The HTTP Protocol

*Before you start, you should create a document (or a just a piece of paper) where you should write down the Status Code generated by each of the following exercises (You need this for exercise 4-c)*

## 1) Monitoring HTTP Headers 1

Create a new NetBeans Maven Web-project.

For this exercise, we will just use the default index.html generated by NetBeans.

Press the run button. When you see the file in the browser (Chrome), open the network tab in the developer window (Windows: Ctrl-shift-j / Mac: Option-Command-i) and press F5

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

**I see the ms. It takes for the page to load the files.**

Hints: In order to better observe the values related to Caching 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.

It renames the file? The index is seen as the file in the top, which is now renamed. WTF am I doing??

### 2) 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(s) being made, and explain the purpose of the connection header.

The requests being made are now 2 GET´s. Whereas the CSS now gives me a 404.

### 3) Monitoring HTTP Headers 3 (Response-codes 3xx)

In the Web-project, created for 1+2, add a new HTML-page called **r.html** and add this text in an h1-tag “**You got redirected to me**”.

Use the Wizard to create a servlet called redirect

Remove the processRequest and the doPost method completely from the generated servlet-code.

In the doGet(..) method replace the call to processRequest with this line: response.sendRedirect("r.html");

While your server is running, open a (Chrome) browser, and Developer Tools and the network tab.

Enter the address for the servlet (**http:localhost:8080/redirect)** into the browser and explain:

* The two HTTP-request you see

You get a 302 which means redirect, and a 304 which means that the first redirect was successful.

This is like a 200 when encountering a 302.

* How the browser knew where to go in the second request

Because of the first request?

### 3a) Redirecting to HTTPs instead of HTTP

In Chrome enter this address (with the developer window + the network-tab open), and exactly as it is spelt: [**http://studypoints.info**](http://studypoints.info)

Explain the first two requests monitored (notice where you requested to go, and where you actually ended).

It says 301, moved permanently. It is moved permanently from HTTP, to HTTPS.

Important: Later this week, you will learn how to set up your own server to use https, **and ONLY https**.

### 4a) Status Codes (5xx)

Use the Wizard to create a servlet called **Ups**

In the processRequest(..) method, just before the try-statement add this code:

int result = 100/0;

While your server is running, open Chrome developer tools and the network tab and then call the servlet.

Write down the response code generated by the server as for the previous exercises

It gives us a 500 arithmeticexception.

### 4b) Status Codes (4xx)

While your server is running, open Chrome developer tools and the network tab, and call this address: **http://localhost:8080/i\_dont\_exist**

Write down the response code generated by the server as for the previous exercises

404 The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

### 4c) Status Codes - Ranges

Your document, containing the Status Codes for all the exercises done so far, should now contain codes like 2xx, 3xx, 4xx and 5xx.

Explain (write down your answer so you won’t forget) the meaning of the first digit in the 3-digit Status Codes you have seen so far.

2 is success.

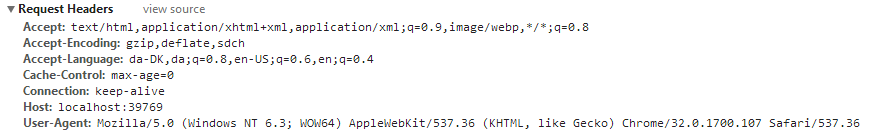
3 is redirect

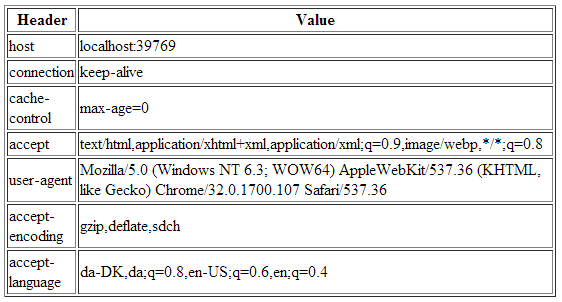
4 is error

5 is error

### 5) Get HTTP Request Headers on the Server

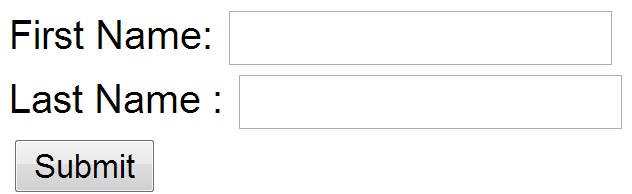
We have seen that an HTTP request from a Browser typically includes a lot of headers with information related to the client.



This information is available to a servlet (actually to any web-server technology) via the request object. Create a Servlet, which should output this information in a table as sketched in this figure (or in any way you like, but don’t focus on presentation).

*Hints: Use the request objects getHeaderXXX methods.*

### 6) 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 browser's address field.

Now it requests something.

Hidden means that it doesn’t show in the url what you request.

Change the forms method-attribute to the value “POST” and test the form. Observe the change in your browsers address field. Figure out (using Chrome Developer Tools), how parameters are passed in, for a POST request.

Now it posts something.

Write down your observations

## Session and Cookies

*For the next two exercises/demos you should create a new Maven web-project. Both the demos use a Servlet.*

### 7) Sessions (Session Cookies)

In your web project use the wizard to generate a new Servlet

1. Enter ***SessionDemo*** as the name of the Servlet and *servlets* as package name
2. Right-click the file and select Run to see “what is does”

You run the servlet and remembers it in the sessionscope.

1. 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>"**);**

**}**

**}**

1. 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.

**The cookie remembers me because the sessionscope is still alive.**

1. While doing the things in step d, you should monitor the content of your local cookies and the HTTP requests being sent, using the development tools in Chrome.

It logs the cookies in the cookies tab. Requests are GET, but I cant find the POST for when I submit?

1. **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 the state is not transmitted from the client to server.

**It is saved in cookies, because HTTP is stateless. This is why cache and cookies are important to make it dynamic? Im not sure if that’s the answer youre looking for.**

Write down your observations

### 8) Persistent Cookies

1. In your web project, use the wizard to generate a new servlet
2. Enter *CookieDemo* as the name of the Servlet and *servlets* as package name
3. 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>"**);**

**}**

**}**

1. 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.
2. 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
3. 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.

It sets the cookie, and then reloads the page?

1. **The most import part of this exercise:**

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.

Well it can save the cookies in the url and “post” it the next time someone uses the same URL.

So lets say someone has been shopping in a webshopp, and because of the https is stateless, the website has no idea who is using it. But it is saved during the first visit as a cookie, which means that if they use the same URL again, it will know because of the header. It will then be posted as a cookie, and then the response from the website will be in a new state??

Write down your observations