

Schooljaar: 2018 – 2019

Datum test / examen:

…

. / .... / ....

Leerkracht :

Bruno SEYS

**Afdeling**

:

**HBO5 INF**

Module

:

**C4 Internet Technologie**

Naam cursist: **Desmet Tom**

Handtekening:

Puntenoverzicht

Permanente evaluatie

/

Schriftelijk examen

/

Mondeling examen

/

Totaal

/

Te herleiden naar

/



**Web Services & XML RPC**

**Discovering Web Services using**

**XML-RPC technology**



# Index

[Index 2](#_Toc534406122)

[1) Study material 3](#_Toc534406123)

[2) Objectives 3](#_Toc534406124)

[3) Challenge 7](#_Toc534406125)

# Study material

* + - * The “Discovering Web Services” presentation
      * Lynda.com “ Foundations of Programming Web Services “ movies
      * Python XML-RPC server script, a basic weather station simulator

(SimpleXMLRPCServer\_init.py)

* + - * Python XML-RPC client script, to retrieve wheather station data

(SimpleXMLRPCClient\_init.py)

* + - * Template for the JavaScript based XML-RPC client
      * Watch and digest the (first nine) "Foundations of Programming Web Services " lynda.com

movies. You should be ready now to experiment with the concept of Web Services in an

XML-RPC python2 environment. If needed, XML basics are very well explained on

<http://www.w3schools.com/xml/default.asp>

* + - * Run and understand the provided python2 client and server scripts. The respective

"xmlrpclib" and "SimpleXMLRPCServer" python2 on-line documentation are an excellent

start point to get grip.

Hint: I have found a series of excellent related Youtube movies which might come in handy.

<https://www.youtube.com/watch?v=uOuNajsNztA>

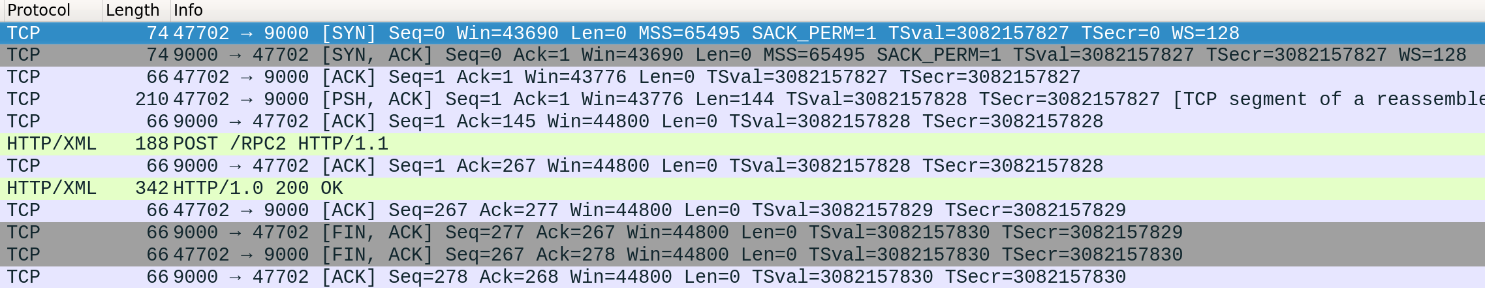
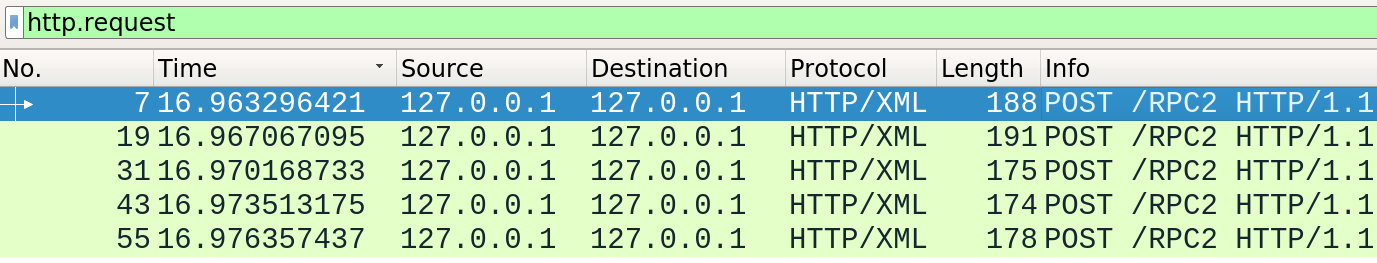
<https://www.youtube.com/watch?v=-Rc3i6sDzNA>

