

# 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

**Observe and explain each of the values monitored:**

The HTTP response status code tells us whether a specific HTTP request has been successfully completed. Here are the observations from the developer window:

Status code	Observation	Explanation
200 OK	The first time the project runs	The request has succeeded.
304 Not Modified	When refreshing (pressing F5) on the page after the project has run successfully	The request has not been modified. It is used for caching. Caching is when the webpage saves information in order to make the webpage run faster.
404 Not Found	When renaming index.html-file before changing the url in the browser to the new name	The server cannot find the requested resource, because we have changed the name. Index.html does not exist.
200 OK	The first time the project runs (after renaming index.html)	The request has succeeded.

## 2) Monitoring HTTP Headers 2

**Observe and explain:**

Here are the observations from the developer window:

Status code	Observation	Explanation
200 OK	The first time the project runs for index, picture and stylesheet	The request has succeeded for index.html, picture and stylesheet.
304 Not Modified	When refreshing project 304 is status for index and stylesheet	The request for index.html and stylesheet has not been modified.
200 OK	When refreshing project 200 is status for picture from internet	The request has succeeded, the project must request the picture each time.
304 Not Modified	When refreshing project 304 is status for picture from project folder	The request for a picture can be saved in caching when the picture is in the project folder.

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

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

Here are the observations from the developer window:

Status code	Observation	Explanation
302 Found	When running the project and calling the server redirect	The requested resource has been changes temporarily. It means that the client should use the same url because new changes might be made later.

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

Explain the first two request monitored entering this address: <http://studypoints.info>

Here are the observations from the developer window:

Status code	Observation	Explanation
301 Moved Permanently	Status for <a href="http://studypoints.info">http://studypoints.info</a>	The requested resource has been changes. I requested to go to this page, but I was given the new url as response.
200 OK	Status for <a href="https://studypoints.info">https://studypoints.info</a>	The request has succeeded, and I ended up on this page instead.

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

Write down the response code generated by the server.

Here are the observations from the developer window:

Status code	Observation	Explanation
500 Internal Server Error	When running the project and calling the server Ups	The server has encountered a situation it does not know how to handle. The project gets an Arithmetic Exception because we try to divide by 0 in the servlet. This unexpected condition is preventing the server from fulfilling the request.

## 4b) Status Codes (4xx)

**Write down the response code generated by the server.**

Here are the observations from the developer window:

Status code	Observation	Explanation
404 Not Found	When trying to call this address <a href="http://localhost:8080/i_dont_exist">http://localhost:8080/i_dont_exist</a>	The server cannot find the requested resource, because we have no server called i_dont_exist.

## 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 the meaning of the first digit in the 3-digit Status Codes.**

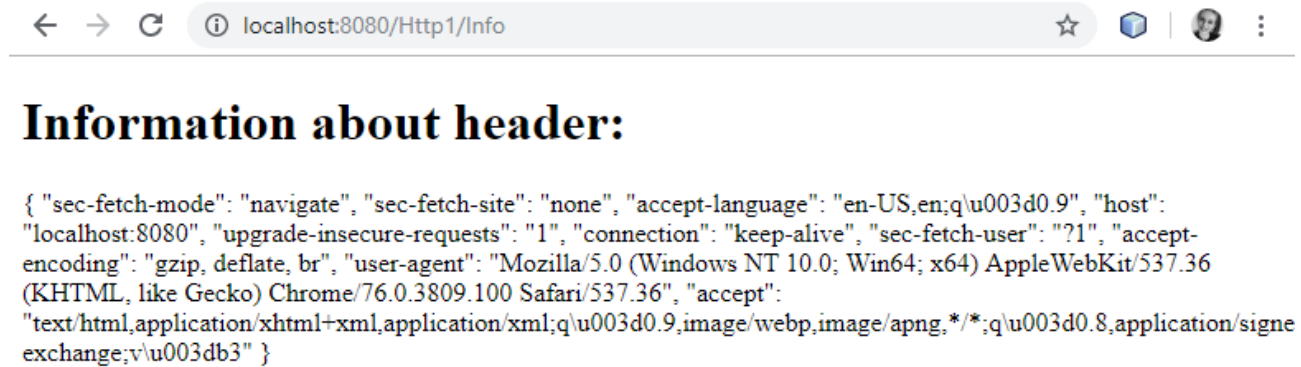
The HTTP response status code tells us whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

