Frank Grevelink & Stijn Klarenbeek

Internettechnologie – Opdracht 3

Documentatie

Opdracht 3

Documentatie HTTP-server

Table of Contents

[1. Voorwoord 3](#_Toc472529580)

[2. Functionaliteit 4](#_Toc472529581)

[3. Systeembeschrijving 5](#_Toc472529582)

[Responsehandler 5](#_Toc472529583)

[Klassendiagram 6](#_Toc472529584)

[4. Tests 7](#_Toc472529585)

# Voorwoord

Bij deze opdracht kon er gekozen worden uit een aantal verschillende protocollen waarmee data verstuurd/ontvangen kan worden. Wij hebben gekozen voor het http-protocol, aangezien we beide graag wilden weten hoe een webserver nou eigenlijk werkt. Ook leek het ons het protocol waar het meest over te vinden is, aangezien er erg veel mee gewerkt wordt.

In dit verslag beschrijven wij welke onderdelen van het http-protocol we hebben geïmplementeerd, op welke manier we dit hebben gedaan en hoe we hebben getest.

# Functionaliteit

In de opdrachtomschrijving staat de volgende functionaliteit:

* .html-files en .jpg files versturen
* *root* van een directory moet beantwoord worden met een index.html
* .*htaccess*-file ondersteuning met gebruikersnaam/wachtwoord (http basic)
* Redirecten van een gebruiker doormiddel van statuscode *301 Moved Permanently*

Wij hebben alle bovenstaande functionaliteit geïmplementeerd en zelf hebben wij de volgende functionaliteit toegevoegd:

* Ondersteuning voor meer bestanden, zoals fonts (*glyphicon* in bootstrap)
* Een *internal server error* met *http-status 500* wanneer er geen response gegenereerd kan worden door een fout.
* Wanneer er een fout ontstaat blijft de server werken en crasht alleen de foute thread.
* De poort waarop de server moet communiceren is mee te geven als argument.
* De juiste slash voor het opzoeken van *files* en *folders* wordt gekozen aan de hand van het OS (Elk OS behalve Windows gebruikt de forward slash).

Wanneer er een request wordt gedaan naar de server krijgt de client altijd een response terug (of er moet echt iets flink fout gaan). Dit request kan een GET, POST, PUT of DELETE zijn. Er wordt echter niets met meegestuurde data gedaan van bijvoorbeeld een POST, PUT of DELETE, deze krijgen geen andere behandeling dan een GET-request. De andere soorten requests hebben we niet ingebouwd wegens tijdgebrek.

# Systeembeschrijving

## 3.1 Start

Het programma wordt gestart vanuit de class *WebServer.java*. Bij het starten van het programma kan als argument een nummer opgegeven worden dat als poort wordt gebruikt. Als er geen nummer op wordt gegeven wordt er poort *8080* gebruikt.

Vervolgens wordt de *run()*-methode aangeroepen. Hierin staat een infinite loop die een nieuwe *ServerSocket* aanmaakt. Deze serversocket wacht op een verbinding van buitenaf op de ingestelde poort. Als deze poort al in gebruik is wordt het programma afgesloten met de melding dat er poort bezet is. Als de poort niet bezet is wordt de class *ResponseHandler.java* geïnstantieerd. Hier wordt de serversocket aan meegegeven, zodat de *ResponseHandler* zelf een response kan versturen via deze serversocket.

## *3.2 ResponseHandler* en *Headers*

In de *ResponseHandler* wordt eerst het *request* dat wordt gedaan door de client uitgelezen om te kijken wat de client precies wil. Vervolgens wordt er gecheckt of er een *basic authentication* mee is gestuurd als *header*. Als dit het geval is wordt het meegestuurde wachtwoord en gebruikersnaam in het geheugen gezet om later te controleren. Ook wordt de class *SystemInformation* geïnstantieerd. Hieruit kan worden opgevraagd welke slash er moet worden gebruikt en welke bestanden er geladen kunnen worden.

Vervolgens wordt er gekeken welke *URL* de client opvraagt. Als deze overeenkomt met */moved* wordt er meteen een response met http-statuscode *301* en de website [*http://www.google.com/*](http://www.google.com/) teruggestuurd als response. Als er geen bestand of map is gedefinieerd in de *URL* wordt er automatisch */index.html* achter geplakt om in dat geval altijd te proberen de index-pagina in de map te openen. Wanneer er wel een bestand is gedefinieerd wordt er gekeken of dit bestand bestaat.

Als dit bestand niet bestaat wordt er een *404* pagina geladen en getoond aan de gebruiker. Wanneer het bestand wel bestaat (Er wordt gekeken in de lijst met bestanden die gevonden zijn tijdens het starten van de *ResponseHandler*) wordt het pad gevormd naar het bestand (*htdocs/bestand.html*). Als er in de map waar het bestand in staat een .htaccess bestand voorkomt wordt er gekeken of de client een gebruikersnaam en wachtwoord heeft meegestuurd. Als dit het geval is worden deze geverifieerd. Als de gegevens kloppen wordt bestand ingeladen.

Op het moment dat deze gegevens niet kloppen wordt er weer opnieuw gevraagd om een gebruikersnaam en wachtwoord (door http-statuscode *401* mee te sturen samen met *WWW-Authenticate: Basic* als extra header).

Na de controle wordt de laatste stap doorlopen, het schrijven van de uiteindelijke response. Er wordt gekeken naar de extensie van het bestand. Wanneer de extensie overeenkomt met een van de ingebouwde extensies wordt het *MIME-Type/Content-Type* in de header goed gezet voor dat type bestand. Vervolgens wordt het bestand ingeladen in het geheugen als *String*. De header wordt nu opgehaald via de *Headers* class, en weggeschreven naar de output. Het bestand dat de client opvraagt wordt vervolgens omgezet in bytes en weggestuurd via de outputStream.

## 3.3 Klassendiagram



# Tests

We hebben alle functionaliteit getest via een *REST-client* (*Boomerang*). Dit hebben we gedaan omdat we op deze manier een goed overzicht krijgen van alles dat de server terugstuurt als response. Het is makkelijk om op deze manier de headers en body van de response te zien, alles is namelijk gewoon *plain-text* zonder opmaak o.i.d.

Het weergeven van de pagina’s en de authenticatie hebben we getest via Firefox, Safari en Chrome. Bepaalde functies werken in sommige browsers niet altijd helemaal lekker. Hoe dit precies komt weten we niet. Zo werkt de *301 Moved Permanently*-header alleen goed in Chrome, maar werkt het custom font juist weer goed in Firefox. Ook worden sommige bestanden soms niet geladen in een browser, maar werken dezelfde bestanden prima met een *REST-client*.