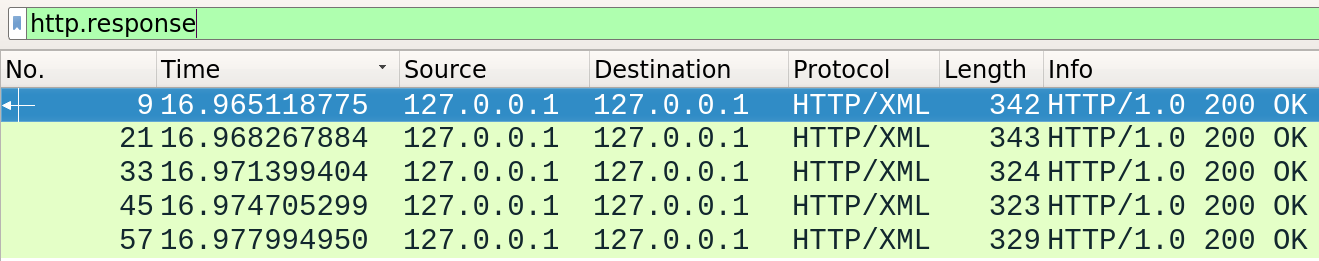
<https://www.youtube.com/watch?v=Ivu0AnVxWsY>

<https://www.youtube.com/watch?v=p1kBNZkV7fk>

# Objectives

1. Analyse the XML-RPC/HTTP protocol stack using Wireshark or tcpdump. Another excellent option is to use the ngrep tool (sudo ngrep -ed lo -W byline).

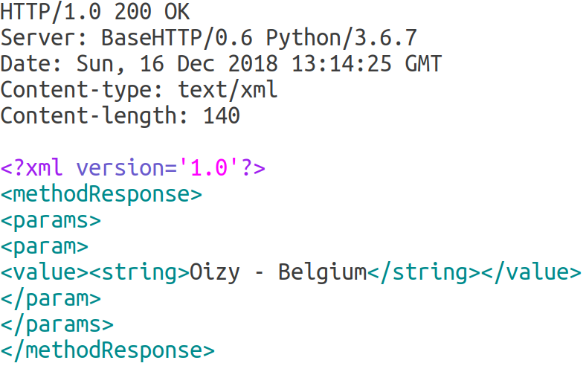
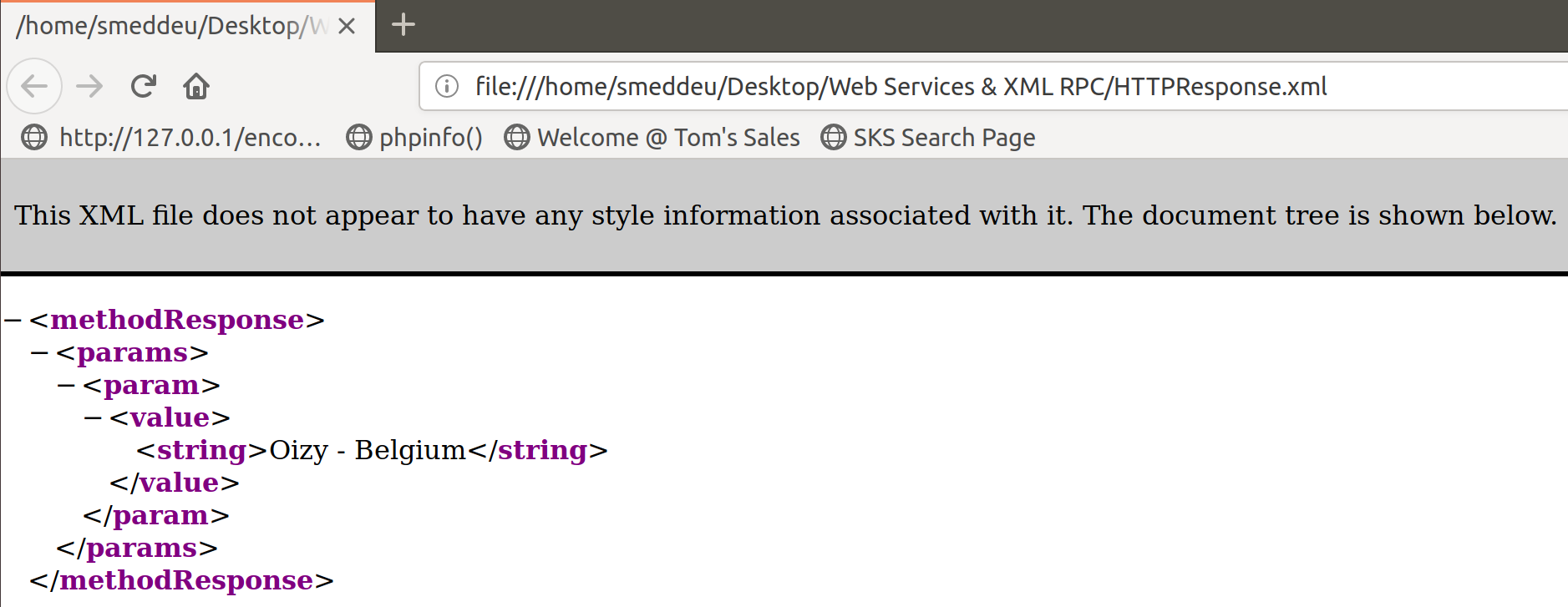
* Identify TCP Handshakes for establishing and closing TCP sessions.
* Discover HTTP Request, HTTP Method, HTTP version, HTTP status code



* Discover XML message format, data structure and data types which are transported in

XML tags (see <http://xmlrpc.scripting.com/spec.html> ).

Hint: One could save the XML in a file and open it in a browser to view properly the

XML tree.

1. How many requests does the client need to send in order to get all required data from the

provider ? Five, I would say! I hope you agree with me.

