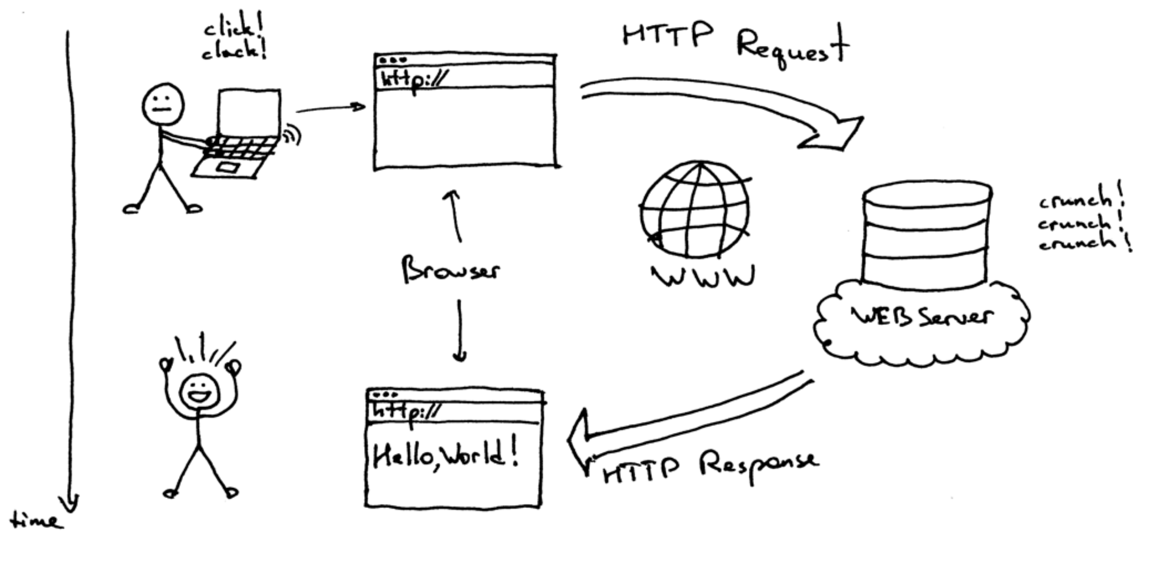
ASSIGNMENT 2

Computer Network : Intordcution

Zacky Kharboutli *Charles Mairey*



Contents

1 Introduction

1.1 Problem overview

This is the second assignment of Computer Networks course. The goal of the assignment is to develop a web server with java that serves static content. The server will be using a server socket and HTTP protocol

- The first part is about creating the web server that accept multiple clients
- The second part is to create the server responses for different situations.
- Using the server with Putty instead of the web browser

The report will address the way the problems were solved and the some pictures were add to show the results.

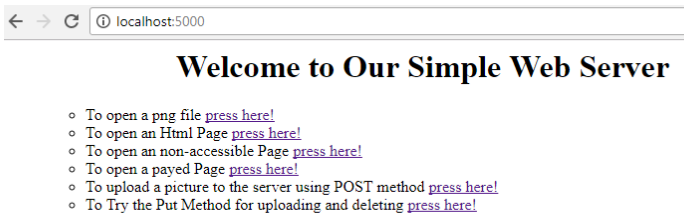
1.2 Responsibilities

Zacky Kharboutli: worked on the server code and the functionality of the server in general. Implemented the error responses and created the base for the whole server. Did the test and the debugging of the server and improved the code quality

Charles Mairey: Implemented the post and the put methods and added some code for the functionality of the server. Did the whole report added the images to it. Moreover, testing the server every time a new function was added.

2 Problem One :

After launching the server, it runs indefinitely until the client manually terminate it. When the socket get a request for a connection, it creates a new thread dedicated to this client's requests. Then requests can be made from the browser using "localhost:5000" to start the request (5000 is the used port and it is hard coded). We started by typing in the browser localhost:5000 and in this case the server will find the index.html in the folders(Note: the code will remove automatically the "/" if it is exist in the end of the path.)



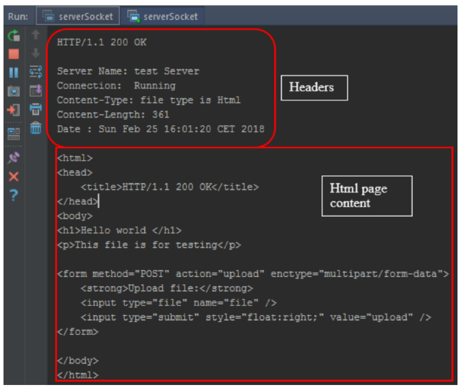
localhost:5000

Welcome to Our Simple Web Server

- To open a png file press here!
- To open an Html Page press here!
- To open an non-accessible Page press here!
- To open a payed Page press here!
- To upload a picture to the server using POST method press here!
- To Try the Put Method for uploading and deleting press here!

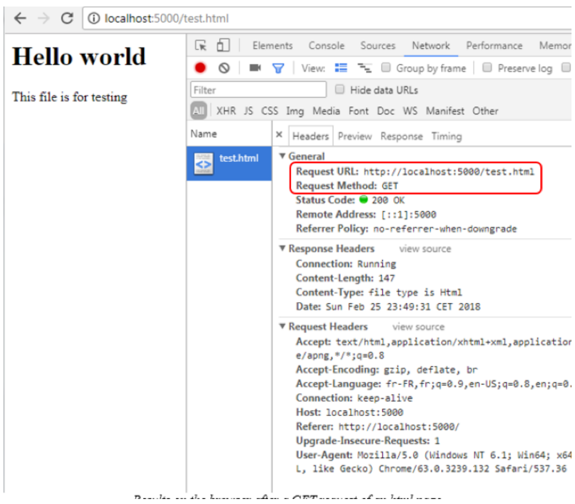
Figure 2.1: index.html represented and it gives links to almost all the function in the server

When requesting "localhost:5000/test.html" for example, the server receive a GET request a return the html page defined by the file test.html if it exists in the main server directory.



```
HTTP/1.1 200 OK

Server Name: test Server
Connection:  Running
Content-Type: file type is Html
Content-Length: 361
Date : Sun Feb 25 16:01:20 CET 2018
<html>
<head>
    <title>HTTP/1.1 200 OK</title>
</head>
<body>
<h1>Hello world </h1>
<p>This file is for testing</p>

<form method="POST" action="upload" enctype="multipart/form-data">
    <strong>Upload file:</strong>
    <input type="file" name="file" />
    <input type="submit" style="float:right;" value="upload" />
</form>

</body>
</html>
```

