Lab assignment 1

TI1506 2017/18; ti1506-students.slack.com

Introduction

The first part of this assignment gives you hands-on experience in http. In the second part you will make a head-start with the design of your **Habit Tracker Web application**.

Remember that this is a group assignment! Work efficiently as a team!

Deadline: when this assignment is due depends on the cluster your team is in. Every team is assessed bi-weekly. Make sure you know your cluster and what this means for your lab deadlines. All of this information is available in the course syllabus (on Brightspace).

In order to pass the assessment, you have to make a valid attempt at working through the lab assignment.

Advice: **do not copy & paste anything from this PDF directly, please type out the commands yourself**. This makes it more likely that you learn what you need to learn for this class and it also avoids errors that your PDF reader may introduce in the copying process.

1. HTTP REQUEST MESSAGES: GET/HEAD

Hints:

- to store telnet's output to file (in addition to printing it on the console), you can use the command "`tee`". As an example:
  `telnet www.microsoft.com 80|tee outfile`
  will save all output to the file `outfile`.

- [carriage return] is a place holder for pressing "Enter"

This exercise requires you to use **telnet**. If you use a Linux derivative (e.g. Ubuntu, Mac OS), open a terminal and you are good to go; if you want to use Windows you will have to install a telnet client such as PuTTY.

Use telnet to request the contents of the bread section of the recipe website `myrecipes.com/bread-recipes`.

Start your "conversation" with the Web server with

```
$telnet myrecipes.com 80
```

1.1) **Write down** the HTTP requests you made, the returned responses (e.g. a page has moved or is faulty) until you receive the contents of the recipes Web page. Always use `HEAD` first to retrieve meta-data about the resource.

1.2) Does the content correspond to what you see when accessing the page with your browser? To check, save the response to a file, use "html" as file ending and open it with your browser.

1.3) What is the purpose of the `ETag` in the header information?

1.4) What do the different `Cache-Control` directives mean?

2. HTTP REQUEST MESSAGE: PUT

While GET and HEAD are request methods accepted by virtually all Web servers, PUT/POST/DELETE are less often available, due to the implications these methods have on the server.

To test your skills in uploading, deleting and posting data, we will make use of http://httpbin.org/, a service designed to test HTTP messages.

Below is an example of how to upload data to the server with PUT:

```
$telnet httpbin.org 80

PUT /put HTTP/1.1
host:httpbin.org
Content-type:text/plain
Content-length:12
[carriage return]
[carriage return]
Hello World!
[carriage return]
```

With this code, we have just created a file on the server called "put" which contains the string "Hello World!". The service sends back in the response the data just uploaded - the response is of content-type json; we are interested in the "data" field, which should contain "Hello World!" if everything worked correctly. Try it out for yourself!

2.1) The content-length is exactly the number of characters (12) of “Hello World!”. What happens if the `Content-length` field is smaller or larger than the exact number of characters in the content?

3. BASIC AUTHENTICATION

Lets now try to request a page, which is set up with HTTP basic authentication.

3.1) First, open http://httpbin.org/basic-auth/user/passwd in your browser. You should see a dialog, requesting username and password. Use "user" as username and "passwd" as password. Reload the Web page - what happens now?

3.2) Now let's see how this works with actual HTTP messages. Start off with a HEAD method to inspect the Web page and document all following steps (requests and responses):

```
$telnet httpbin.org 80
HEAD /basic-auth/user/passwd HTTP/1.1
host:httpbin.org
[carriage return]
[carriage return]
```

Then, use the Authorization field to provide username and password to the server. To encode the username and password, you can use any of the freely available base-64 en/decoders, e.g. http://decodebase64.com . Remember that username and password should be combined as username:password.

3.3) Now close the TCP connection and start a new one, using again `telnet httpbin.org 80`. Request the same page - what happens? Is the behavior the same as reloading the page in the browser?

4. HABIT TRACKER APPLICATIONS

A general note: The Web course book develops a TODO list Web application throughout the different chapters. The assignments in this course focus on the development a different but similar application. Since a similar project is developed in the book and in class, if you find some aspects of the assignments very difficult, the book should help you along.

Habit trackers are applications that allow you to set and track your habits---the good as well as the bad ones. Habit trackers either rely on self-reports or use data from sensors (FitBit, etc.). Many habit trackers are only available as native apps (Android, iOS). In this class project we focus on building a **Web-based habit tracker** application instead that relies on **self-reports**.

In this assignment you will start developing your own habit tracker application by first considering existing applications and then designing your own.

4.1) Select two habit tracker applications, either from the following list or of your own choice (they should be **accessible from the Web**, without installing a native app):

- https://www.stridesapp.com/
- https://habitica.com/
- https://chains.cc
- http://www.irunurun.com/
- http://www.joesgoals.com/
- https://everydaycheck.com/

4.2) Use the application's standard interface. Consider their design based on the Web design principles covered in class. Which **2 design aspects** stand out in each of the two applications (in the **positive or negative** sense)?

5. USABILITY TEST

5.1) For each of the two chosen applications, perform a usability test yourself by performing the following actions: (1) create a habit (e.g. "early morning yoga") that should happen every weekday (not at the weekend) and (2) a habit that should happen once a week (e.g. "go swimming"); (3) self-report on both habits; (4) check the progress on how well you are doing; (5) delete one habit. **Report** on which habit tracker you performed the tests on. **Report** how much time and how many clicks it took you to perform the actions.

6. HABIT TRACKER LIST FEATURES

6.1) Create a list of 10 habit tracker features that you think a habit tracker Web application should have[1]. To get you started, here are two essential features: (i) create a habit to track, and, (iii) delete a habit.

7. YOUR OWN HABIT TRACKER: DESIGN PHASE

Similar to the wireframe example in the course book (read Chapter 2 if you have not done so yet), start designing your own habit tracker application. You will develop this application throughout the course. Take inspirations from the habit trackers you looked at. Your Web application should be designed for the standard Desktop interface (i.e. not mobile). Before you start, decide on your target audience.

7.1) Create a design for the entry page (splash screen[2]): think of a name for your application, a short description & a logo. Feel free to use images with Creative Commons license.

7.2) Create a design for the habit tracker page; allow at least the following: (i) add/delete a habit, (ii) label a habit as either a positive (a habit you want to build up) or a negative (a habit you want to get rid off) one and (iii) visually track the habits progress over time; add another two features to your design that you think are useful (take inspiration from 3.).

Note: the wireframes can be done with pen and paper or a software tool (e.g. Gliffy, Balsamiq, Moqups or even Photoshop) of your choice.

7.3) Once you have completed the design of your app, head over to **TI1506's Brightspace**, go to "Discussions" and then the forum **HABIT TRACKER APP DESIGNS.** Create a **thread**

[1] Feel free to take inspirations from native app habit trackers or from offline (i.e. pen-and-paper) habit trackers.
[2] An example of a splash screen is given in Figure 2-10 (p. 40) of the Web course book.

with your team's name as subject/title (e.g. "Minor 01" or "TI-CF 12") and post your team's proposed design. Include in the post the following:

- the two designs (entry page and habit tracker page);
- who you envision your target audience to be;
- a paragraph on the reasoning behind your design and the design goals.

8. YOUR OWN HABIT TRACKER APP: HTML

8.1) Similar to the course book, take your design as a starting point and create the respective two HTML documents (note that these documents should only contain HTML, **no CSS or JavaScript**). Note: if you already want to start writing CSS/JS, be aware that during the assessment of this assignment, this additional work is ignored.