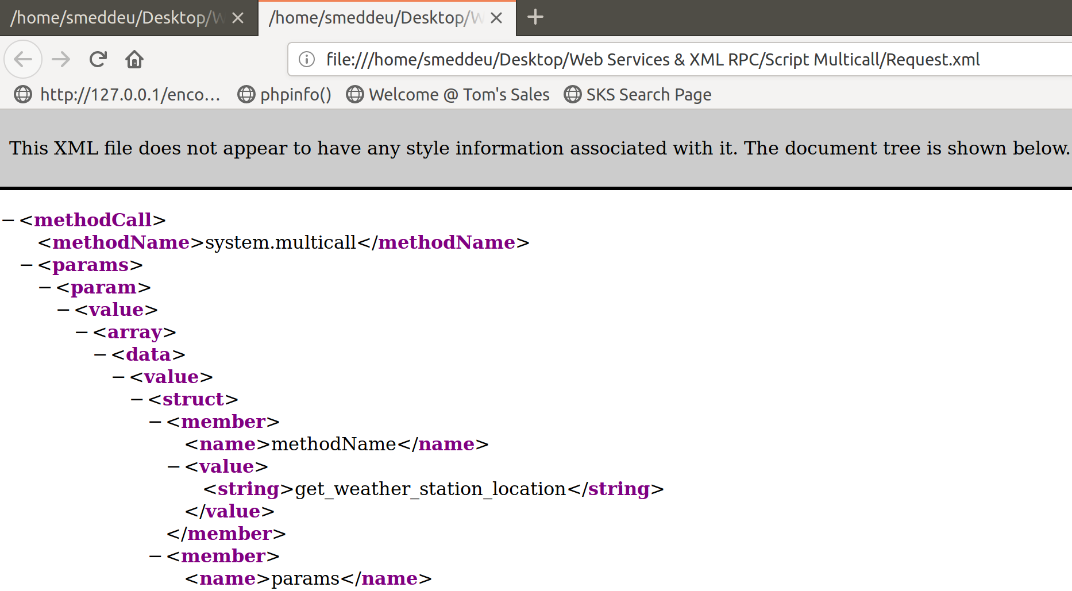
Use the xmlrpclib "multicall" method to reduce the number of methods to one.

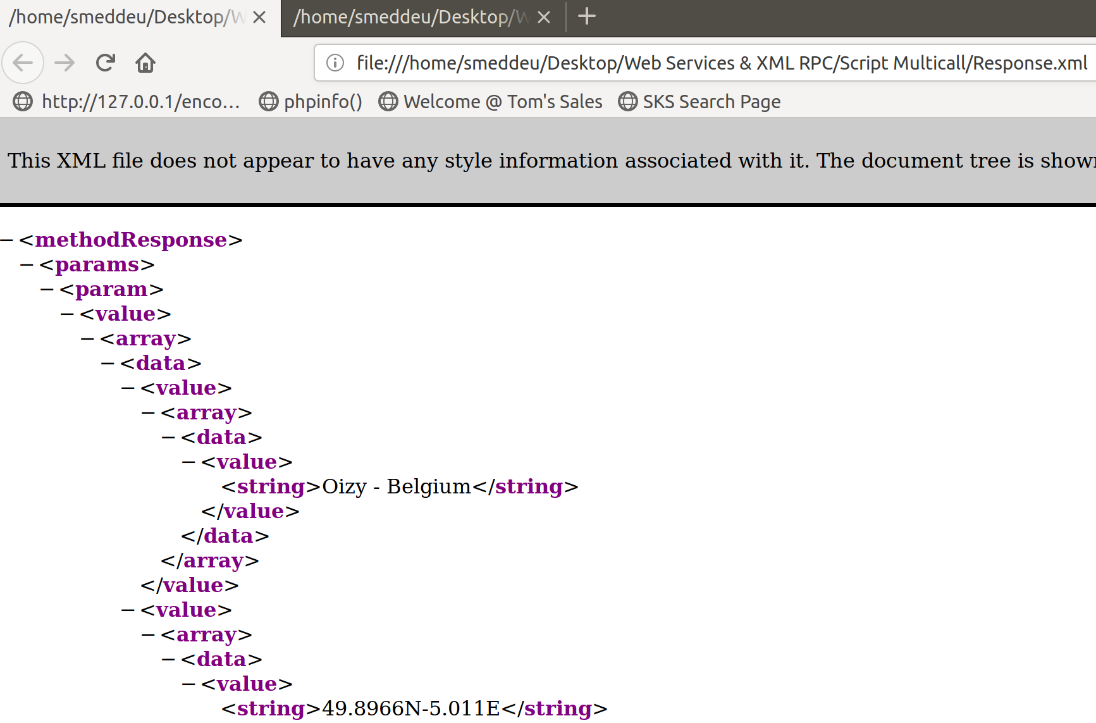
Modify both requester and provider scripts to get it work.

Do you discover new XML tags and structures in the “MethodCall” and

“MethodResponse” ?

In the MethodCall, the methodName “system.multicall” is brought up and the server functions that the

client wants are given to the server as parameters in an array.