Figure 2.2: What the server return to the browser after a GET request of an html page



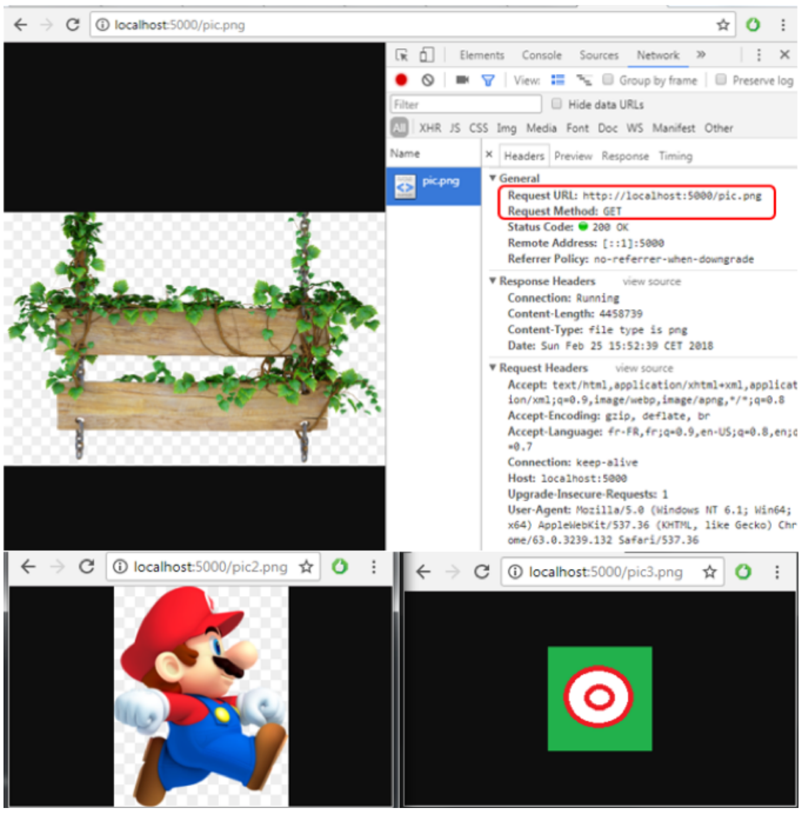
Hello world

This file is for testing

```
▼ General
   Request URL: http://localhost:5000/test.html
   Request Method: GET
   Status Code: ● 200 OK
   Remote Address: [::1]:5000
   Referrer Policy: no-referrer-when-downgrade
▼ Response Headers     view source
   Connection: Running
   Content-Length: 147
   Content-Type: file type is Html
   Date: Sun Feb 25 23:49:31 CET 2018
▼ Request Headers     view source
   Accept: text/html,application/xhtml+xml,applicatio
   e/apng,*/*;q=0.8
   Accept-Encoding: gzip, deflate, br
   Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.
   Connection: keep-alive
   Host: localhost:5000
   Referer: http://localhost:5000/
   Upgrade-Insecure-Requests: 1
   User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64
   L, like Gecko) Chrome/63.0.3239.132 Safari/537.36
```

Figure 2.3: The server also accept GET request of a .png file. It's requested with the form l̈ocalhost:5000/pic.pngf̈or example. If the file exists on the server, it will be shown on the browse

The server also accepts GET request of a .png file. It's requested with the form "localhost:5000/pic.png" for example. If the file exists on the server, it will be shown on the browser.



```
▼ General
    Request URL: http://localhost:5000/pic.png
    Request Method: GET
    Status Code: 200 OK
    Remote Address: [::1]:5000
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Connection: Running
    Content-Length: 4458739
    Content-Type: file type is png
    Date: Sun Feb 25 15:52:39 CET 2018
▼ Request Headers      view source
    Accept: text/html,application/xhtml+xml,applicat
    ion/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
    Accept-Encoding: gzip, deflate, br
    Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q
    =0.7
    Connection: keep-alive
    Host: localhost:5000
    Upgrade-Insecure-Requests: 1
    User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64;
    x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr
    ome/63.0.3239.132 Safari/537.36
```

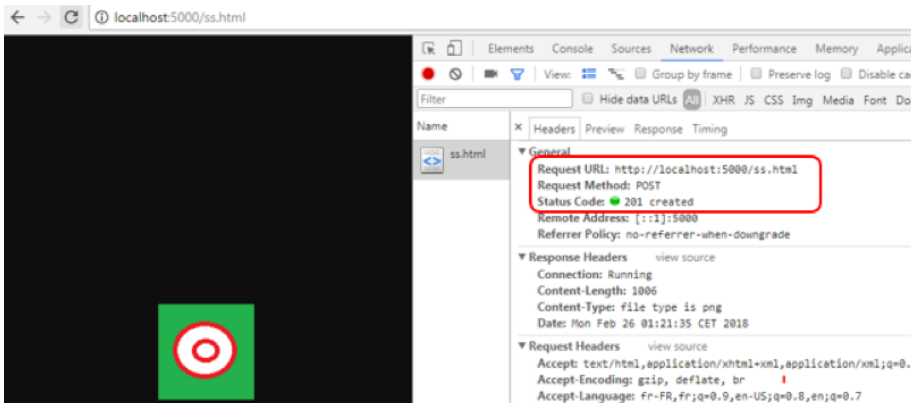
Figure 2.4: Results on the browser after a GET request of different images (low and high quality)

3 Problem Two

We implemented different server responses for several error occurrence due to the request or the server. The following status code have been implemented: 200, 201, 302, 402, 403, 404 ,415 and 500.

3.1 Implementation of 201 Created status code

This code is to be printed when the client use the POST method to upload an image format to the server. The header status will be set to 201 which means that the file is created on the server.

