# Web- and Databasetechnology

## Lab assignment 1

## 2 HTTP request messages GET&HEAD

### 2.1

Http requests made to retrieve content of nu.nl/sport:

```
telnet nu.nl 80

HEAD / HTTP/1.1
host:nu.nl
...

telnet www.nu.nl 80

HEAD /sport HTTP/1.1
host:www.nu.nl
...
GET /sport HTTP/1.1
host:www.nu.nl
```

### 2.2

The answer to the `GET` request firstly includes the Header of the HTTP response. The browser cannot interpret this so it needs to be removed from the message. The body of the message is actually the same as the one the browser gets when opnening the website in the browser but all the resources that are existing on the server are not on the local machine and therefore images are not shown when opening the response file in the browser.

### 2.3

This is the response to the `HEAD /sport HTTP/1.1` command:

```
HTTP/1.1 200 OK
Date: Mon, 16 Nov 2015 11:49:36 GMT
Content-Type: text/html; charset=utf-8
Expires: Mon, 16 Nov 2015 11:50:36 GMT
Vary: Accept-Encoding
Last-Modified: Mon, 16 Nov 2015 11:49:36 GMT
```

```
Cache-Control: max-age=60
Age: 32
Connection: keep-alive
Via: 1.1 sanin507.ixa
```

From the Expires and Last-Modified date we see that the Expires date is exactly one minute later than the Last-Modified date.

### 2.4

When doing another `HEAD` request before the expiration date, the exact same message is replied. When doing the request after the expiration date, a new message is created with updated `Last-Modified`, `Date` and `Expires` fileds.

## 3 HTTP request message PUT

### 3.1

When sending `PUT` messages with not matching content-length, telnet is interpreting the rest of the message as if it was typed before/after the request. That is if the given length is longer than the message, the following carriage-returns are interpreted as part of the message. If the content-length is smaller, only the specified amount of characters are sent and the remaining part of the typed body are interpreted as a new HTTP request.

## 4 Basic authentication

### 4.1

A json object is returned with a user field which shows the user name and a boolean whether the authentication was successful. When reloading the page, the same object is shown without the need to login again.

### 4.3

It does not work, the response is again `401 UNAUTHORIZED`.

## 5 Survey TODO applications

## 5.2

Here are some web design princicles below:

1.do not make me think(minimising cognitive efforet)
2.minimize noise&clutter
3.if you cannot make it self-evident, make it selft-explanatory

1 https://www.toodledo.com/
There are a lot of texts on its first page to explain what is toodledo , nobody will read in fact….So I can not know well about the web at my first signt .Also I think it is a little noise especially the three pictures on first page .

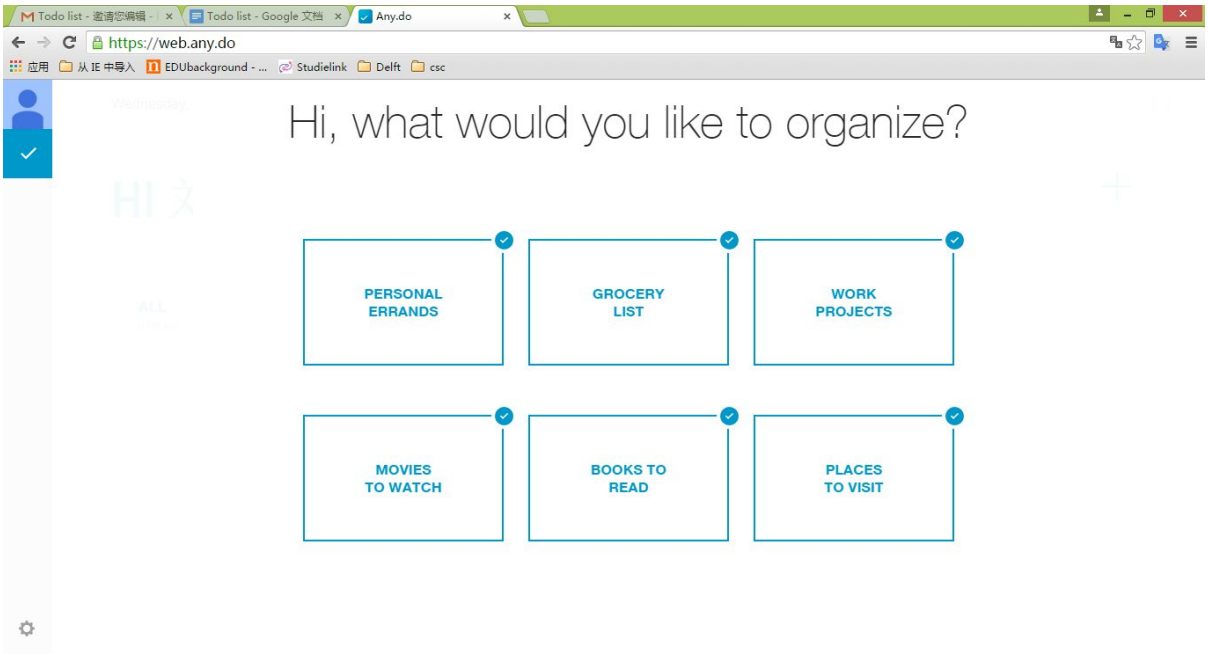
2.http://www.any.do/
I like this website most. It does very well in minimising cognitive effort and self-explanatory. On the entry page ,"Sign up for free" is so obvious that I do not need to look for it. And there are titles and subtitles to make me understand what it is aimed for. The text is short and very easy to understand.

