Lab assignment 1

TI1506 2016/17; 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 TODO 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 Blackboard).

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. Virtual Machine

Install the virtual machine. It contains all the necessary programs to conduct the assignments in this class. We use node.js and MySQL as our main tools – if you feel comfortable installing those tools yourself, feel free to do so instead.

Instructions of how to install the virtual machine are in the syllabus.

2. 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 the virtual machine, 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 sport section of `www.nu.nl` (`www.nu.nl/sport`). Start your "conversation" with the Web server with

```
$telnet www.nu.nl 80
```

2.1) 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 sport Web page. Always use `HEAD` first to retrieve meta-data about the resource.

2.2) 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.

2.3) When does the content expire according to the HTTP header (and what does it mean in practice)?

2.4) What is the HTTP response if the `If-Modified-Since` field is used and set before/after the `Last-Modified` value shown?

2.5) What happens if you start the conversation with `$telnet nu.nl 80` instead of `$telnet www.nu.nl 80`?

---

3. 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!

3.1) 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?

4. BASIC AUTHENTICATION

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

4.1) 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?

4.2) 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.

4.3) 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?

5. SURVEY TODO APPLICATIONS

A general note: The Web course book develops a TODO list Web application throughout the different chapters. The assignments follow this idea; though implemented features are often different from those in the book. Ideally, since a very similar project is developed in the book and in class, if you find some aspects of the assignments very difficult, the book will help you along.

TODO lists are everywhere and many different Web applications have been developed to that effect. In this assignment you will start developing your own TODO application by first considering existing applications and then designing your own. Note, that although of course TODO applications for Windows, Android, etc. exist, in this class we focus on Web-based TODO applications.

5.1) Select two TODO Web applications, either from the following list or select some of your own choice (they should be accessible from the Web, without having to download software):

- http://todoist.com/
- http://www.any.do/
- http://www.rememberthemilk.com/
- https://www.toodledo.com/
- http://todo.ly/

5.2) Use the application's standard interface (NOT the mobile interface). Consider their design based on the Web design principles covered in class. Which **2-3 design aspects** stand out in each of the two applications (in the **positive and/or negative** sense)?

6. USABILITY TEST

6.1) For each of the two chosen applications, perform a small-scale usability test yourself by performing the following action: create a todo list item whose deadline is on November 30, about which you will receive a reminder a day before the actual deadline. **Report** on which

TODO list you performed the tests. **Report** how much time and how many clicks it took you to perform each action. Were the actions intuitive to perform?

7. TODO LIST FEATURES

7.1) Create a list of 10 basic TODO list features that you think every TODO Web application should have. To get you started, here are three essential features: (i) create a todo list item, (ii) sort the todo list according to date, and, (iii) delete a todo list item.

8. YOUR OWN TODO APP: DESIGN PHASE

Similar to the wireframe example in the course book, start designing your own TODO application. You will develop this application throughout the course. Feel free to take inspirations from the TODO applications 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** (e.g. students, grandmas, children …)

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

8.2) Create a design for the todo list page; allow at least the following actions: add/delete an item, add a deadline to an item, add an importance rating to an item (can be a simple binary rating), sort the items according to due date, set a todo to done (it is no longer active); 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.

8.3) Once you have completed the design of your app, head over to **TI1506's Blackboard**, go to "Discussions" and then the forum **TODO APP DESIGNS.** Create a **thread** with your team’s name as subject/title (e.g. “Minor 01” or “TI-ACE 12”) and post your team’s proposed design. Include in the post the following:

- the two designs (entry page and todo list page);
- who you envision your target audience to be;
- a paragraph (minimum 100 words) on the reasoning behind your design and the design goals.

9. YOUR OWN TODO APP: HTML

9.1) 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**).