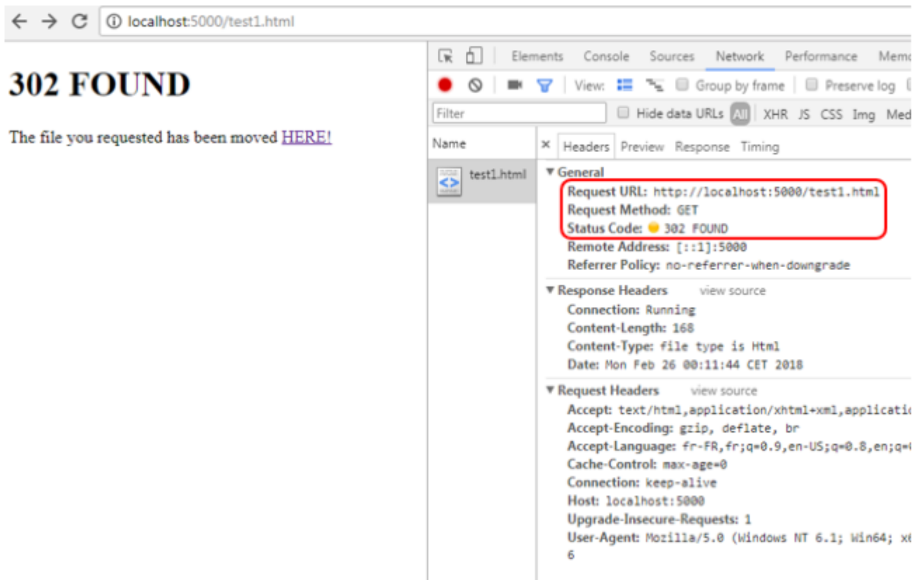


```
▼ General
    Request URL: http://localhost:5000/ss.html
    Request Method: POST
    Status Code: 201 created
    Remote Address: [::1]:5000
    Referrer Policy: no-referrer-when-downgrade
▼ Response Headers      view source
    Connection: Running
    Content-Length: 1006
    Content-Type: file type is png
    Date: Mon Feb 26 01:21:35 CET 2018
▼ Request Headers      view source
    Accept: text/html,application/xhtml+xml,application/xml;q=0.
    Accept-Encoding: gzip, deflate, br
    Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
```

Figure 3.5: Image successfully created after POST request from client

3.2 Implementation of 302 Found status code:

This status code represent a request for a file that has been temporarily removed on the URL given in the message that is printed on the screen.



localhost:5000/test1.html

302 FOUND

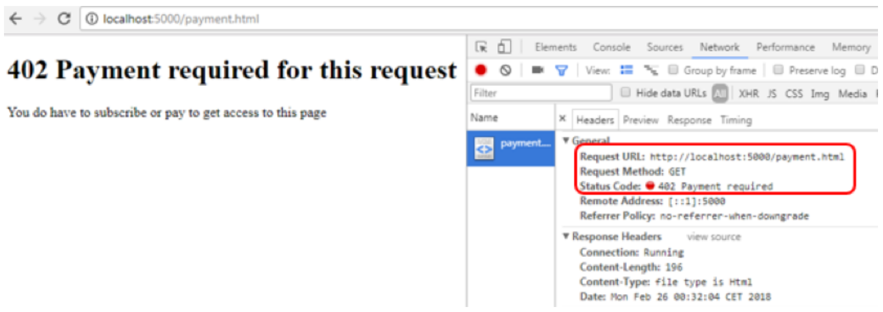
The file you requested has been moved HERE!

```
General
  Request URL: http://localhost:5000/test1.html
  Request Method: GET
  Status Code: 302 FOUND
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
Response Headers
  Connection: Running
  Content-Length: 168
  Content-Type: file type is Html
  Date: Mon Feb 26 00:11:44 CET 2018
Request Headers
  Accept: text/html,application/xhtml+xml,applicati
  Accept-Encoding: gzip, deflate, br
  Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=
  Cache-Control: max-age=0
  Connection: keep-alive
  Host: localhost:5000
  Upgrade-Insecure-Requests: 1
  User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x
  6
```

Figure 3.6: Client trying to access the temporarily moved file, providing the client a link to the new one

3.3 Implementation of "402 Payment Required" status code:

If the client try to access the html file "payment.html" for which you have to pay to have access to, it returns a page with the code "402 Payment Required" informing the client that the file isn't for free.



localhost:5000/payment.html

402 Payment required for this request

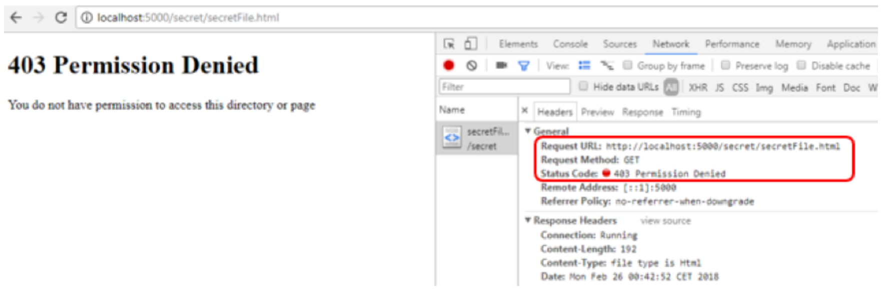
You do have to subscribe or pay to get access to this page

```
General
  Request URL: http://localhost:5000/payment.html
  Request Method: GET
  Status Code: 402 Payment required
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
Response Headers
  Connection: Running
  Content-Length: 196
  Content-Type: file type is Html
  Date: Mon Feb 26 00:32:04 CET 2018
```

Figure 3.7: Client trying to access a paid file

3.4 Implementation of "403 Forbidden" status code:

This error response pop up when the client tries to access an non-accessible file. Here the server tells the client that the request is understood but unauthorised to perform.



403 Permission Denied

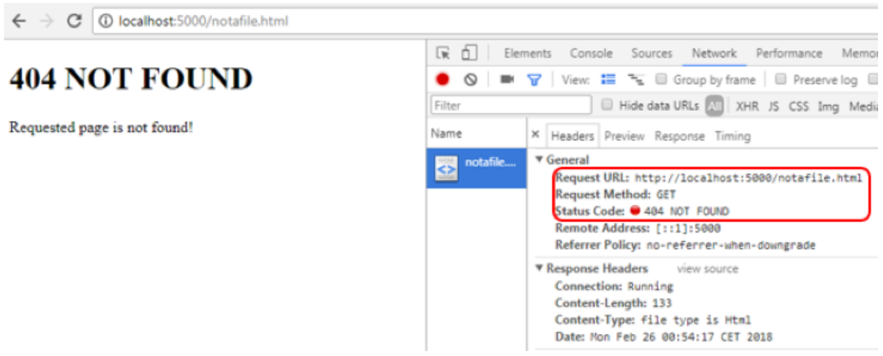
You do not have permission to access this directory or page

```
▼ General
  Request URL: http://localhost:5000/secret/secretFile.html
  Request Method: GET
  Status Code: ● 403 Permission Denied
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
▼ Response Headers    view source
  Connection: Running
  Content-Length: 192
  Content-Type: file type is Html
  Date: Mon Feb 26 00:42:52 CET 2018
```

Figure 3.8: Client trying to access a forbidden resource

3.5 Implementation of "404 Not Found" status code:

This error will be thrown when the server cannot find the requested resource of the client.