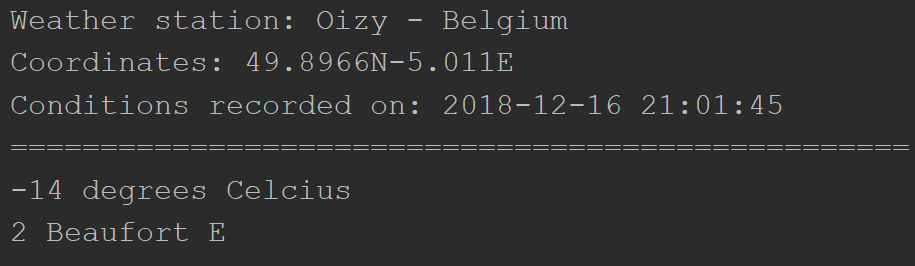
This is the same for the MethodResponse back to the client, value’s are given back as an array with the correct corresponding datatypes.

1. For what reason would the listMethods() method be useful ?

Can be used to return A docstring of the available methods that are registered for use on the server.

['get\_temperature', 'get\_weather\_station\_coordinates', 'get\_weather\_station\_location',……

1. Add some code lines to get the server date/time when pulling the weather info.

Do you see the new data type introduced ?

def get\_date():

my\_datetime = datetime.datetime.now()

return my\_datetime.strftime("%Y-%m-%d %H:%M:%S")

server.register\_function(get\_date)

No, the server will return the datetime value as string as seen in the code above.

1. At the end of the ride, save Server and Client scripts as SimpleXMLRPCServer\_final.py and SimpleXMLRPCClient\_final.py

# Challenge

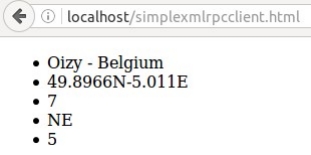
I have highlighted in the Web Services presentation: “Web based API's allow interaction between

devices running on different architectures and speaking different native languages”.

Hence, could you develop an elementary JavaScript that provides identical functionality as the

“SimpleXMLRPCClient\_int.py” script talking to the python SimpleXMLRPCServer ?

You can integrate the code into the “simplexmlrpcclient\_template.html” page.

The output might look like the figure here below.

No need to implement the “multicall” feature as within python2 environment.

I recommend to use the “MIMIC XML-RPC JavaScript API”. Starting point is

http://mimic-xmlrpc.sourceforge.net/

You will require a basic Web server to retrieve the “simplexmlrpcclient.html” page via TCP 80.

This html page encapsulates the JavaScript that on his turn will call the XML-RPC Server via TCP

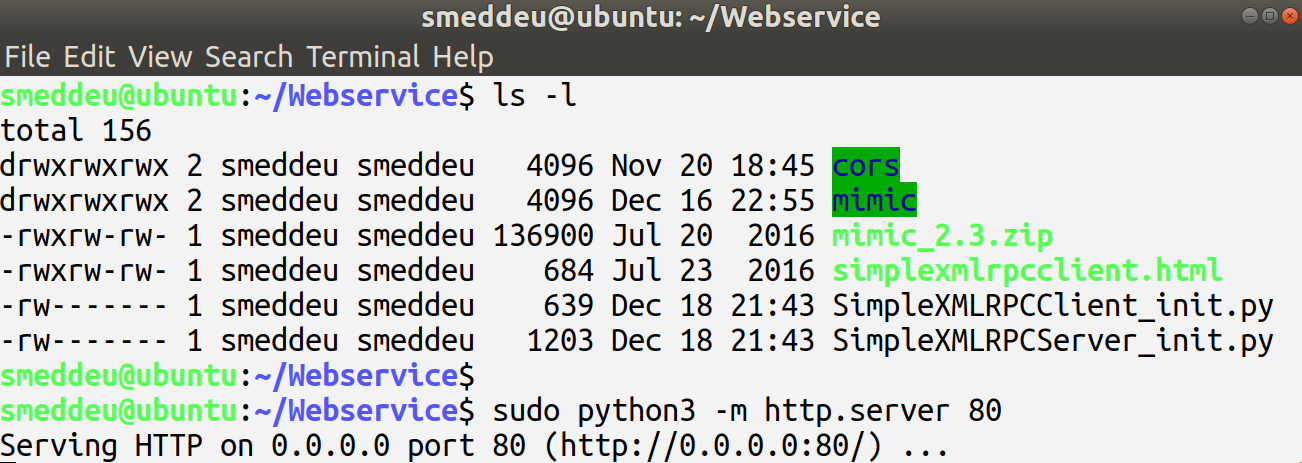
9000.