Status code	Explanation
100-199	Informational responses Status codes in the range 100-199 is informative to the client. It explains what is happening while the program is running.
200-299	Successful responses Status codes in the range 200-299 indicates that the request has succeeded.
300-399	Redirects Status codes in the range 300-399 indicates that the request has been modified in order to make it run successfully. It describes the changes made and shows how the program ended up running instead in order to run successfully.
400-499	Client errors Status codes in the range 400-499 indicates that the client has made an error in the request that should be fixed before the program is able to run successfully.
500-599	Server errors Status codes in the range 500-599 indicates that there is an error within the server which makes it unable to run the request successfully.

## 5) Get HTTP Request Headers on the Server

Create a Servlet, which should output information related to the client.

I have created a servlet called Info that outputs information about the client. Here is a screenshot of the webpage, when I run the servlet:



Here is my code from my NetBeans project Http1, which is uploaded on GitHub as well.

```
@WebServlet(name = "Info", urlPatterns = {"/Info"})
public class Info extends HttpServlet {

    private String getHeadersInfo(HttpServletRequest request) {

        Map<String, String> map = new HashMap<String, String>();
        Enumeration headerNames = request.getHeaderNames();
        while (headerNames.hasMoreElements()) {
            String key = (String) headerNames.nextElement();
            String value = request.getHeader(key);
            map.put(key, value);
        }
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        String json = gson.toJson(map);
        return json;
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        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 Info</title>");
            out.println("</head>");
            out.println("<body>");

            out.println("<h1>Information about header:</h1>");
            out.println(getHeadersInfo(request));

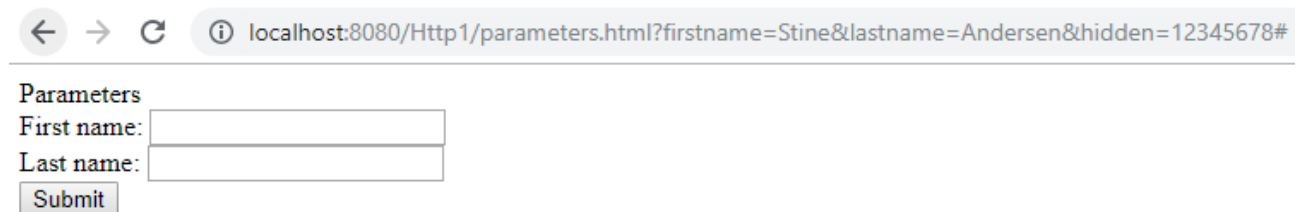
            out.println("</body>");
            out.println("</html>");

        }
    }
}
```

## 6) Get/Post-parameters

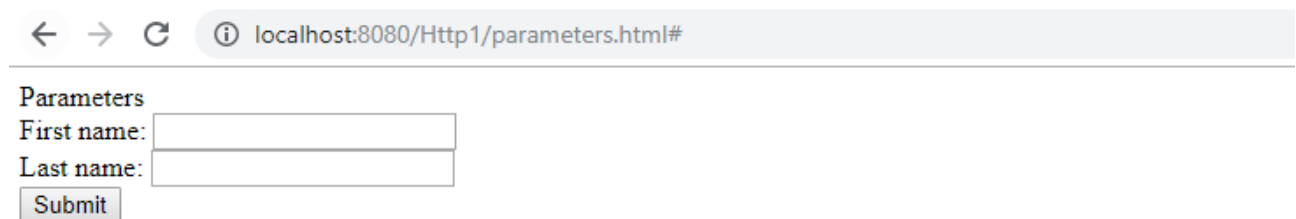
**Set the forms method-attribute to the value “GET” and test the form.  
Observe what happens in your browser's address field.**