404 NOT FOUND

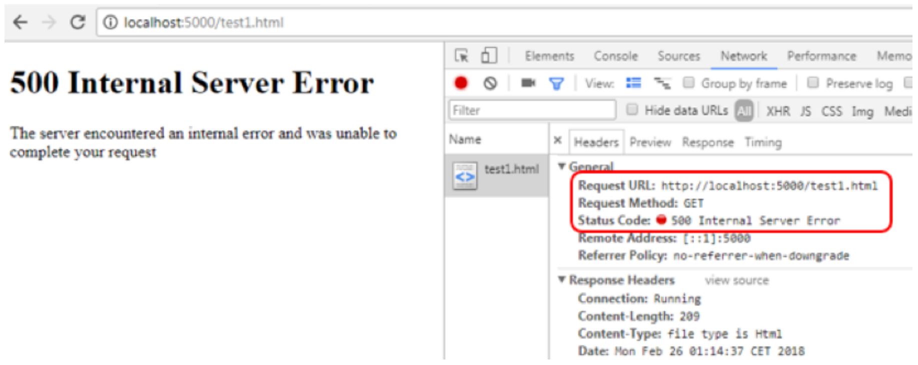
Requested page is not found!

```
▼ General
  Request URL: http://localhost:5000/notafile.html
  Request Method: GET
  Status Code: ● 404 NOT FOUND
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
▼ Response Headers    view source
  Connection: Running
  Content-Length: 133
  Content-Type: file type is Html
  Date: Mon Feb 26 00:54:17 CET 2018
```

Figure 3.9: The file cannot be found so "404 Not Found" status code is thrown

3.6 Implementation of "500 Internal Server Error" status code:

When the server face an unexpected error that prevents it from answering the request, this error response will be showed on the screen.



500 Internal Server Error

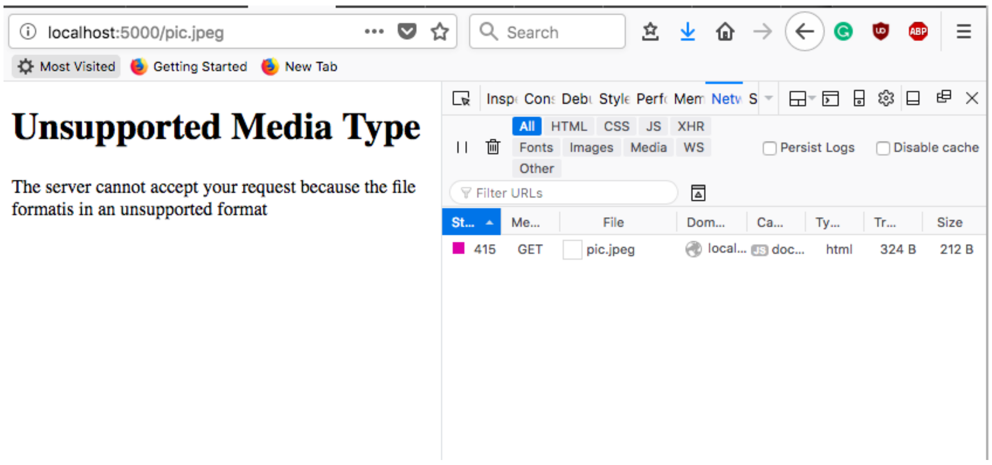
The server encountered an internal error and was unable to complete your request

```
▼ General
  Request URL: http://localhost:5000/test1.html
  Request Method: GET
  Status Code: ● 500 Internal Server Error
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
▼ Response Headers    view source
  Connection: Running
  Content-Length: 209
  Content-Type: file type is Html
  Date: Mon Feb 26 01:14:37 CET 2018
```

Figure 3.10: Server throws an internal error response

3.7 Implementation of 415 Unsupported Media Type:

This error occurs when ever the client tries to access media of format that the server does not support. e.g. client try to load a jpeg picture on the server.



localhost:5000/pic.jpeg

Unsupported Media Type

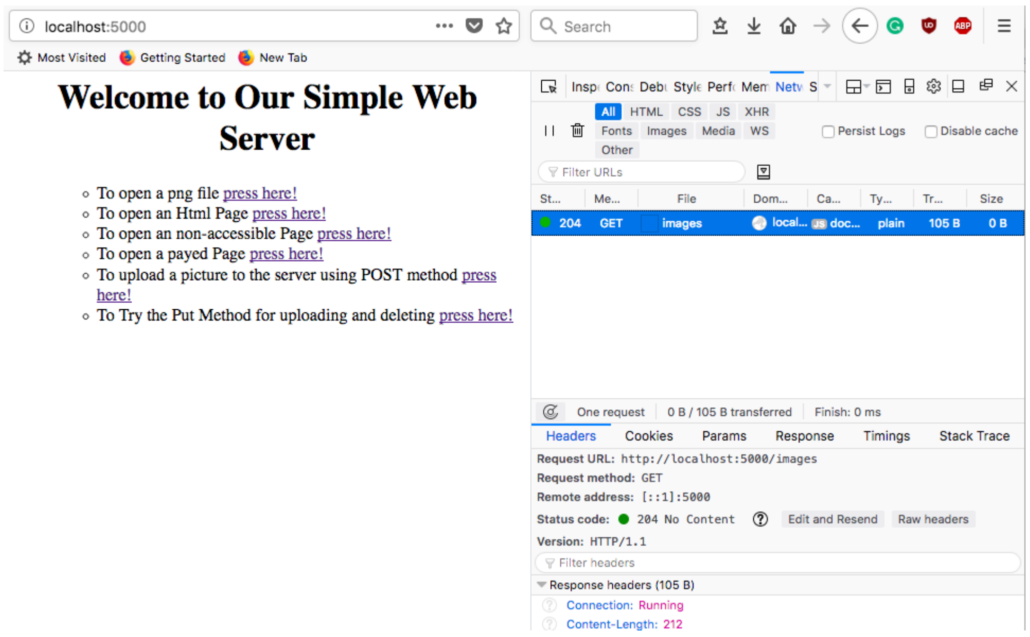
The server cannot accept your request because the file formatis in an unsupported format

St...	Me...	File	Dom...	Ca...	Ty...	Tr...	Size
415	GET	pic.jpeg	local...	doc...	html	324 B	212 B

Figure 3.11: File pic.png is not supported and thus 415 error response is shown

3.8 Implementation of 204 No Content:

The server will inform the client that the requested resource has no content with error response 204.
This means that the server successfully respond to the request but it will not change the document view from where the request was made.



localhost:5000

Welcome to Our Simple Web Server

- To open a png file press here!
- To open an Html Page press here!
- To open an non-accessible Page press here!
- To open a payed Page press here!
- To upload a picture to the server using POST method press here!
- To Try the Put Method for uploading and deleting press here!

