Egill Anton Hlööversson, egilla14, kt. 190193-3309
Edda Steinunn Rúnarsdóttir, eddasr15, kt. 241095-2909

Class Assignment II

PART 1: TELNET

1. Connect to http://example.com

```
$ telnet example.com 80
```

1.1. Construct a basic HTTP request

```
GET / HTTP/1.1
Host: example.com
```

1.2. Inspect the response

Server sends back a HTTP response with status code 200 (OK) indicating success. The response has various headers including those intended for cache control, then specifies content type as HTML and the request body contains HTML content in accordance.

2. Connect to https://veft-http.herokuapp.com

```
$ telnet veft-http.herokuapp.com 80
```

2.1. Start by constructing a basic HTTP request for the root (/tasks)

```
GET /tasks HTTP/1.1
Connection: keep-alive
Host: veft-http.herokuapp.com
```

2.2. Go further down and construct a HTTP request to /tasks/in_progress

```
GET /tasks/in_progress HTTP/1.1
Host: veft-http.herokuapp.com
```

2.3. Use a query parameter, named assignedTo, to /tasks/{name-of-list} to filter out only tasks assigned to a specific person

```
GET /tasks/done?assignedTo=Mom HTTP/1.1
Host: veft-http.herokuapp.com
```

2.4. Issue the same request as above, but with the unsafe characters as value of assignedTo

```
GET /tasks/done?assignedTo=Edda Steinunn HTTP/1.1
Host: veft-http.herokuapp.com
```

(Note that this request of course does not work as a valid HTTP request due to the HTTP request parser on the server side inferring that whatever comes after the URL (white space seperated) is the HTTP version according to the HTTP RFC standard and of course "Steinunn" is not a valid HTTP version).

2.5. Issue the same request as above, but now with the unsafe characters properly encoded

```
GET /tasks/done?assignedTo=Edda%20Steinunn HTTP/1.1
Host: veft-http.herokuapp.com
```

2.6. Go further down and get a specific task by id /tasks/in_progress/{task-id}

```
GET /tasks/in_progress/4 HTTP/1.1
Host: veft-http.herokuapp.com
```

2.7. Use content negotiation to retrieve the content from /tasks/in_progress/{task-id} in different formats, e.g. JSON, XML, HTML

```
GET /tasks/in_progress/4 HTTP/1.1
Host: veft-http.herokuapp.com
Accept: text/html
---
GET /tasks/in_progress/4 HTTP/1.1
Host: veft-http.herokuapp.com
Accept: text/xml
---
GET /tasks/in_progress/4 HTTP/1.1
Host: veft-http.herokuapp.com
Accept: application/json
```

2.8. Construct a HTTP request to /tasks/instructions

```
GET /tasks/instructions HTTP/1.1
Host: veft-http.herokuapp.com
```

2.9. Use content negotiation to retrieve the content in different languages, e.g. Icelandic, English, German and Japanese

```
GET /tasks/instructions HTTP/1.1
Host: veft-http.herokuapp.com
Accept-Language: is-IS
---
GET /tasks/instructions HTTP/1.1
Host: veft-http.herokuapp.com
Accept-Language: en-EN
---
GET /tasks/instructions HTTP/1.1
Host: veft-http.herokuapp.com
Accept-Language: ja
---
GET /tasks/instructions HTTP/1.1
Host: veft-http.herokuapp.com
Accept-Language: de-DE
```

2.10. Issue a HTTP POST request to /tasks/{name-of-list} to create a new task. The data should be sent with the request body as JSON. The model includes: name, description and assignedTo

```
POST /tasks/in_progress HTTP/1.1
Host: veft-http.herokuapp.com
Content-Type: application/json; charset=utf-8
Content-Length: 79

{
      "name":"Watch Paint Dry",
      "description":"For Fun",
      "assignedTo":"Edda Steinunn"
}
```

2.11. Take a look at the response headers for the POST request and try following the provided URL

The response server sends to my post request explicitly tells me that the task I just posted is at URL http://veft-http.herokuapp.com/tasks/in_progress/74 via the "Location" header. Indeed this URL responds with the task I just created when I request for this URL;

```
GET /tasks/in_progress/74 HTTP/1.1
Host: veft-http.herokuapp.com
```

2.12. Issue a HTTP PUT request to /tasks/{name-of-list}/{task-id}/assign/{name-ofassignee}

```
PUT /tasks/in_progress/74/assign/Edda%20St. HTTP/1.1
Host: veft-http.herokuapp.com
```

PART 2: CURL

1. Connect to http://example.com

1.1. Construct a basic HTTP request

```
curl -X GET http://example.com
```

1.2. Inspect the response

Server sends back a HTTP response with status code 200 (OK) indicating success. The response has various headers including those intended for cache control, then specifies content type as HTML and the request body contains HTML content in accordance.

