Exercises - HTTP

You can do most of the exercises in this document by yourself, but they are meant as exercises with a supplementary discussion in the class, so you will gain a lot more from participating in the class.

Monitoring HTTP Headers 1

Create a new NetBeans project (type Web-project -> select all default values in the wizard)
For this exercise we will just use the default index.html generated by NetBeans).

Press the run button. When the file is shown in the browser (Chrome), open the developer window (Ctrl-shift-j) and press F5

Observe and explain each of the values monitored (use view source to see the plain messages).

Hints: In order to better observe the values related to Cashing you might need to:

Go back to NetBeans and rename your file to index1.html

Go back to your browser and (while the developer window is open) change the url to point to the new file.

Observe the values

Press F5 and observe the values again.

Explain what you see.

Monitoring HTTP Headers 2

Add an image to the page

Add an external style sheet to the page (<link rel="stylesheet" type="text/css" href="myStyle.css">)

Reload the page again, observe the request being made, and explain the purpose of the Connection: header.

Monitoring HTTP Headers 3

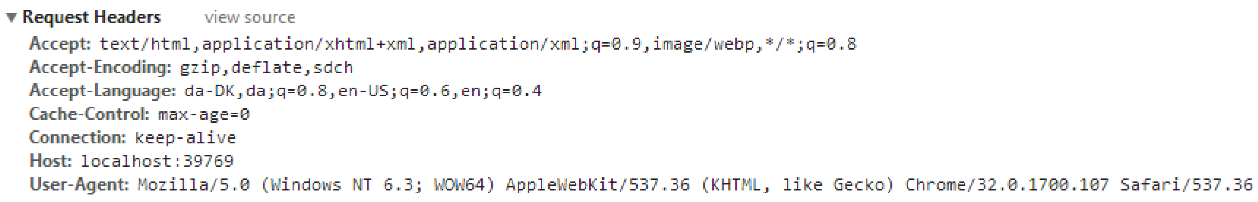
In Google Chrome enter this address (with the developer window open, and **exactly** as it is spelled):

http://www.fronter.com/cphbusiness

Explain, as well as you can (at a conceptual level), the first three request monitored.

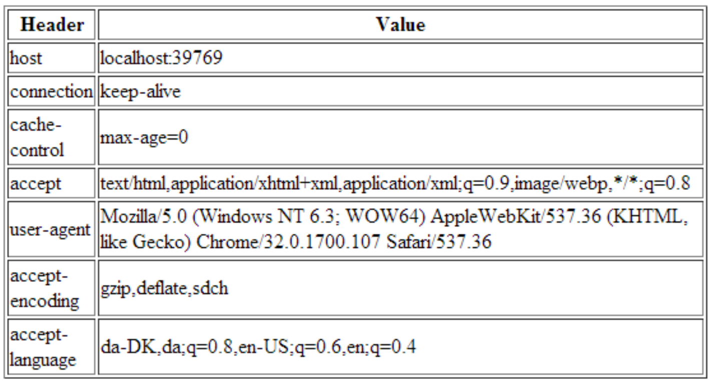
Get HTTP Request Headers on the Server

We have just seen that a HTTP request from a Browser typically includes a lot of headers with information related to the client.



```
▼ Request Headers      view source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
   Accept-Encoding: gzip,deflate,sdch
   Accept-Language: da-DK,da;q=0.8,en-US;q=0.6,en;q=0.4
   Cache-Control: max-age=0
   Connection: keep-alive
   Host: localhost:39769
   User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36
```

This information is available to the servlet via the request object. Create a Servlet, which should output this information in a table as sketched below (or in any way you like, don't think about presentation).

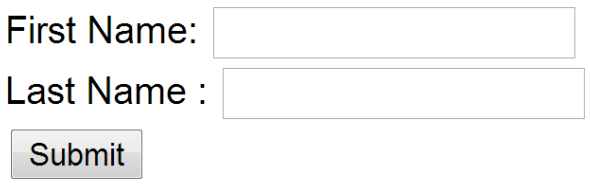


Header	Value
host	localhost:39769
connection	keep-alive
cache-control	max-age=0
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
user-agent	Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36
accept-encoding	gzip,deflate,sdch
accept-language	da-DK,da;q=0.8,en-US;q=0.6,en;q=0.4

Hints: Use the request objects getHeaderXXX methods.