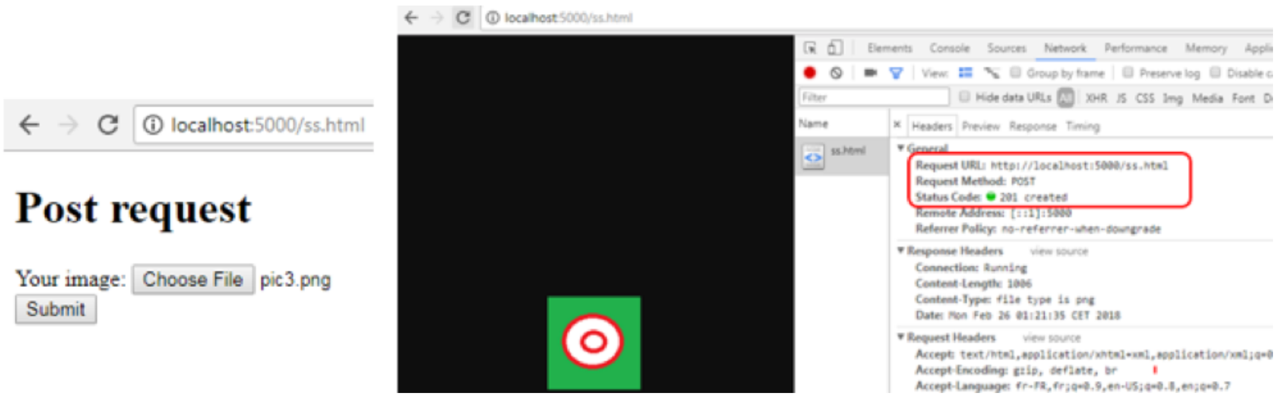
St...	Me...	File	Dom...	Ca...	Ty...	Tr...	Size
204	GET	images	local...	doc...	plain	105 B	0 B

```
One request    0 B / 105 B transferred    Finish: 0 ms
Request URL: http://localhost:5000/images
Request method: GET
Remote address: [::1]:5000
Status code: 204 No Content
Version: HTTP/1.1
Response headers (105 B)
Connection: Running
Content-Length: 212
```

Figure 3.12: the request resource has no content so the path is not changed but the response header is 204

3.9 VG-task 1:

We created an html file containing a JavaScript POST form where the client can choose a file from his computer and send it to the server using this form. The file is first encoded in base64 before being sent. The name of the file is also sent in the request. After receiving the request, the server decode the data, get the file name and create the file accordingly in a specific folder(subdir).



localhost:5000/ss.html
Post request
Your image: Choose File pic3.png
Submit
Elements Console Sources Network Performance Memory
Group by frame Preserve log
Hide data URLs All XHR JS CSS Img Media Font
Name
ss.html
Headers Preview Response Timing

```
▼ General
  Request URL: http://localhost:5000/ss.html
  Request Method: POST
  Status Code: ● 201 created
  Remote Address: [::1]:5000
  Referrer Policy: no-referrer-when-downgrade
▼ Response Headers    view source
  Connection: Running
  Content-Length: 1006
  Content-Type: file type is png
  Date: Mon Feb 26 01:21:35 CET 2018
▼ Request Headers    view source
  Accept: text/html,application/xhtml+xml,application/xml;q=0
  Accept-Encoding: gzip, deflate, br
  Accept-Language: fr-FR,fr;q=0.9,en-US;q=0.8,en;q=0.7
```

Figure 3.13: Client choose to send the file "pic3.png", then getting the picture as a result with the "201 Created" status code meaning it has been successfully created on the server

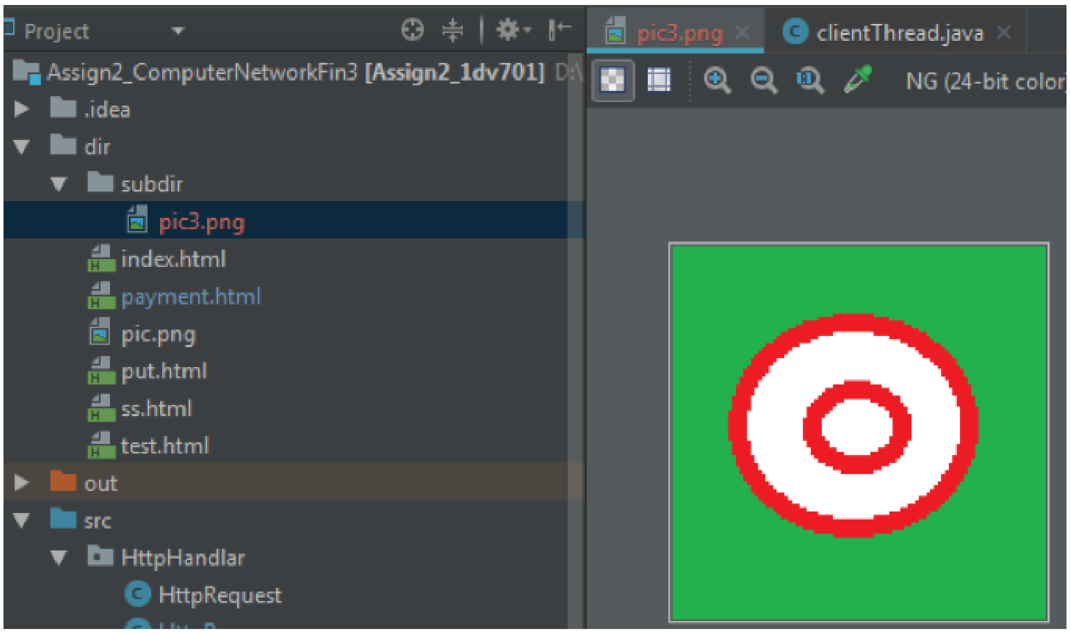
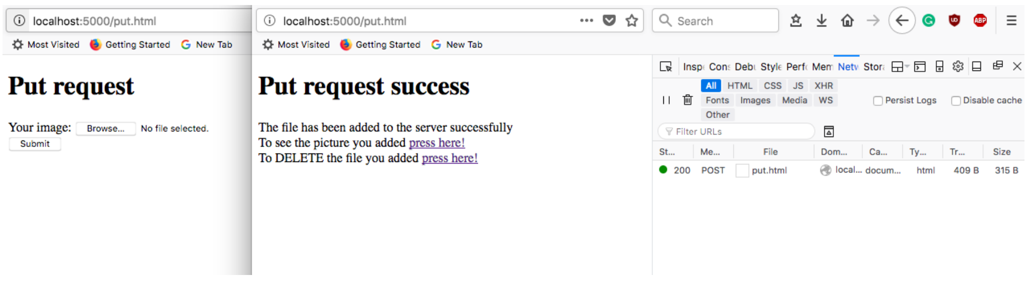


