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**](http://studypoints.info)

Here are the observations from the developer window:

|  |  |  |
| --- | --- | --- |
| **Status code** | **Observation** | **Explanation** |
| 301 Moved Permanently | Status for <http://studypoints.info> | 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 <https://studypoints.info> | 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 <http://localhost:8080/i_dont_exist> | 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.**

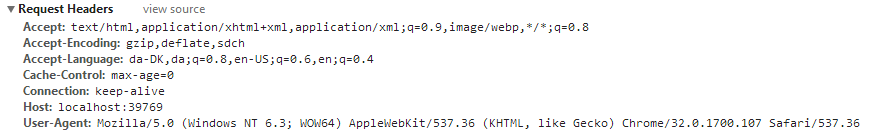
The HTTP response status code tells us whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

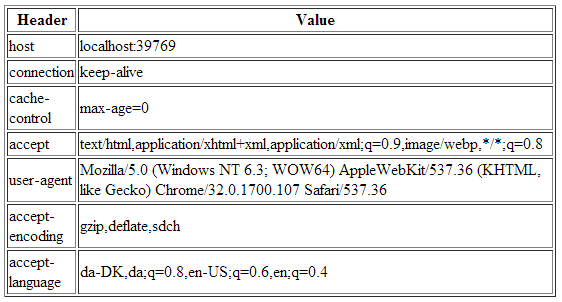
|  |  |
| --- | --- |
| **Status code** | **Explanation** |
| 100-199 | Informational responses  Status codes in the range 100-199 is informative to the client. It explains what is happening while the program is running. |
| 200-299 | Successful responses  Status codes in the range 200-299 indicates 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

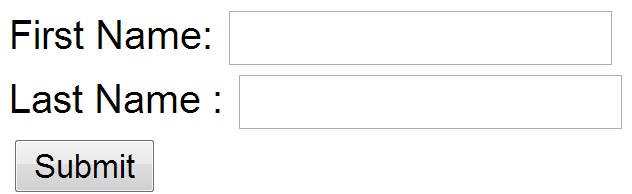
We have seen that an HTTP request from a Browser typically includes a lot of headers with information related to the client.



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.

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.

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”
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)** **{**

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.
2. 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.
3. **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.

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.
4. **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.

Write down your observations