Launch the inbuilt Python2 Web Server module SimpleHTTPServer. Invoke the server by typing

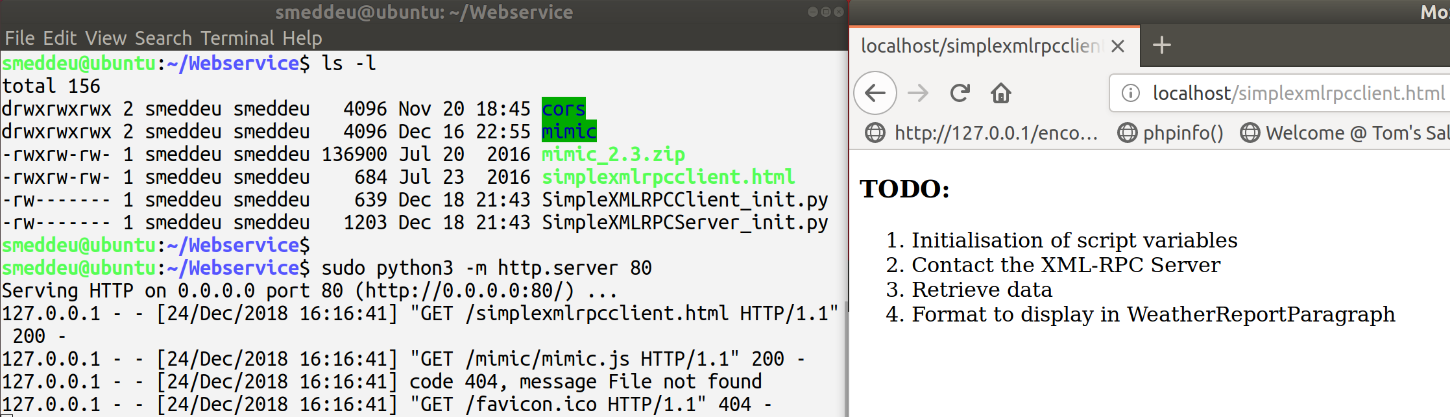
“sudo python -m SimpleHTTPServer 80”. Very basic stuff but it is all what you need.

First we shutdown the Apache2 server as this is running on Port 80.

* **Sudo systemctl stop apache2**

We place all the needed files in the respective folder and then run the Python3 webserver module command. **!! ATTENTION !!** **The given Python2 command by the tutor is outdated!**

**Testing the provided script by the tutor:**



There are a couple of hurdles to make this story successful. First of all CORS comes into theater.

For security reasons, modern browsers do not allow access across domains. This means that both

the web page and the XMLRPC data it tries to load, must be located on the same server. Refer to

the the attached “What is CORS ?” document to understand the mess where you got yourself into.

Second issue is that by default the standard Python3 SimpleXMLRPCServer does not implement

the OPTIONS operator. He following links can point you in the right direction.

https://stackoverflow.com/questions/3248320/xml-rpc-javascript-unsupported-method-options

<http://spynaej.eu/links/?searchtags=sop>

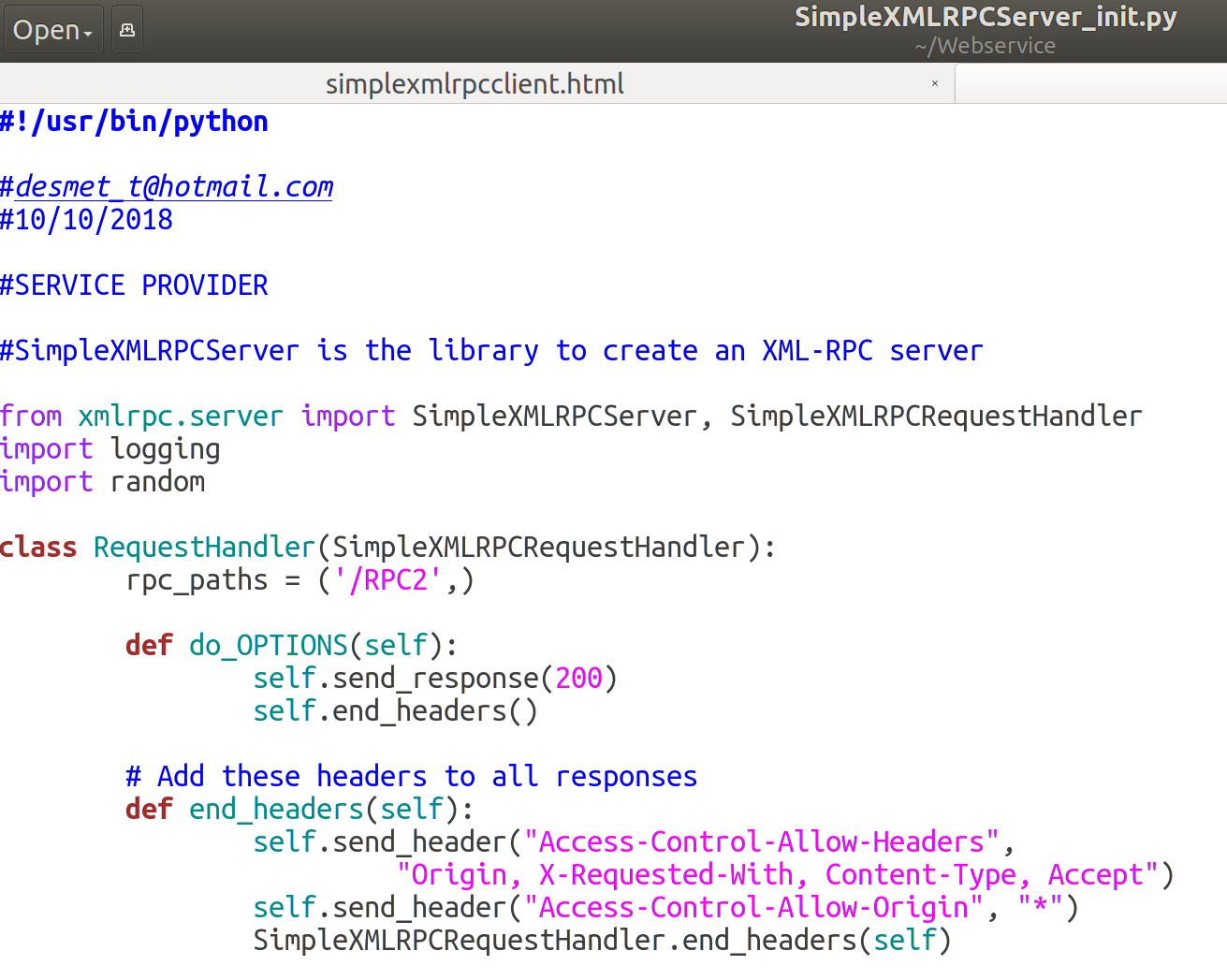
Analysing HTTP headers and data exchange packets using Wireshark and the Firefox Web