2. Connect to https://veft-http.herokuapp.com

2.1. Start by constructing a basic HTTP request for the root (/tasks)

```
curl -X GET https://veft-http.herokuapp.com/tasks
```

2.2. Go further down and construct a HTTP request to /tasks/in_progress

```
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress
```

2.3. Use a query parameter, named assignedTo, to /tasks/{name-of-list} to filter out only tasks assigned to a specific person

```
curl -X GET
https://veft-http.herokuapp.com/tasks/in_progress?assignedTo=Michael
```

2.4. Issue the same request as above, but with the unsafe characters as value of assignedTo

```
curl -X GET
https://veft-http.herokuapp.com/tasks/in_progress?assignedTo=Egill
Anton
```

(Note that this request of course does not work as a valid HTTP request due to the HTTP request parser on the server side inferring that whatever comes after the URL (white space seperated) is the HTTP version according to the HTTP RFC standard and of course "Steinunn" is not a valid HTTP version).

2.5. Issue the same request as above, but now with the unsafe characters properly encoded

```
curl -X GET
https://veft-http.herokuapp.com/tasks/in_progress?assignedTo=Egill%20An
ton
```

2.6. Go further down and get a specific task by id /tasks/in_progress/{task-id}

```
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress/1
```

2.7. Use content negotiation to retrieve the content from /tasks/in_progress/{task-id} in different formats, e.g. JSON, XML, HTML

```
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress/1  -H
"Accept: application/json"
---
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress/1  -H
"Accept: text/xml"
---
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress/1  -H
"Accept: text/html"
```

2.8. Construct a HTTP request to /tasks/instructions

```
curl -X GET https://veft-http.herokuapp.com/tasks/instructions
```

2.9. Use content negotiation to retrieve the content in different languages, e.g. Icelandic, English, German and Japanese

```
curl -X GET https://veft-http.herokuapp.com/tasks/instructions -H
"Accept-Language: is-IS"
---
curl -X GET https://veft-http.herokuapp.com/tasks/instructions -H
"Accept-Language: ja"
---
curl -X GET https://veft-http.herokuapp.com/tasks/instructions -H
"Accept-Language: en-EN"
---
curl -X GET https://veft-http.herokuapp.com/tasks/instructions -H
"Accept-Language: de-DE"
```

2.10. Issue a HTTP POST request to /tasks/{name-of-list} to create a new task. The data should be sent with the request body as JSON. The model includes: name, description and assignedTo

```
curl -X POST https://veft-http.herokuapp.com/tasks/in_progress -H
"Content-Type: application/json" -d "{\"name\":\"Work\",
\"description\":\"Lorem Ipsum\", \"assignedTo\":\"Egill\"}"
```

2.11. Take a look at the response headers for the POST request and try following the provided URL

The response server sends to my post request explicitly tells me that the task I just posted is at URL https://veft-http.herokuapp.com/tasks/in_progress/61 via the "Location" header. Indeed this URL responds with the task I just created when I request for this URL;

```
curl -X GET https://veft-http.herokuapp.com/tasks/in_progress/61
```

12. Issue a HTTP PUT request to /tasks/{name-of-list}/{task-id}/assign/{name-ofassignee}

```
PUT /tasks/in_progress/74/assign/Edda%20St. HTTP/1.1
Host: veft-http.herokuapp.com
```

PART 3: POSTMAN

See solution for part 3, Postman, in the form of screenshots under /part3_postman. Each screenshot is named after corresponding problem part identifier, e.g. problem 2.2. Is named 2.2.

PART 4: FIDDLER / CHARLES

See solution for part 4 in the form of screenshots under /part4_fiddler. Each screenshot is named after corresponding problem part identifier, e.g. problem 2.7. Is named 2.7.
NOTE: For this part we used Fiddler over Charles