I have created a html-file called parameters.html with a form with text inputs, a hidden input and a submit box. The form's action attribute is # and the method attribute type is GET. Here is a screenshot of the webpage, when I run the file:

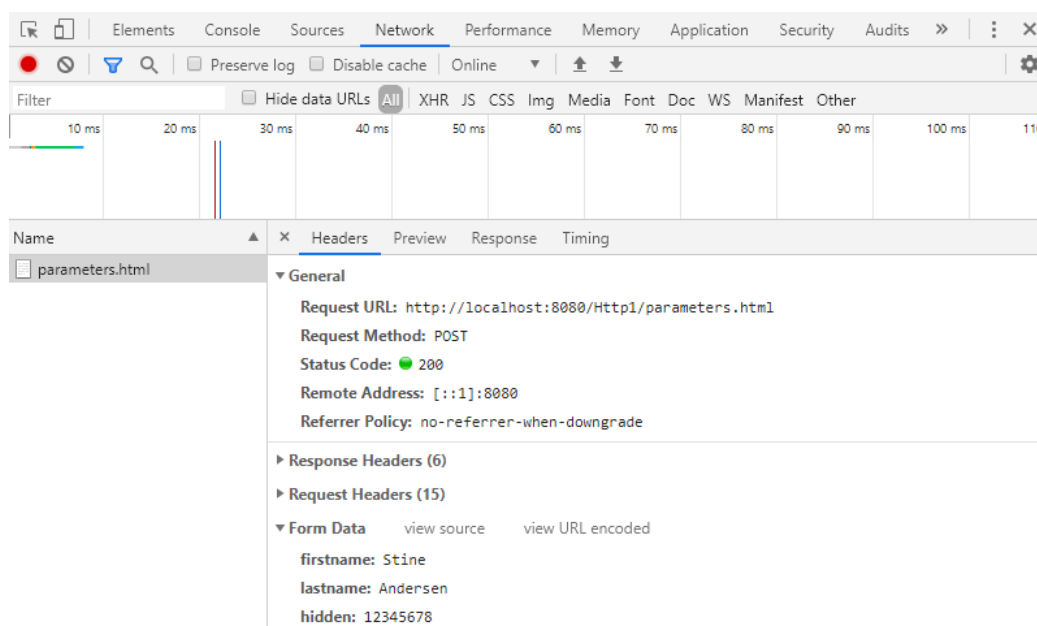


The browser's address field is showing the input parameters in the address field. You are able to see the input parameters for first name and last name and the hidden input.

**Change the forms method-attribute to the value “POST” and test the form. Observe the change in your browsers address field.**



The browser's address field is no longer showing the input parameters. The parameters are instead sent to the server using the request body. This is a more secure way to send data, because it hides your data and adds them to your header, without the data being shown. The data is still saved in the form, if you check the headers of the file parameters.html:



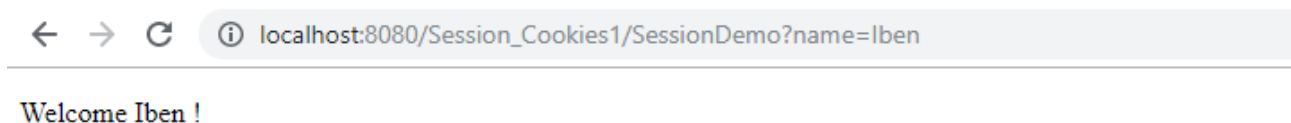
## Session and Cookies

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

### 7) Sessions (Session Cookies)

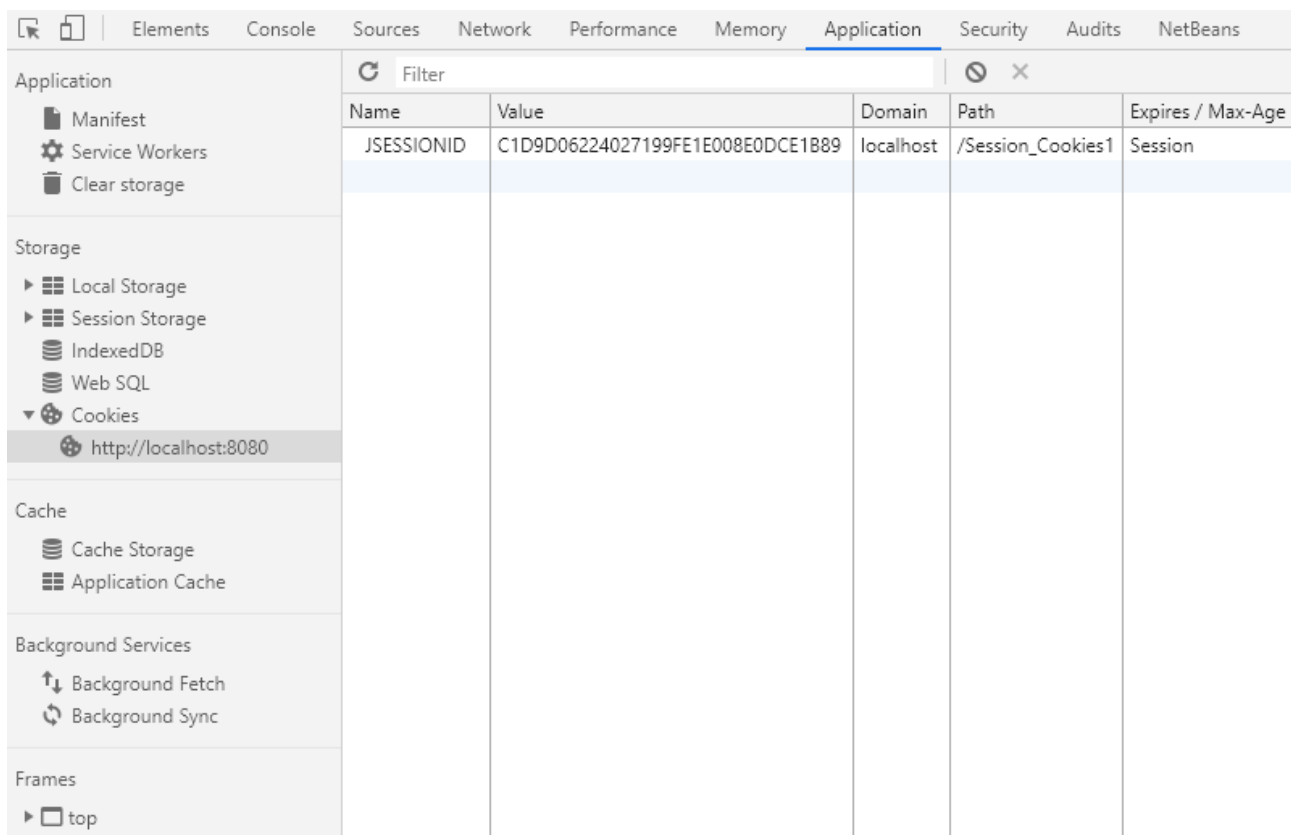
**Explain (on paper) using both words and images how the Server can maintain state between subsequent calls even when the state is not transmitted from the client to server.**

I enter my name and press submit, and my input parameter is saved in the browser's address field.



Even when I close the tab and load the page again in a new tab using the url I copied from the browser, the server has maintained the state.

If you look in the development tools in Chrome, you can see that input is saved in a session cookie. The server is able to maintain my input because it has been saved in a session cookie. Session cookies are temporary cookie files , which are erased when you close your browser.