Figure 3.14: The image has been created in the folder subdir

3.10 VG-task 2:

This task has been made in the same way the first VG-task was made. The client can upload an image to the server by using a put method or delete an existing one. The idea of PUT method is that the client cannot add the same image twice or replace the one already exist.
First the client will upload an image to the server and then the it will become a POST method after uploading it(see figure 3.15).

The client now can delete it as it is exist and this is performed by PUT method as well (see figure 3.16).
If the client tries to upload the same image, the request will be canceled as a PUT-method cannot be performed twice(See figure 3.17).


localhost:5000/put.html

Put request

Your image: Browse... No file selected.
Submit

localhost:5000/put.html

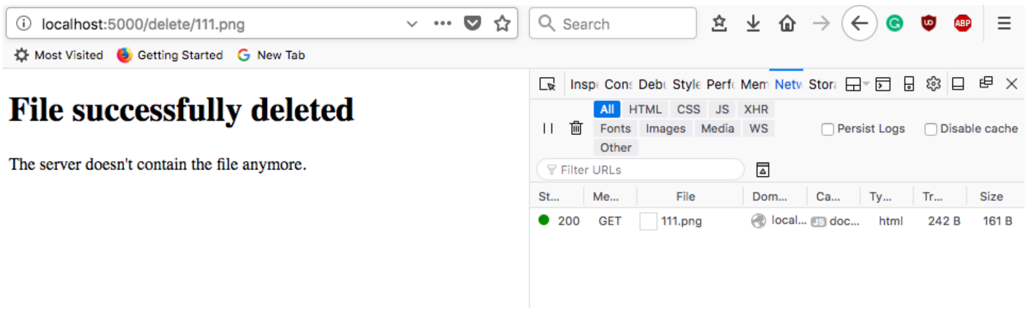
Put request success

The file has been added to the server successfully
To see the picture you added press here!
To DELETE the file you added press here!

St...	Me...	File	Dom...	Ca...	Ty...	Tr...	Size
● 200	POST	put.html	local...	docum...	html	409 B	315 B

Most Visited
Getting Started
New Tab
Search
Insp Con Debi Style Perf Mem Netw Stor
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Filter URLs

Figure 3.15: The image has been successfully added to the server via PUT-method


localhost:5000/delete/111.png

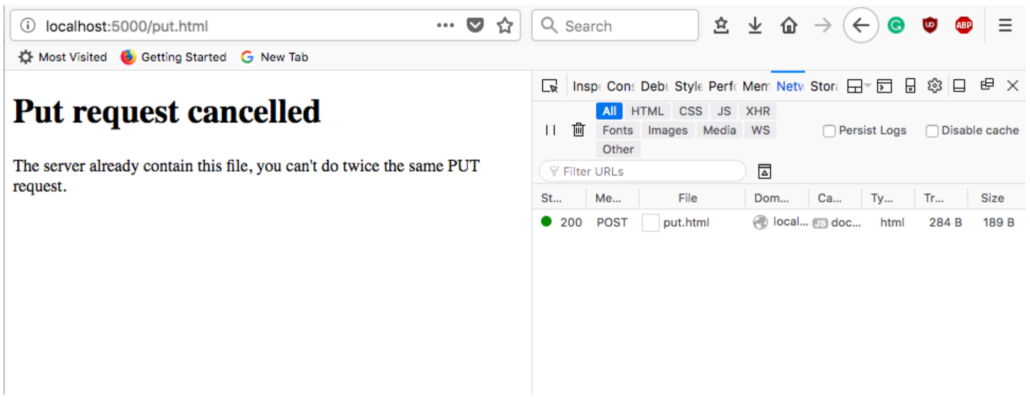
File successfully deleted

The server doesn't contain the file anymore.

St...	Me...	File	Dom...	Ca...	Ty...	Tr...	Size
● 200	GET	111.png	local...	doc...	html	242 B	161 B

Most Visited
Getting Started
New Tab
Search
Insp Con Debi Style Perf Mem Netw Stor
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Filter URLs

Figure 3.16: Using put method to delete an image from the server


localhost:5000/put.html

Put request cancelled

The server already contain this file, you can't do twice the same PUT request.

