DFTI Worksheet 1

Clients, Servers and HTTP; HTML/CSS revision

Introduction

In the DFTI unit we will be developing dynamic websites which *interact* with the user. In the practical tutorial sessions, you will be building up *HitTastic!*, a dynamic, interactive music website, which, when you complete it, will allow the user to search a database of top 40 hits.

This first worksheet serves four purposes:

- to familiarise yourself with file transfer to a web server;

- to understand communication between web browsers and web servers;

- to introduce the PHP learning environment *EPHP* and use it to become familiar with how clients and servers communicate with each other via HTTP requests and responses;

<u>You should note down everything you learn today - and every week - in a logbook, either in a file on the computer (via a word processor/text editor) or on paper. This logbook will help you when you do the assessment.</u>

Part One - Creating a basic HTML page and uploading it to the server

Create a web page

Presented below is a very basic web page, using the basic HTML that you learnt into the first year. Copy and paste this into Notepad++ or another text editor, and save it to your U: drive with the filename "index.html"

```
<!DOCTYPE html>

<html>

<head>

<title>HitTastic!</title>

</head>

<body>

<h1>HitTastic!</h1>

<p>Search and download your favorite 40 hits on
HitTastic! Whether it's pop, rock, rap, or pure liquid
cheese you're into, you can be sure to find it on
HitTastic! With the full range of top 40 hits from the
past 60 years on our database, you can guarantee you'll
find what you're looking for. Plus with our Year Search
(coming soon!) find out exactly what was in the chart in
any year in the past 60 years. </p>

<form method="get">

<fieldset>

<label>Please enter an artist:</label>

<input name="theArtist" />

<input type="submit" value="Go!" />

</fieldset>

</form>

</body>

</html>
```

The Neptune Server

You will upload the class work to a server called Neptune. You each will have your own Neptune login of the form ephpXXX, where XXX is a three-digit number. You can get these from the unit website as instructed in the lecture.

Uploading work using FileZilla

Open FileZilla to transfer the HTML page above to Neptune. FileZilla is an FTP client; it is used to transfer files from one machine to another. We will demonstrate FileZilla in class. Login to Neptune with the details:

Host (URL of server): `ephp.solent.ac.uk`

Username: `your Neptune username`

Password: `your Neptune password`

This will upload the file via FTP.

Prove that it has indeed uploaded to Neptune by opening your web browser and going to this address:

http://ephp.solent.ac.uk/~ephpXXX/index.html

Part Two - Exploring Web Communication In More Detail

We will be using a simple visualisation environment - **EPHP** - for basic PHP web development, which has been developed by the unit leader of DFTI.

EPHP has been installed on Neptune, and is available at:

http://ephp.solent.ac.uk/

You must use Chrome, Firefox or Edge to access it; Internet Explorer will not work as it does not follow modern web standards.

In this section you will use EPHP to allow you to visualise the communication between a client and a server.

Note how it consists of three windows, the client, the network and the server – your lecturer will explain this in class.

Question 1

a) Login to EPHP (http://ephp.solent.ac.uk) using your ephpXXX login, the same one you used on FileZilla to transfer files to the server.
Enter the address of your index page on the Neptune server *within EPHP*, not in a separate browser tab. To do this, go to the “Simulation” tab on the client side (left hand side) of EPHP and enter:

http://ephp.solent.ac.uk/~ephpXXX/index.html

(where *ephpXXX* is your Neptune username). Click “Go”

b) Watch the animation until you see the page you asked for in the “client” window of EPHP.

At this point, your lecturer will explain what happens in between the user entering a URL in the web browser and the web page coming back as a response.

Question 2

a) Download the HitTastic! logo PNG image from the URL
http://ephp.solent.ac.uk/~ephp001/hittastic.png
and save it to your U: drive. Use FileZilla to transfer it to your space on Neptune.

(Apologies for the bad artwork - I am not an artist!)

b) Type in the URL of your index page (not the image) once again in EPHP. Stop the HTTP request half way across by pressing the "Stop" button and edit it directly to request the PNG image instead.

Restart the request and allow the response to come back to the client side. What appears and why?

Question 3

Roll the mouse over the “HTTP response” box and examine the structure of the HTTP response. At this point, we will discuss the structure of the HTTP response in detail.

Write down two ways in which the HTTP response for the image differs from that returned when your requested an HTML page. One should be from the header, and one from the content.

i) Change in header:

ii) Change in content:

Question 4

Using what you saw in the previous exercises (questions 2 and 3), how do you think that your web browser knows how to display a web page sent back from the website as a web page, and an image sent back from the website as an image? *(Multiple choice - choose one answer – highlight below)*
i) By the file extension, e.g. .html means a web page, and .jpg means a JPEG image.
ii) By examining the HTTP response for the presence of tags such as <html> and <body>. If tags are there, the browser displays it as an HTML page. If not, it's treated as an image.
iii) By examining the content-type: line of the HTTP header.

We will take a vote on this and then your lecturer will give you the correct answer.

Question 5

a) Use EPHP to request the PNG image again. Stop the HTTP response halfway back once again. Change the "content-type" (otherwise known as the **MIME type**) to text/plain. Restart the HTTP response once again. Now what happens when the response is loaded into the browser? Why? (answer in logbook)

b) Now request your index page again in EPHP, I.e

http://ephp.solent.ac.uk/~ephpXXX/index.html

Once again, stop the HTTP response halfway through.

Change the **content-type** line in the HTTP header to the following:
i) text/plain
ii) image/png

What happens in each case, and why? (answer on the board, once again)

Question 6

Now request this page in EPHP. It's a page that doesn't exist.

http://ephp.solent.ac.uk/~ephpXXX/imaginary.html

Look at the HTTP response returned this time. In what important way does it differ from the previous HTTP responses?

This is the point I am expecting you to get by the end of the first session of the week.

Part Three - HTML and CSS revision

This exercise gives you the opportunity to further revise HTML before we begin PHP, next week. There are HTML revision notes available on the unit website, https://edward2.solent.ac.uk/course/.

1. Create a second page, signup.html. Give the form a method of "post" rather than "get". (you will see why later in the unit). This should allow a user to sign up for HitTastic! Add a form with six fields:
   - Username
   - Password
   - Name
   - Day of birth
   - Month of birth
   - Year of birth

Again, shortly you will make the signup actually work. Also, again make sure the form has a submit button.

Again, upload it to the Neptune server via FileZilla or within EPHP.

Also, once again prove that it has indeed uploaded to Neptune by opening a separate tab in your web browser and going to

http://ephp.solent.ac.uk/~ephpXXX/signup.html

2. At the bottom of the index.html page, add a hyperlink to the signup.html page. On the signup page, add a hyperlink back to the main page. Test the hyperlink works by viewing them in your browser - first go to the index page at

http://ephp.solent.ac.uk/~ephpXXX/index.html and follow the link to the signup page.

3. Style your two pages using an external CSS stylesheet.

If you finish, read ahead on PHP (Topic 2) and start next week's worksheet.