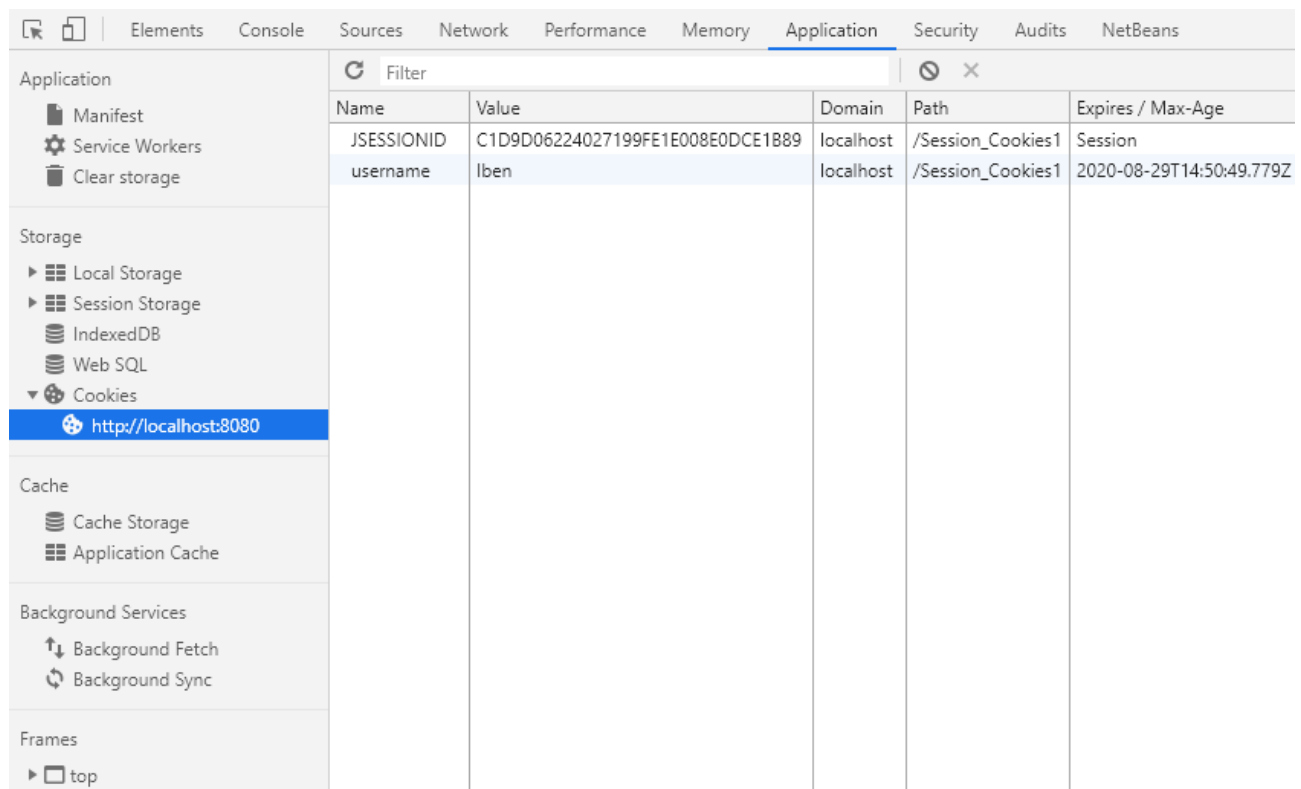
## 8) Persistent Cookies

**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.**

Now even when I close the browser and revisit the page the server has maintained the state.

If you look in the development tools in Chrome, you can see that input is saved as in exercise 7. The difference is, that in exercise 7 the input was saved in a temporary session cookie file, and now the input is saved in a persistent cookie file.

A persistent cookie file is saved in a subfolder in your browser. The file stays there even when your browser has been closed down. You can either delete it manually or you can let your browser delete the file, when it reaches its expiration date. You can even see the expiration date on the list below.



The screenshot shows the Chrome DevTools Application tab. The left sidebar is expanded to 'Cookies' for the URL 'http://localhost:8080'. The main panel displays a table of cookies.

Name	Value	Domain	Path	Expires / Max-Age
JSESSIONID	C1D9D06224027199FE1E008E0DCE1B89	localhost	/Session_Cookies1	Session
username	Iben	localhost	/Session_Cookies1	2020-08-29T14:50:49.779Z