St...	Me...	File	Dom...	Ca...	Ty...	Tr...	Size
● 200	POST	put.html	local...	doc...	html	284 B	189 B

Most Visited
Getting Started
New Tab
Search
Insp Con Debi Style Perf Mem Netw Stor
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
Filter URLs

Figure 3.17: The client tried to add the same image again. in PUT method this is invalid request

The difference between POST and PUT:

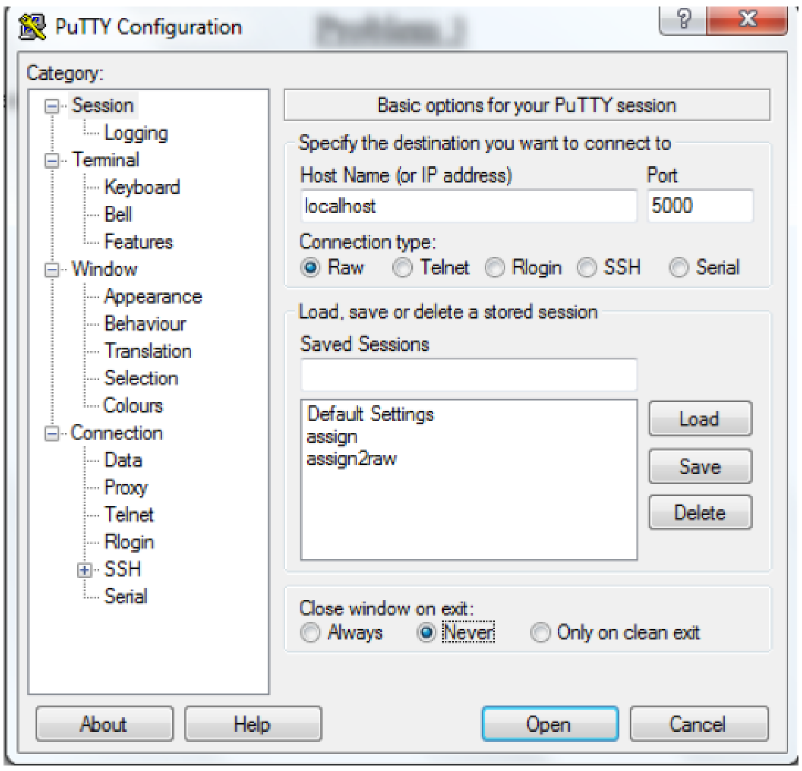
POST

Post method is basically used to create. Post method is not sensitive for the location or the url of the requested method. So performing a post method to upload the same image will led to duplicate the image on the server in different locations.

PUT put method is generally used for updating more than creating because it works on replacing what ever was exist in the target path. So either creating or updating it will do the same. So in other word, to perform a Put method the path or the url should be indicated and the method should use this specific url to overwrite or create the requested resource. Every time we upload an image with Put method it will always be in the same requested location.

4 Problem Three:

We download and installed PuTTY v0.7 on Windows 7. Following the tips from the lab tutor (Michael Johansson) we used the connection type "Raw" instead of "Telnet" because the latter adds unnecessary characters


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): localhost

Port: 5000

Connection type: [✓ Raw] Telnet Rlogin SSH Serial

Load, save or delete a stored session

Saved Sessions

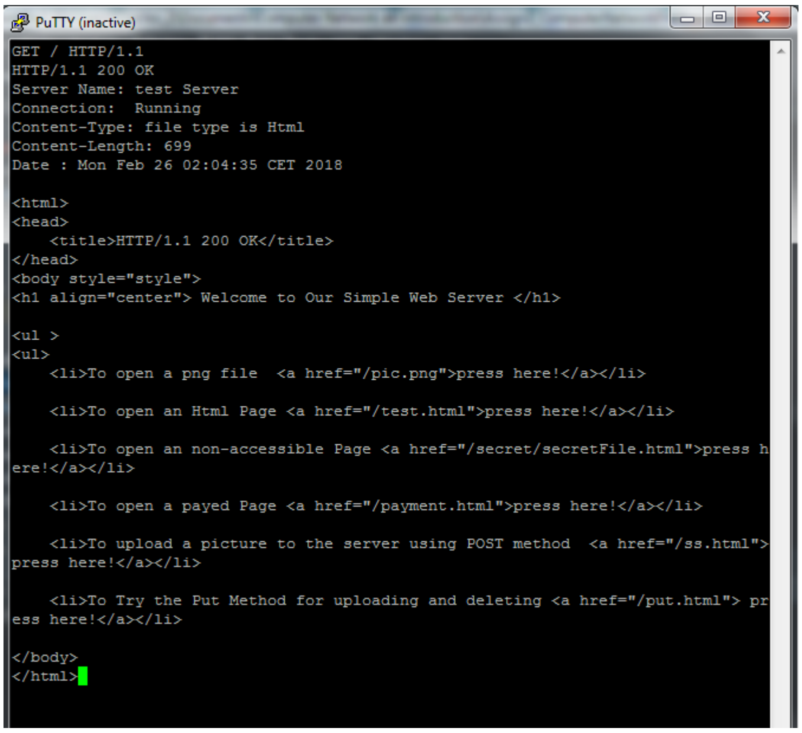
Default Settings
assign
assign2raw

Close window on exit: Always [✓ Never] Only on clean exit

Figure 4.18: PuTTY configuration to connect to localhost on port 5000

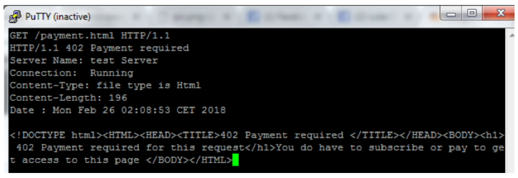
Starting the connection open the terminal and wait for input commands. To enter the index.html of the server's main folder, we type "GET / HTTP/1.1". "GET" specifies that this is a GET request. The slash indicate the root which will redirect to the index.html. "HTTP/1.1" specifies the version of HTTP to use for this request.

The non-sense shown by the terminal(figure 4.22) is PuTTY interpreting the image binary in a plain text and this explains these un understandable charachteres.



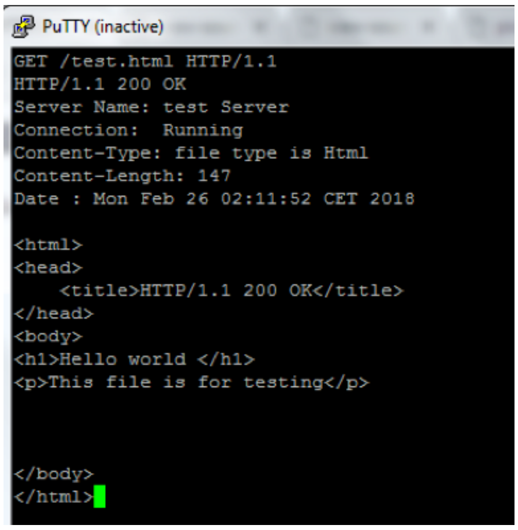
```
GET / HTTP/1.1
HTTP/1.1 200 OK
Server Name: test Server
Connection:  Running
Content-Type: file type is Html
Content-Length: 699
Date : Mon Feb 26 02:04:35 CET 2018

<html>
<head>
    <title>HTTP/1.1 200 OK</title>
</head>
<body style="style">
<h1 align="center"> Welcome to Our Simple Web Server </h1>

<ul >
<ul>
    <li>To open a png file  <a href="/pic.png">press here!</a></li>

    <li>To open an Html Page <a href="/test.html">press here!</a></li>

    <li>To open an non-accessible Page <a href="/secret/secretFile.html">press h
ere!</a></li>

    <li>To open a payed Page <a href="/payment.html">press here!</a></li>

    <li>To upload a picture to the server using POST method  <a href="/ss.html">
press here!</a></li>

    <li>To Try the Put Method for uploading and deleting <a href="/put.html"> pr
ess here!</a></li>

</body>
</html>
```

