# The HTTP Protocol

## Monitoring HTTP Headers 1

What is observable is the different requests, and responses from communicating with the webapplication. The different information which can be seen are some of the following:

- Status code
    - This explains what was the response from the given request to the server, in my case it is a 200, meaning OK (everything is good)
- Method
    - This defines the type of request the browser send to the webapplication, in my case it is a GET request. In this case it I was only interested in getting what ever was on Index.html.
- Domain
    - This is simply the ‘target’ of the request that was sent. In my case it was simply ‘localhost:8080’ because I launched the webapplication from the IDE.
- Type
    - This specifies what type of content was involved, due to my request for the index.html – the type was html.

## Monitoring HTTP Headers 2

When adding the image and the reference to the stylesheet, these changes are also reflected in the requests going to the webapplication. Now we are not only request the index.html, we are also requesting the  stylesheet and the image on index.html. We can also see that the types in the request are different, for the image it is **jpeg** and for the stylesheet it is **css**.

**The connection header**,  controls whether or not the connection should stay open after the current transaction, fx with the value **keep-alive**, the server is asked to keep the connection alive after the transactions has been completed.

## Monitoring HTTP Headers 3

What is happening is that the Servlet (RedirectServlet.java) is registered to the URL-pattern /Redirect, this means that when typing in http://localhost:8080/Web_Project/pages/Redirect the webapplication, finds the corresponding servlet (matching *Redirect) and executes the code specified for the GET-*

*request. And since we specified in the servlet, that we want to redirect to* **pages/redirect.html** it simply redirects to this HTML page.

## Redirecting to HTTPs instead of HTTP

We can see the first request is made for http://studypoints.info, but then we are redirected to https://studypoints.info.

So we actually asked for the insecure connection to studypoints.info, but the webapplication chose to redirect to the secure connection, by simply forwarding us to ‘https

## Status Codes (5xx)

When trying to access this new Servlet, it throws a status code 500 – Internal Server Error

## Status Codes (4xx)

When trying to acess **i_dont_exist**, the webapplication respond with a status code 404 – Not Found

## Status Codes – Ranges

- Status codes 2xx
  - These codes tells us that nothing went wrong and everything is ok.
- Status codes 3xx
  - These codes tells us somthing about that we were redirected.
- Status codes 4xx
  - These is about the user, fx the user is trying to get something which can’t be found, or the user doesn’t have permission to do a given action.
- Status codes 5xx
  - These tells us that something went wrong on the server.

# Get HTTP Request Headers on the Server

The request header details page can be found on /Web_Project/Request

# GET/POST-Parameters

When the method of the form is set to GET, the hidden input value is visible in the URL of the page as well as the given input for both the first and last name.

If the method is set to POST, the differect input is not visible in the URL, but by inspecting the POST request in the developer tool, it is visble in the tab ‘Params’

# Sessions (Session Cookies)

```
13  @     protected void processRequest(HttpServletRequest request,
14                                      HttpServletResponse response)
15              throws ServletException, IOException {
16          String name = request.getParameter( s: "name");
17          if (name != null) {
18              request.getSession().setAttribute( s: "name", name);
19          } else {
20              name = (String) request.getSession().getAttribute( s: "name");
21          }
22          response.setContentType("text/html;charset=UTF-8");
23          try (PrintWriter out = response.getWriter()) {
24              out.println("<!DOCTYPE html>");
25              out.println("<html>");
26              out.println("<head>");
27              out.println("<title>Servlet SessionDemo</title>");
28              out.println("</head>");
29              out.println("<body>");
30              if (name != null) {
31                  name = (String)request.getSession().getAttribute( s: "name");
32                  out.println("<p> Welcome " + name  + " !</p>");
33              } else {
```

On lines 17 through 21, we can see how we save the state of the session. If we have an input from the page we store it in session scope and if we do not, we fetch a name from the session scope. This has the effect that as long as the user is connected to the webapplication, the state will remain, but as soon as the connection is broken nothing is stored.

# Persistent Cookies

```
11
12  @WebServlet(name = "CookieDemo", urlPatterns = "/CookieDemo")
13  public class CookieDemo extends HttpServlet {
14      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
15              throws ServletException, IOException {
16          String name = request.getParameter( s: "name");
17          if (name != null) {
18              Cookie cookie = new Cookie( name: "username", name);
19              cookie.setMaxAge(60 * 60 * 24 * 365);
20              response.addCookie(cookie);
21          }
22          Cookie[] cookies = request.getCookies();
23          if (cookies != null) {
24              for (Cookie cookie : request.getCookies()) {
25                  if (cookie.getName().equals("username")) {
26                      name = cookie.getValue();
27                  }
28              }
```

On lines 17 through 28, we can see how we handle saving cookies. When we have a new name, we create a new cookie for holding this information. Thereafter we check if we have any cookies already stored. If we stored cookies, we check them for a username and if we find one, we store the value in name.