

How to create tiny HTTP server in Java

□ Posted on [May 8, 2017](#) by [adm](#) Posted in [JavaIO](#) — [0 Comments](#)

For really small applications and if you don't want to include a server like Tomcat is useful to implement a lightweight socket HTTP Server.

The following tiny HTTP Server is designed to serve two paths but will be easy to extend:

- home page: <http://localhost:8000/>
- hello page: <http://localhost:8000/hello/{name}>

Additional, a page not found is returned to any other pages.

Server

The code is fairly simple, although there are a few important lines. In the *main* method a *SocketServer* object is created and then a while loop is used to keep the application alive and accept new client connections. In the *run* method, line 40 the entire request string is read, line by line.

Additional information can be found as comments in the code.

```
package com.admfactory.http;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.StringTokenizer;
```

```

public class HTTPServer extends Thread {

    private Socket client = null;
    private BufferedReader inClient = null;
    private DataOutputStream outClient = null;

    public HTTPServer(Socket cl) {
        client = cl;
    }

    public void run() {
        try {
            System.out.println("The Client " + client.getInetAddress() + ":" +
client.getPort() + " is connected");

            inClient = new BufferedReader(new
InputStreamReader(client.getInputStream()));
            outClient = new DataOutputStream(client.getOutputStream());

            String requestString = inClient.readLine();
            String headerLine = requestString;

            /** sometimes headerLine is null */
            if (headerLine == null)
                return;

            StringTokenizer tokenizer = new StringTokenizer(headerLine);
            String httpMethod = tokenizer.nextToken();
            String httpQueryString = tokenizer.nextToken();

            System.out.println("The HTTP request string is ....");
            while (inClient.ready()) {
                /** Read the HTTP request until the end */
                System.out.println(requestString);
                requestString = inClient.readLine();
            }

            if (httpMethod.equals("GET")) {
                if (httpQueryString.equals("/")) {
                    /** return the home page */
                    homePage();
                } else if (httpQueryString.startsWith("/hello")) {
                    /** return the hello page */
                    helloPage(

```

```

httpQueryString.substring(httpQueryString.lastIndexOf('/') + 1,
httpQueryString.length()));
    } else {
        sendResponse(404, "<b>The Requested resource not
found.</b>");
    }
    } else {
        sendResponse(404, "<b>The Requested resource not found.</b>");
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

/**
 * Method used to compose the response back to the client.
 *
 * @param statusCode
 * @param responseString
 * @throws Exception
 */
public void sendResponse(int statusCode, String responseString) throws
Exception {

    String HTML_START = "<html><title>HTTP Server in Java</title><body>";
    String HTML_END = "</body></html>";
    String NEW_LINE = "\r\n";

    String statusLine = null;
    String serverdetails = Headers.SERVER + ": Java Server";
    String contentLengthLine = null;
    String contentTypeLine = Headers.CONTENT_TYPE + ": text/html" +
NEW_LINE;

    if (statusCode == 200)
        statusLine = Status.HTTP_200;
    else
        statusLine = Status.HTTP_404;

    statusLine += NEW_LINE;
    responseString = HTML_START + responseString + HTML_END;
    contentLengthLine = Headers.CONTENT_LENGTH + responseString.length() +
NEW_LINE;

    outClient.writeBytes(statusLine);
    outClient.writeBytes(serverdetails);

```

```

        outClient.writeBytes(contentTypeLine);
        outClient.writeBytes(contentLengthLine);
        outClient.writeBytes(Headers.CONNECTION + ": close" + NEW_LINE);

        /** adding the new line between header and body */
        outClient.writeBytes(NEW_LINE);

        outClient.writeBytes(responseString);

        outClient.close();
    }

    /**
     * Method used to compose the home page response to the client
     *
     * @throws Exception
     */
    public void homePage() throws Exception {
        StringBuffer responseBuffer = new StringBuffer();
        responseBuffer.append("<b>HTTPServer Home Page.</b><BR><BR>");
        responseBuffer.append("<b>To see the Hello page use :  
http://localhost:5000/hello/name</b><BR>");
        sendResponse(200, responseBuffer.toString());
    }

    /**
     * Method used to compose the hello page response to the client
     *
     * @throws Exception
     */
    public void helloPage(String name) throws Exception {
        StringBuffer responseBuffer = new StringBuffer();
        responseBuffer.append("<b>Hello:</b>").append(name).append("</b><BR>");
        sendResponse(200, responseBuffer.toString());
    }

    /**
     * class used to store headers constants
     *
     */
    private static class Headers {
        public static final String SERVER = "Server";
        public static final String CONNECTION = "Connection";
        public static final String CONTENT_LENGTH = "Content-Length";
        public static final String CONTENT_TYPE = "Content-Type";
    }

```

```
/**
 * class used to store status line constants
 */
private static class Status {
    public static final String HTTP_200 = "HTTP/1.1 200 OK";
    public static final String HTTP_404 = "HTTP/1.1 404 Not Found";
}

public static void main(String args[]) throws Exception {

    ServerSocket server = new ServerSocket(5000, 10,
InetAddress.getByName("127.0.0.1"));

    System.out.println("HTTPServer started on port 5000");

    /**
     * loop to keep the application alive - a new HTTPServer object is
     * created for each client
     */
    while (true) {
        Socket connected = server.accept();
        HTTPServer httpServer = new HTTPServer(connected);
        httpServer.start();
    }
}
}
```

Console output

The console will output for each client that is connecting something similar with the following printout (all the headers, the path, and some additional information).

```
HTTPServer started on port 5000
The Client /127.0.0.1:65169 is connected
The Client /127.0.0.1:65168 is connected
The HTTP request string is ....
GET / HTTP/1.1
Host: localhost:5000
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/57.0.2987.133 Safari/537.36
```

```
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: en-US,en;q=0.8,ro;q=0.6
Cookie: JSESSIONID=d2nnb0znqcy11s6hgZ86x5whq; _ga=GA1.1.1963935862.1484820976
AlexaToolbar-ALX_NS_PH: AlexaToolbar/alx-4.0.1
The Client /127.0.0.1:65170 is connected
```

Page output

Home page

Hello page

Page not found

References

- [ServerSocket](#)

Share this article



Recent Posts

- [How to rename a git branch](#) December 11, 2018
- [How to remove a remote from git](#) August 5, 2018
- [How to remane a remote from git](#) August 3, 2018
- [How to add a remote to git](#) August 1, 2018
- [Java Custom Annotations Example](#) July 26, 2018
- [How to execute a shell command from Java](#) June 20, 2018
- [How to generate UUID / GUID in Java](#) June 2, 2018
- [How to change the hostname in Ubuntu](#) June 1, 2018
- [How to change Tomcat default port](#) April 4, 2018
- [Maven local repository location](#) April 3, 2018
- [How to read XML file in Java using SAX Parser](#) April 2, 2018
- [Tomcat default administrator password](#) April 1, 2018
- [How to check if a file exists in Golang](#) March 15, 2018
- [How to marshal a JAXB object into org.w3c.dom.Document](#) March 10, 2018
- [How to Install GoLang 1.10 on Ubuntu](#) March 1, 2018



