Distributed Programming I (03MQPOV)

Laboratory exercise n.4

HTTP captures and inspection

The purpose of this exercise is to inspect HTTP requests and responses. Use the "curl" program under Linux in order to send HTTP requests and receive corresponding responses. It is possible to save separately the header of the HTTP response sent by the server by means of the –D option and the full response by means of the –o option. After each test, check the contents of the header and of the response. Also, use wireshark (the program already used in previous exercises) in order to monitor the full data exchange on the TCP channel (use interface eth0 for captures).

Here is a list of the tests to be performed:

1) Request a specific resource

```
curl -D header.txt 'http://media.polito.it' -o response.html
```

2) Send request to a search engine with parameter encoded in the URL string

```
curl -D header.txt 'http://search.yahoo.com/search?p=hello' -o response.html
```

Note that the server tries to set a cookie (where do you see it?)

3) Send request to a search engine with parameter in the body of the message (POST method)

```
curl -D header 'http://search.yahoo.com/search' -d 'p=hello' -o response.html
```

Repeat the previous requests in the Chrome or Chromium browser, and show the inspector window by pressing CTRL + SHIFT + I (note that use instructions are available at https://developer.chrome.com/devtools) and focus attention on the **Elements** and **Network** tabs:

4) Go to `http://media.polito.it` by typing the URL in the browser bar and verify that it is possible to highlight the various elements of the page, corresponding to the various HTML elements, by simply selecting the HTML code itself. Moreover, check that using the inspect element command from the contextual menu (right mouse button) on the various elements of the page, the corresponding HTML code is shown.

5) Go to `http://search.yahoo.com/search?p=hello` by typing the URL in the browser bar. Note that in the Network tab it is possible to view the temporal diagram of the various HTTP transactions and the content of the request and response headers for each of them by clicking on it (or by clicking on View Source). This is possible even if the protocol used is HTTPS.

Experiments with HTML5 and CSS

HTML files can be written using a normal text editor (like Notepad, gedit, nano, or vi). The file name can have any of the two extensions **.htm** or **.html**.

If pages are of static type and are properly written with relative links, then it is possible to position them in any folder of the filesystem and start navigating by directly opening them with any browser.

If pages are dynamic (server-side), then it is necessary to install them on a real web server, which will be shown in the next lab.

For all the proposed exercises, always check the produced pages with at least two different browsers (e.g. Firefox and Chrome), changing the size of the window in order to verify the graphic quality of your solution.

Exercise 4.1

Develop an HTML5 page that uses different HTML5 document structuring elements: **`<header>`**, **`<nav>`**, **`<article>`**, **`<section>`**, **`<aside>`**, **`<footer>`**. Add some text to each element so that you can see where they are placed by default by the various browsers.

Exercise 4.2.1

Add an internal CSS style (in the header of the HTML5 page) in order to display a visible border for each of the previously used HTML5 elements, and observe the result.

Exercise 4.2.2

Move the previously defined CSS rules to an external stylesheet file, which must be loaded automatically with the HTML page. Check that everything keeps working as before.

Exercise 4.3

Add a form to the central part of the previous HTML5 page (in a **`<section>`** element). The form should include typical HTML5 controls, such as:

- Email
- URL
- Number
- Range
- Color
- Date

Remember to assign each control an appropriate name by means of the **`name`** attribute. Test various input values and observe how the browser behaves.

Set the **`action`** attribute of the form to the page itself (localhost), and the method to GET. After having selected a value for each one of the controls, submit the form by means of the submit button.

Observe, on the browser URL bar, the values the browser would send to the server.

Change the method to POST, and observe the behavior by means of the browser debug tool (Network tab), looking at the contents of the HTTP request. Verify that the behavior does not change if the browser changes.

Exercise 4.4

Add a **`<video>`** element to your previous HTML5 page, and enable automatic video controls, so that they are automatically added by the browser when the page is displayed. The **`<video>`** element must point to a multimedia file previously put I n the same folder of the HTML5 file. Verify that the classic control functions (play, pause, seek) work.

Exercise 4.5 (HTML validation)

Check the correctness of all the generated pages by using the W3C validation service:

http://validator.w3.org

Correct any error signalled by the service.

Exercise 4.6 (CSS layout)

Modify the personal web pages of exercise 4.3.1 in order to respect the following specifications:

- the page layout (position of elements header, aside, footer, etc) must be corrected, if necessary, by means of an external CSS (for this purpose, you can use the guides and tutorials available online, e.g. http://www.w3schools.com/Css/css_float.asp).
- the header must have a red border
- the footer must have a white border in its upper part

Verify that the layout is properly maintained even when the user reduces the horizontal size of the browser window.

Exercise 4.7 (Optional)

Explore some parts of the HTML and CSS code provided with the examples of a web programming framework such as for example Bootstrap, which is available at the URL http://getbootstrap.com/getting-started/