Figure 4.19: The terminal prints the whole html code returned by the server that is the html page for index.html



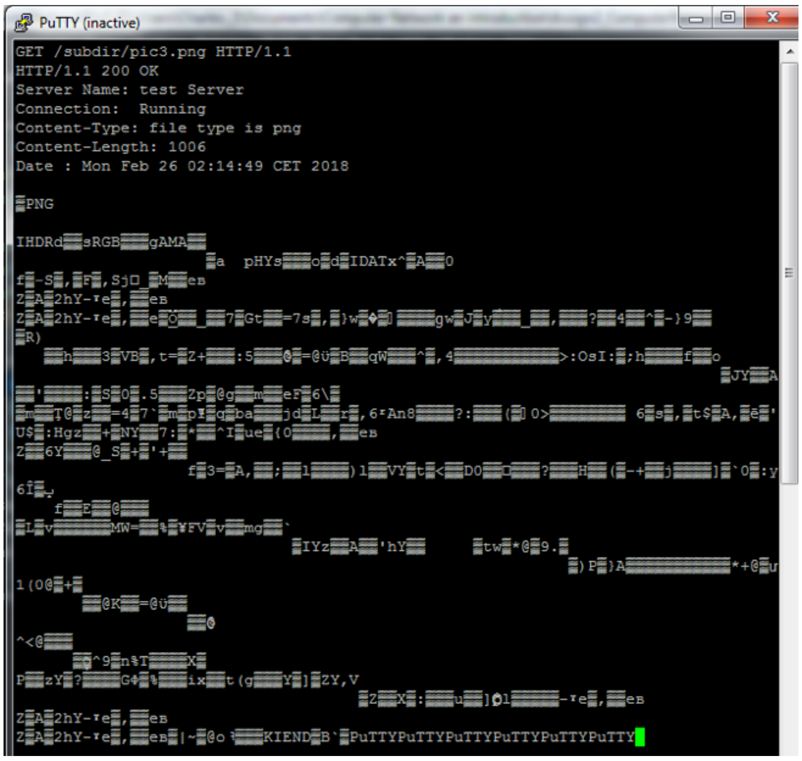
```
GET /payment.html HTTP/1.1
HTTP/1.1 402 Payment required
Server Name: test Server
Connection:  Running
Content-Type: file type is Html
Content-Length: 196
Date : Mon Feb 26 02:08:53 CET 2018

<!DOCTYPE html><HTML><HEAD><TITLE>402 Payment required </TITLE></HEAD><BODY><h1>
 402 Payment required for this request</h1>You do have to subscribe or pay to ge
t access to this page </BODY></HTML>
```

Figure 4.20: When GET-requesting the paid file "payment.html", the terminal shows the page's code informing of the "402 Payment Required" status code



```
GET /test.html HTTP/1.1
HTTP/1.1 200 OK
Server Name: test Server
Connection:  Running
Content-Type: file type is Html
Content-Length: 147
Date : Mon Feb 26 02:11:52 CET 2018

<html>
<head>
    <title>HTTP/1.1 200 OK</title>
</head>
<body>
<h1>Hello world </h1>
<p>This file is for testing</p>



</body>
</html>
```

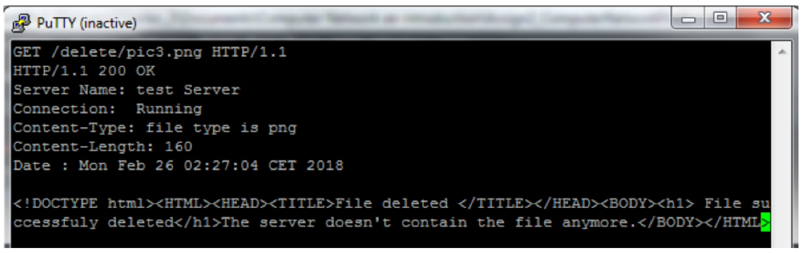
Figure 4.21: "test.html" file content returned after a GET request



```
GET /subdir/pic3.png HTTP/1.1
HTTP/1.1 200 OK
Server Name: test Server
Connection:  Running
Content-Type: file type is png
Content-Length: 1006
Date : Mon Feb 26 02:14:49 CET 2018

▒PNG

IHDRd▒▒sRGB▒▒▒gAMA▒▒
                         ▒a  pHYs▒▒▒o▒d▒IDATx^▒A▒▒0
f▒-S▒,▒F▒,Sj□_▒M▒▒eв
Z▒A▒2hY-ᴦe▒,▒▒eв
Z▒A▒2hY-ᴦe▒,▒▒e▒O▒▒_▒▒7▒Gt▒▒=7s▒,▒}w▒♦▒] ▒▒▒▒gw▒J▒y▒▒▒_▒▒,▒▒▒?▒▒4▒▒^▒-}9▒▒
▒R)
[illegible]
```

```
Z▒A▒2hY-ᴦe▒,▒▒eв
Z▒A▒2hY-ᴦe▒,▒▒eв▒|~▒@o ▒▒▒KIEND▒B`▒PuTTYPuTTYPuTTYPuTTYPuTTYPuTTY
```

Figure 4.22: PuTTY printing the response from the browser after requesting a png



```
GET /delete/pic3.png HTTP/1.1
HTTP/1.1 200 OK
Server Name: test Server
Connection:  Running
Content-Type: file type is png
Content-Length: 160
Date : Mon Feb 26 02:27:04 CET 2018

<!DOCTYPE html><HTML><HEAD><TITLE>File deleted </TITLE></HEAD><BODY><h1> File su
ccessfuly deleted</h1>The server doesn't contain the file anymore.</BODY></HTML>
```

Figure 4.23: PuTTY showing the server's response after requesting the removal of the file "pic3.png"