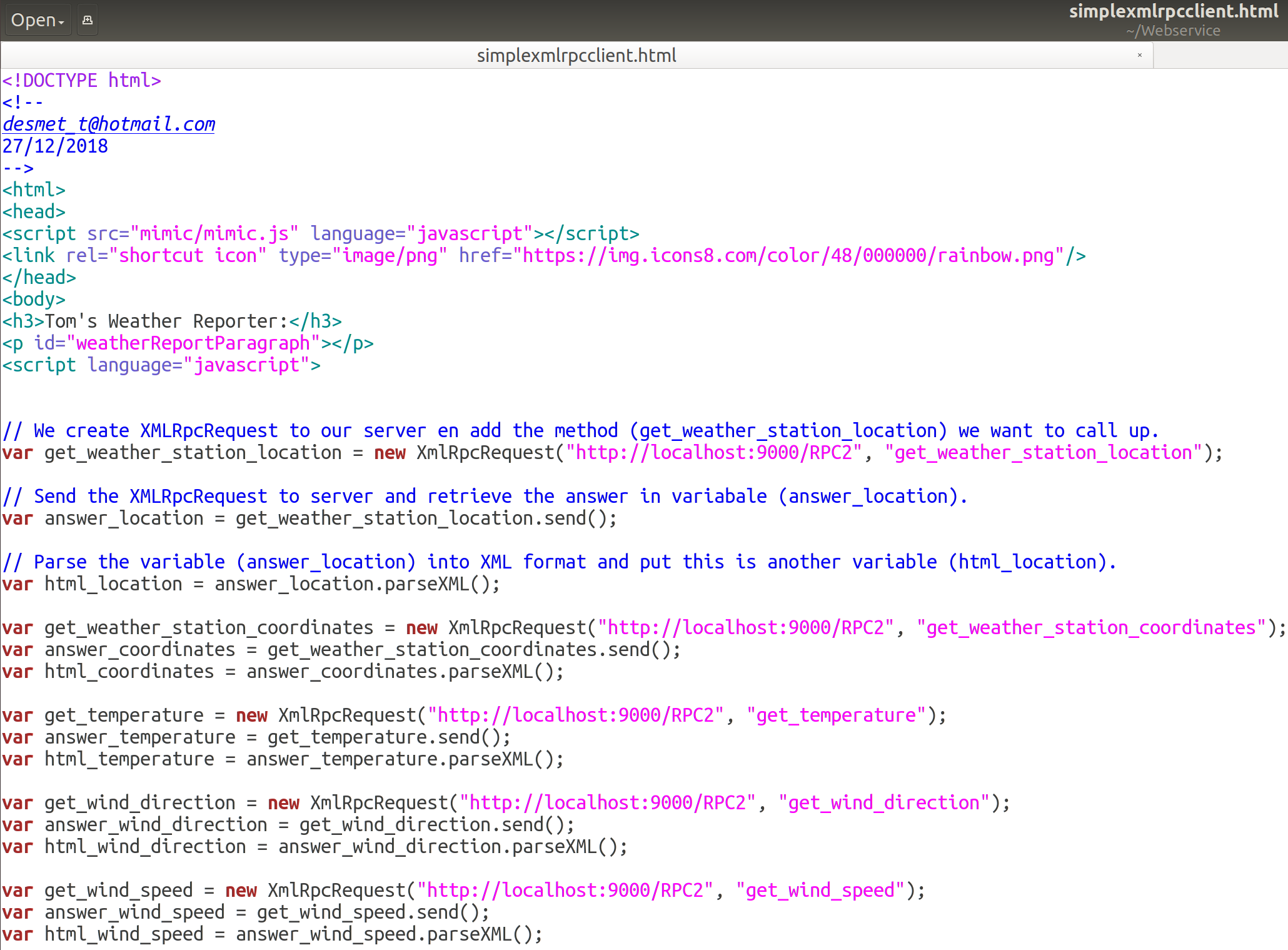
Developer Toolkit (F12 key) are the keys to success.