## 6 Usability test

1. https://www.toodledo.com/
It takes me about two minutes to perform the tests. First I click register which is very quickly .Then I created a task through the "+" botton. However It confused me where to set the due time I tried several times and alse I can not find the reminder one day before deadline. It only has choices of 'daily ""weekly ".biweekly' 'monthly' and etc
'

2.http://www.any.do/

I faied to creat a task!!
After I click '"sign up for free" then "google" then "accept", I finished registering a new account . However when I log in, it stayed on the same page no matter which frame I clicked. It just did not work




## 7. Features

- Share todo list
- Set a reminder on a List or an item of a list
- Order todo lists according to data, topic
- Add todo list items
- Remove todo list items
- Mark a list item as done
- Undo changes
- Assign importance to list items
- Login with account to keep personal data
- Handle multiple lists

## 8.3 Design goals

Hi everyone,
We are Jan and Yixuan and we are the designers of the Todo Web App TODAy. Our target audience are students. We are students ourselves and therefore understand the problems that students have with organizing their days in an efficient way.
Students are extremely intelligent and capable people with a continuous flow of ideas and the will to make the world a better place. Unfortunately students also have a busy schedule which makes them lose the overview. Our goal is to support students and empower them to use their time as good without forgetting the small things. TODAy will help you get things done now instead of waiting till the last minute.Just do it from today not tomorrow!

We envision the app to make it very quick to add an idea or a thought as a note, but also to create large list in collaboration with other students to make task division and tracking a breeze when working in groups. Also, group assignment can be shared with others.
To avoid missing deadlines, there is a notice especially for alarming before the task date. (Students in tudelf can also export their timetable to our web app)

# Assignment 2

## 1 Step-by-step plan

**Adding an item with due date and importance rating**

1. Add an onclick event to the “Add todo-item”
2. Create a javascript function that creates a new item with the cursor ready to write the text, shows a field for adding a due date and buttons for the importance rating
3. Attach a javascript function to the due-date field to set the date when pressing enter
4. Attach a javascript function to the importance buttons which creates as attribute in the html

**Delete a todo item**

1. Write a javascript function which removes the todo-item
2. Attach the function to the onclick event of the delete button

**Set the status of a todo to “done”**

1. Each todo-item should have a checkbox
2. Write a javascript function that adds a “done” html attribute to the todo-item when the box is checked

**Change an existing todo item’s content (i.e. todo text, due date, importance rating and status)**

1. Write a javascript function that shows a cursor in the text of the todo-item
2. Attach the function to the onclick event of the text

**Sort the items according to due date / importance**

1. A button should exist for sorting the items by date or importance
2. Write a javascript function that sorts the entries by their date and importance attribute
3. Attach the javascript function to the buttons

**Move an item up or down manually**

1. Two buttons for the moving should exist
2. Write a javascript function that moves html elements up or down
3. Attach the javascript functions to the onclick events of the buttons

## 2.object-oriented programming

there are three  three design patterns:  1. basic constructor 2.prototype-based constructor 3.  module
We choose prototype-based constructor pattern;
Each possible item on our todo-list page will be an Item which is the base prototype for all types of items. For example a todo-item is an item, but a shared todo-list with many todo-item is also an item. This way we can reuse a lot of code.

## 4.Start my sql server

## 5.CONNECT to SQL SERVER via command line

```
mysql> HELP SHOW DATABASES
Name: 'SHOW DATABASES'
Description:
Syntax:
SHOW {DATABASES | SCHEMAS}
        [LIKE 'pattern' | WHERE expr]
```

SHOW DATABASES lists the databases on the MySQL server host. SHOW SCHEMAS is a synonym for SHOW DATABASES. The LIKE clause, if present, indicates which database names to match. The WHERE clause can be given to select rows using more general conditions, as discussed in http://dev.mysql.com/doc/refman/5.5/en/extended-show.html.

