CS378 (Prof. Varsha Apte) Mini Project: HTTP Server

Devdeep Ray, 120050007

November 6, 2014

Abstract

This is a documentation of various features implemented in the HTTP server created as the mini project for CS 378. The server was implemented in Java and implements a some features of a HTTP/1.1 server. Various features and implementation details are given below.

1 Overall list of features

The following server related features were partially/fully implemented

1. GET
2. POST
3. HEAD
4. User html folders
5. CGI script execution in GET and POST
6. CGI enabler config file
7. HTTP keep-alive
8. Multi threading
9. Pipelining
10. Queueing
11. File caching
12. Statistics collection

The code has the following features

1. Soft coded strings
2. Configuration file for various settings
3. Debugging tools

2 Server features

2.1 GET

The GET method is primarily used to download a page from the server. CGI scripts can be executed using GET by passing arguments in the URL in the *?name=value* format, if the URL points to a valid CGI script. The body is ignored in the GET request

2.2 POST

The POST method is used to send data to the server. The destination file should be a valid CGI script. The arguments to the CGI script should be in the body and are given to the script via *stdin.*

2.3 HEAD

The HEAD method just returns the header of a GET request.

2.4 User html folders

Users can host public_html in their user folders. The user root directory must be set using the config file. public_html can also be changed to something else using the config file. The URL format is *http://serveraddress/ username/pathtouserfile* .

2.5 CGI script execution in GET and POST

GET and POST that point to valid CGI files will get executed correctly according to the different variable passing conventions of GET and POST.

2.6 CGI enabler config file

To enable CGI files in a particular folder to be executable, a file called *cgienab* must be present. The name of this file can be changed to something else through the config file. Also, the kinds of CGI scripts supported can be configured through the config file.

2.7 HTTP keep-alive

HTTP keep-alive has been implemented. A browser can use the same TCP connection for downloading multiple things. The number of transactions that happen on one connection can be limited by a setting in the config file.

2.8 Multi threading

The server supports multi threading. This can be turned on or off using the config file. As soon as a connection is accepted, all the processing is passed on to a thread in a thread pool, and the server continues to listen to new connections.

2.9 Pipelining

When keepalive is on, requests to the server and the responses are pipelined. When requests come from the client, they are put into a queue, and a processor thread drains this queue, generates responses and puts then in another queue to be sent. A sender thread picks out responses from this queue and sends them to the socket.

2.10 Queueing

The connections are queued into the thread pool. Whenever a thread gets free, it picks up one connection from this queue and processes it.

2.11 File Caching

A file cache is implemented. The cache has multiple sections. The file data is cached, the path status (file or directory) is cached, whether a file can be executed or not is cached, etc.

2.12 Statistics collections

A statistics collection system is implemented. With this, we can get connection statistics for each file. Stats logging can be enabled or disabled in server settings. We collect connection initiation time, closing time, data trasferred(sent and received), number of transactions on one tcp connection, etc.

3 Code features

3.1 Soft coded strings

Most string constants used in the program are all in a separate file, so that changing them will be easier if we want to change some things related to the behaviour of the HTTP protocol. For example, the names of the common attributes are stored in this single file, and whenever we need it elsewhere, for example, while generating a response, we use the string stored in the corresponding variable instead of a string constant.

3.2 Configuration file

The configuration file can be used to change a lot of settings. There is a brief description of how to use this later.

3.3 Debugging tools

Debug class is used to print debugging messages. Each class can be represented by a single bit position in an integer, and a mask is definied in the server settings. If the mask bit corresponding to the class bit is one, then we print the debug messages. This allows easy swithing on and off of debugging messages.

4 Using the config file

The config file must be in res/serverSettings.conf of the project root while running it in eclipse. The parameters are represented as *name : value* pairs. Here is an example config file:

```
socket-timeout : 10000
keep-alive : true
port-num : 8080
file-cache-timeout : 60000
file-cache-enabled : true
file-cache-size : 30
multi-threaded : true
pipe-enabled : true
max-session-queue : 10
min-session-threads :10
max-session-threads : 20
thread-keep-alive-time : 120000
keep-alive-max-requests : 10
debug-level : 0
cgi-conf-name : cgienab
not-found-file-path : res/notfound.html
redir-file-path : res/redir.html
internal-error-file-path : res/internalerror.html
bad-request-file-path : res/badrequest.html
cgi-extensions : py sh
text-types : htm html css js
web-root-path : /users/ug12
user-root-path : /users/ug12
user-html-folder : public_html
```