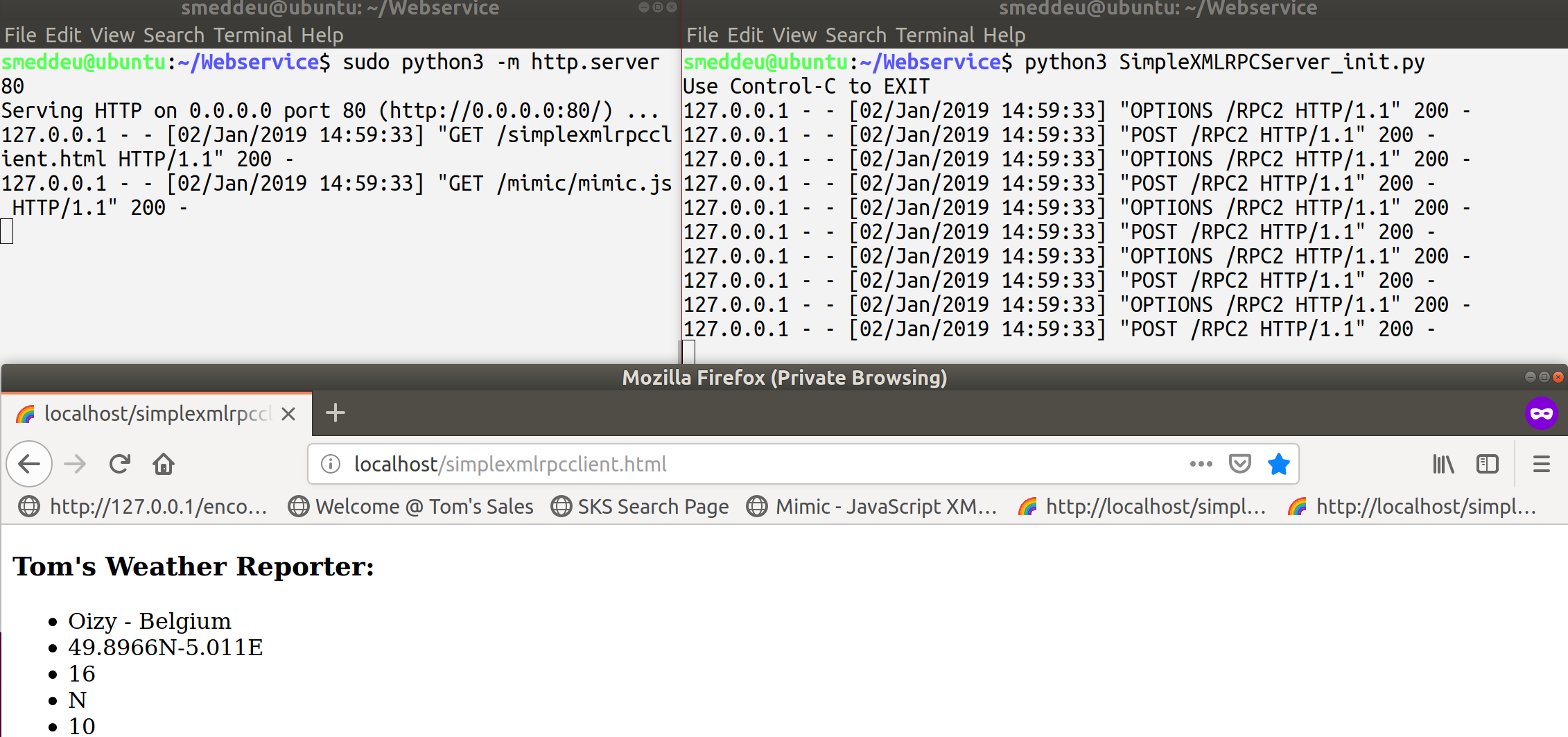
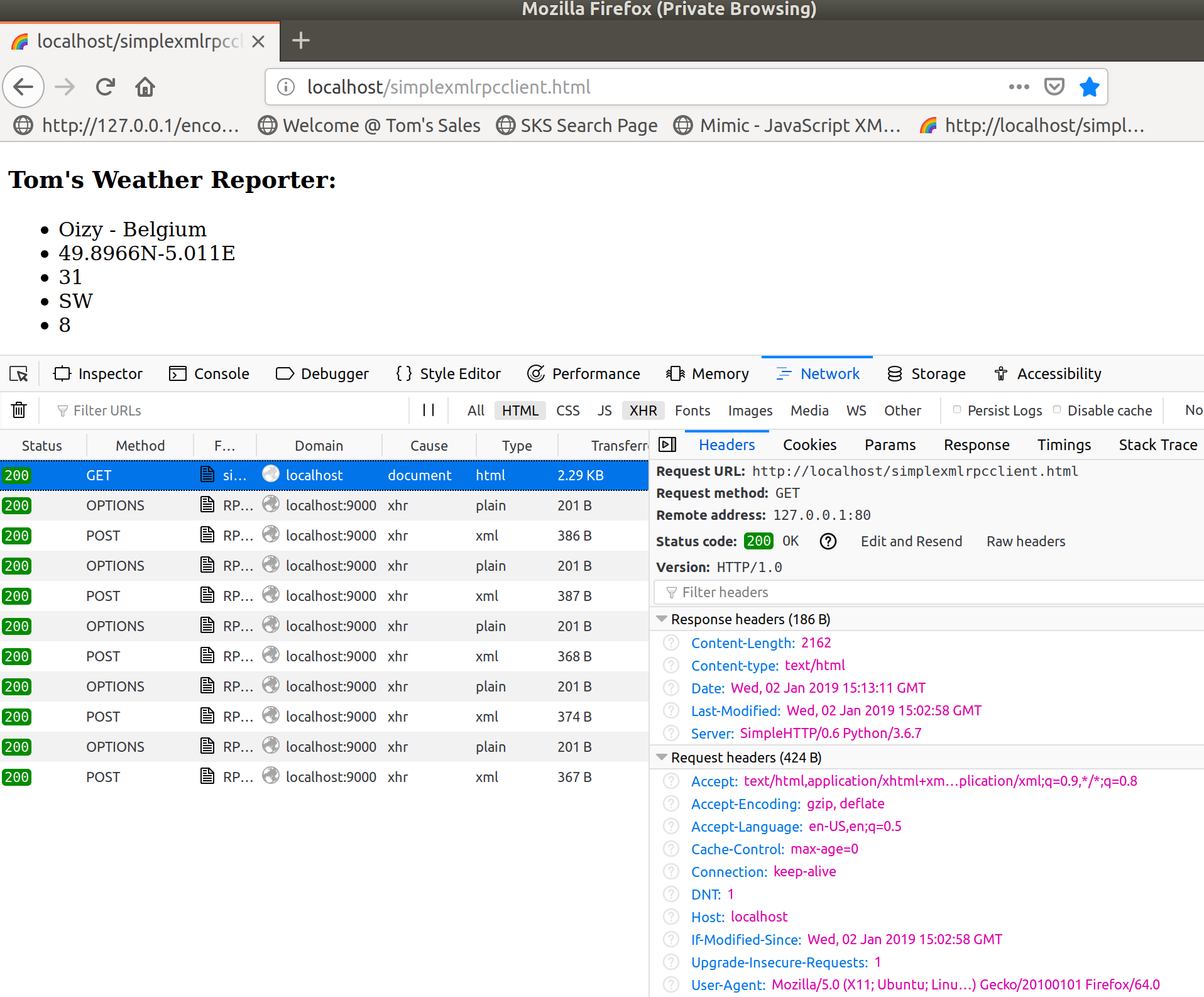
**We implement the following code into the header to add our Favicon to the webpage.**

