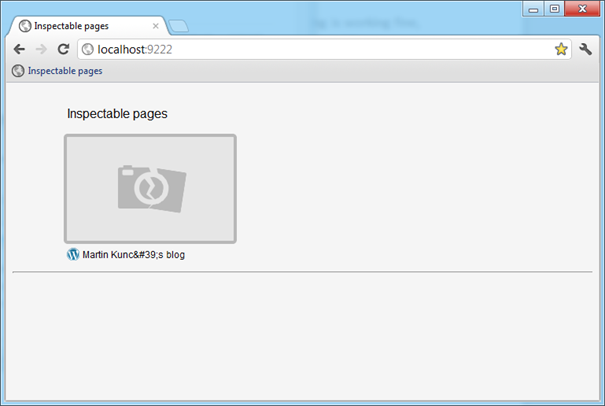
You might want to change the directory to some other temp directory of client user.

Once you start client Chrome for the different user, you just go to the url:

[http://localhost:9222](http://localhost:9222/)

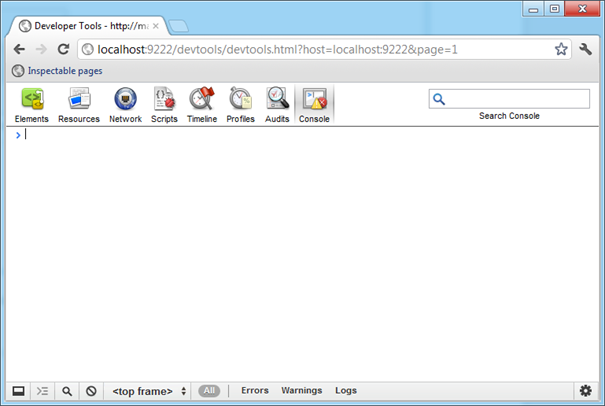
Which will basically open a webpage, offering you to select which tab in Chrome you want to debug remotely.

The page might look like:

[](https://markcz.files.wordpress.com/2012/02/remotedebugginginspectablepages.png)

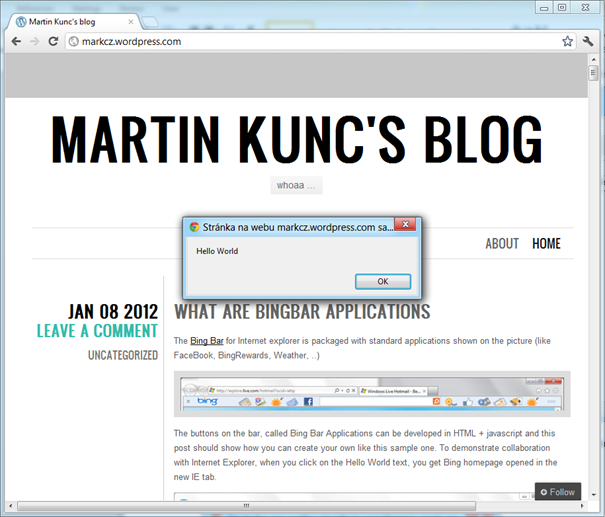
Note: You should been opening this page with Chrome, because other browsers does not support Remote Debugging.

Selecting a Page, you should have Debugging Tools open for the page. It looks similar to the local Debugging Tools:

[](https://markcz.files.wordpress.com/2012/02/remotedebuggingtools.png)

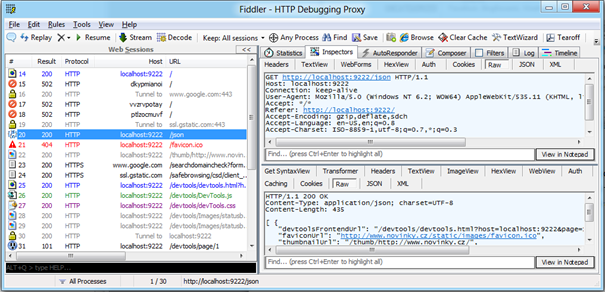
Then, just to try simple command, we might want to test alert being invoked in the debugged browser, simply switch to console and like on a local environment use:

alert('Hello World');

[](https://markcz.files.wordpress.com/2012/02/alertonpage.png)

That was the simple test.

Lets return to automation. Chrome Remote Debugging page describes the protocol as JSON based, with some examples. I wanted to catch some traffic flowing from Remote Debugging Chrome page going to debugged Chrome. To that, I have used [Fiddler2](http://fiddler2.com/fiddler2/) Http Proxy, once I set that to my system proxy, I was able to catch the initial protocol flow. The rest of the protocol flow goes over WebSockets.

[](https://markcz.files.wordpress.com/2012/02/fiddler-json.png)

**WebSockets in place**

When you look at the result in fiddler, (or simply use your browser and go to[***http://localhost:9222/json***](http://localhost:9222/json). Note: When you dont see WebSocketDebuggerUrl in response, you are probably connected to RemoteDebugging endpoint with a different client) you will see something like:

[ {

   "devtoolsFrontendUrl": "/devtools/devtools.html?host=localhost:9222&page=1",

   "faviconUrl": "http://s2.wp.com/i/favicon.ico?m=1311976027g",

   "thumbnailUrl": "/thumb/https://markcz.wordpress.com/",

   "title": "Martin Kunc's blog",

   "url": "https://markcz.wordpress.com/",

   "webSocketDebuggerUrl": "ws://localhost:9222/devtools/page/1"

} ]

As you can see, this is pure Json. Anyway, what we are interested in the most is

   "webSocketDebuggerUrl": "ws://localhost:9222/devtools/page/1"

This element points to WebSockets endpoint, where Chrome is willing to listen for Json formatted requests.

The problem here is that currently Fiddler does not have nice support for [WebSockets](http://en.wikipedia.org/wiki/WebSocket). Even if they goes over same ports, they are socket based. Fiddler beta has preliminary support for it in its Log tab, but its not so fancy formatted.

**WebSockets in .NET**

I had troubles with looking for a nice WebSockets Client implementation and then finally I found [WebSocket4Net](http://websocket4net.codeplex.com/), which is WebSocket client implementation primarly used for SuperWebSocket server, but works pretty nice here.

**Preparing Json commands for Remote Debugging protocol**

Using the fiddler and its Log tab I was able to see enough samples of Json commands, which Remote DP is using. It is also described [here](https://developer.chrome.com/devtools/docs/debugger-protocol). I have used v 0.1 of it, because I currently have Chrome 17.

**Finally driving Chrome**

Here is my snippet how I am driving Chrome in C#:

|  |
| --- |
| var chrome = new Chrome("http://localhost:9222");  var sessions = chrome.GetAvailableSessions();  Console.WriteLine("Available debugging sessions");  foreach (var s in sessions)  Console.WriteLine(s.url);  if (sessions.Count == 0)  throw new Exception("All debugging sessions are taken.");  // Will drive first tab session  var sessionWSEndpoint = sessions[0].webSocketDebuggerUrl;  chrome.SetActiveSession(sessionWSEndpoint);  chrome.NavigateTo("http://www.google.com");  var result = chrome.Eval("document.getElementById('lst-ib').value='Hello World'");  result = chrome.Eval("document.forms[0].submit()");  Console.ReadLine(); |

Thats basically all, it is just a proof of concept application.

Sources are available on the GitHub link: <https://github.com/markcz/automate-chrome>