You see only those databases for which you have some kind of privilege, unless you have the global SHOW DATABASES privilege. You can also get this list using the mysqlshow command.

If the server was started with the --skip-show-database option, you cannot use this statement at all unless you have the SHOW DATABASES privilege.

URL: http://dev.mysql.com/doc/refman/5.5/en/show-databases.html

```
mysql> SHOW DATABASES
        -> ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| example            |
| mysql              |
| performance_schema |
| todo               |
+--------------------+
5 rows in set (0.00 sec)
```

## 6 EXPLORE THE STRUCTURE OF THE todolist database

```
mysql> USE todo
Database changed
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| todo       |
+------------+
1 row in set (0.00 sec)

mysql> SHOW TABLES;
+----------------+
| Tables_in_todo |
+----------------+
| ItemTag        |
| Tag            |
| ToDoAssignment |
| ToDoItem       |
| ToDoList       |
| User           |
+----------------+
6 rows in set (0.00 sec)

mysql> DESC ItemTag
    -> ;
+--------+---------+------+-----+---------+-------+
| Field  | Type    | Null | Key | Default | Extra |
+--------+---------+------+-----+---------+-------+
| ToDoId | int(11) | NO   | PRI | NULL    |       |
| TagId  | int(11) | NO   | PRI | NULL    |       |
+--------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> DESC Tag;
+-------+---------+------+-----+---------+----------------+
| Field | Type    | Null | Key | Default | Extra          |
+-------+---------+------+-----+---------+----------------+
| Id    | int(11) | NO   | PRI | NULL    | auto_increment |
| Text  | text    | YES  |     | NULL    |                |
+-------+---------+------+-----+---------+----------------+
2 rows in set (0.00 sec)

mysql> DESC ToDoAssignment;
+------------+-----------+------+-----+---------+----------------+
| Field  | Type         | Null | Key | Default | Extra          |
```

```
+------------+-----------+------+-----+---------+----------------+
| ToDoId     | int(11)   | NO   | PRI | NULL    | auto_increment |
| AssigneeId | int(11)   | NO   | PRI | NULL    |                |
| AssignDate | timestamp | YES  |     | NULL    |                |
+------------+-----------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> DESC ToDoList;
+--------------+------------+------+-----+---------+----------------+
| Field        | Type       | Null | Key | Default | Extra          |
+--------------+------------+------+-----+---------+----------------+
| Id           | int(11)    | NO   | PRI | NULL    | auto_increment |
| Name         | text       | YES  |     | NULL    |                |
| CreationDate | timestamp  | YES  |     | NULL    |                |
| Owner        | int(11)    | YES  | MUL | NULL    |                |
| IsPublic     | tinyint(1) | YES  |     | NULL    |                |
+--------------+------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> DESC User;
+----------+----------+------+-----+---------+----------------+
| Field    | Type     | Null | Key | Default | Extra          |
+----------+----------+------+-----+---------+----------------+
| Id       | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name     | text     | YES  |     | NULL    |                |
| Email    | text     | YES  |     | NULL    |                |
| Username | tinytext | YES  |     | NULL    |                |
| Password | text     | YES  |     | NULL    |                |
+----------+----------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

**RELATION Schema:**

ItemTag(ToDoId,TagId)

↗ ↖

**Attribute Name:** ToDoId **Domain:**int(11) (primary key)

**Attribute Name:**TagId **Domain:**int(11) (primary key)

Tag(Id,Text)

↗ ↖

**Attribute Name:**Id **Domain:**int(11) (primary key)

**Attribute Name**:Text **Domain:**text

ToDoAssignment(ToDoId,AssigneeId, AssignDate)

**Attribute: Name:**ToDoId **Domain:**int(11) (primary key)

**Attribute: Name:**AssigneeId **Domain:**int(11) (primary key)

**Attribute: Name:**AssignDate **Domain:**timestamp

ToDoList(Id,Name,CreationDate,Owner, IsPublic)

(primary key)

**Attribute: Name:**Id **Domain:**int(11)

**Attribute: Name:**Name text

**Attribute: Name:**CreationDate text

**Attribute: Name:**Owner tinytext

**Attribute: Name:**IsPublic text

**User(Id, Name, Email, Username, Password)**

(primary key)

**Attribute:**

| **Name:** | Id | Name | Email | Username | Password |
|---|---|---|---|---|---|
| **Domain:** | int(11) | text | text | tinytext | text |