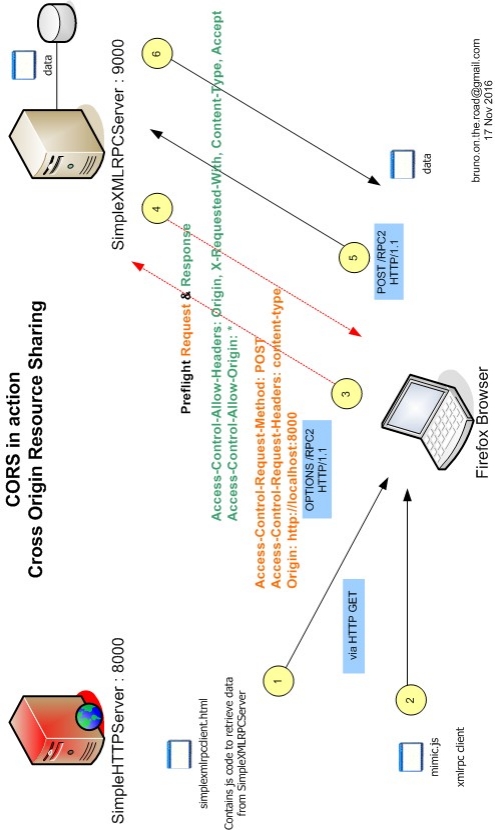
**SimpleXMLRPC Server:**

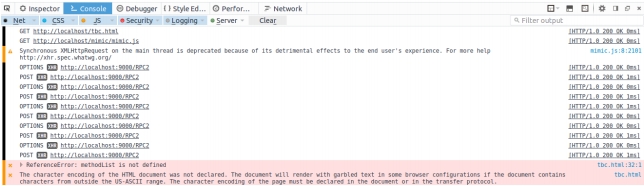
**We add the following code to the existing SimpleXMLRPC Server python3 code.**

**We’ll create an class that will properly handle all our requests, as provided in the hyperlinks above.**

**SimpleXMLRPC Client HTML:**

**Results:**





**FYI: XHR, AJAX, JSON & XML-RPC**

The above screenshot shows “XHR” symbols. All modern browsers have a built-in

XMLHttpRequest object to request data from the server. The keystone of AJAX (Asynchronous

JavaScript and XML) is the XMLHttpRequest. More info, usage and examples on

http://w3schools.com/xml/ajax\_intro.asp .

The name implies that XML is used. However, many prefer to use URL-encoded data when sending

data from the client to the server and JSON as the response. URL-encoded data sent by JavaScript

code matches HTML form data sent by a browser when a user clicks the submit button.