Get/Post-parameters

Create a new html-file in the web-project made in exercise 1.
Add a form to the file, including two text input boxes and a submit button as sketched below:



Add an extra input field to the form with type="hidden", name="hidden" and value=12345678.

Add the value "#" for the forms action attribute.

Set the forms method-attribute to the value "GET" (actually the default value) and test the form. Observe what happens in your browsers address field.

Change the forms method-attribute to the value "POST" and test the form. Observe the change in your browsers address field. See whether you can find out, how the parameters are passed in, for a POST request.

<u>Make sure you understand the difference between GET and POST, also the non-idempotent part.</u>

Session and Cookies

For the next two exercises/demoes you should create a new web-project.

Both the demoes uses a Servlet and this will hopefully be the last time where you will use a Servlet for presentation ☺

Sessions (Session Cookies) Class exercise/demo

a) In your web project use the wizard to generate a new Servlet
b) Enter *SessionDemo* as the name of the Servlet and *servlets* as package name
c) Right click the file and select Run to see "what is does"
d) Change the generated processRequest(..) method as sketched below.

```
protected void processRequest(HttpServletRequest request,
                              HttpServletResponse response)
     throws ServletException, IOException {
  String name = request.getParameter("name");
  if (name != null) {
    request.getSession().setAttribute("name", name);
  } else {
    name = (String) request.getSession().getAttribute("name");
  }
  response.setContentType("text/html;charset=UTF-8");
  try (PrintWriter out = response.getWriter()) {
    out.println("<!DOCTYPE html>");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet SessionDemo</title>");
    out.println("</head>");
    out.println("<body>");
    if (name != null) {
      name = (String)request.getSession().getAttribute("name");
      out.println("<p> Welcome " + name  + " !</p>");
    } else {
      out.println("<h2>Please enter your name, and submit</h2>");
      out.println("<form action='SessionDemo'>");
      out.println("<input type='input' name='name'>");
      out.println("<input type='submit'></form>");
    }
    out.println("</body>");
    out.println("</html>");
  }
}
```

e) Enter your name and press submit, copy the URL in the browser into your clipboard, close the tab (but not the browser) and load the page again in a new tab using the URL in the clipboard.
f) While doing the things in step e, you should monitor the content of your local cookies and the HTTP requests being sent, using the development tools in Chrome.
g) Most import part of this exercise:
Explain (on paper) using both words and images how the Server can maintain state between subsequent calls even when using a stateless protocol

Persistent Cookies   Class exercise/demo

a) In your web project use the wizard to generate a new servlet
b) Enter *CookieDemo* as the name of the Servlet and *servlets* as package name
c) Change the generated processRequest(..) method as sketched below.

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
                              throws ServletException, IOException {
String name = request.getParameter("name");
if (name != null) {
        Cookie cookie = new Cookie("username", name);
        cookie.setMaxAge(60 * 60 * 24 * 365);
        response.addCookie(cookie);
}
Cookie[] cookies = request.getCookies();
if (cookies != null) {
        for (Cookie cookie : request.getCookies()) {
        if (cookie.getName().equals("username")) {
                  name = cookie.getValue();
        }
        }
}
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet CookieDemo</title>");
        out.println("</head>");
        out.println("<body>");
        if (name != null) {
        out.println("<p> Welcome " + name + " !</p>");
        } else {
        out.println("<h2>Please enter your name, and submit</h2>");
        out.println("<form action='CookieDemo'>");
        out.println("<input type='input' name='name'>");
        out.println("<input type='submit'></form>");
        }
        out.println("</body>");
        out.println("</html>");
}
}
```

d) Enter your name and press submit, copy the URL in the browser into your clipboard, close the tab (but not the browser) and load the page again in a new tab using the URL in the clipboard.
e) Now close your browser (you could even close your laptop, but don't ;-) , open it again and load the page again using the URL in the clipboard
f) While doing the things in step e, you should monitor the content of your local cookies and the HTTP requests being sent, using the development tools in Chrome.
g) Most import part of this exercise:
h) Explain (on paper) how Cookies can be used to maintain "state" on the client between subsequent calls to a server, even when a browser has been closed down.