# Web Server Built in Java

Emir D. & Peng J.

# 2. HTTP

- Basic Outline

-Get a HTTP request from browser.

-Process this request.

-Pass it to response class.

-Generate output to the browser.

# HTTP

- How we will run our server and get the request from browser?

main.xml | HTTPResponse.java | ConnectionHandler.java | HTTPRequest.java | Main.java

```java
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {

    private final static Logger logger = Logger.getLogger(Main.class.getName());

    ServerSocket serverSocket;

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

        new Main().runServer(); //to avoid any problem with static fields
        logger.setLevel(Level.INFO);
    }

    public void runServer() throws Exception{
        System.out.println("Server is started...");
        serverSocket = new ServerSocket(9876);

        //for accepting requests
        acceptRequests();
    }

    private void acceptRequests() throws Exception{

        logger.info("Server is ready to accept request.");

        while(true){ //we have to accept all the request

            //connection to client is in the form of socket which contain the stream for input
            //and output
            Socket s = serverSocket.accept();
            ConnectionHandler ch = new ConnectionHandler(s);

            //ch is the thread, so
            ch.start(); // this will call the run method automatically
        }
    }
}
```

Problems | Javadoc | Declaration | Console | LogCat

No consoles to display at this time.

Writable | Smart Insert | 4 : 33

# HTTP

- Accepting Browser Requests

main.xml | HTTPResponse.java | ConnectionHandler.java | HTTPRequest.java | Main.java

```java
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {

    private final static Logger logger = Logger.getLogger(Main.class.getName());

    ServerSocket serverSocket;

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

        new Main().runServer(); //to avoid any problem with static fields
        logger.setLevel(Level.INFO);
    }

    public void runServer() throws Exception{
        System.out.println("Server is started...");
        serverSocket = new ServerSocket(9876);

        //for accepting requests
        acceptRequests();
    }

    private void acceptRequests() throws Exception{

        logger.info("Server is ready to accept request.");

        while(true){ //we have to accept all the request

            //connection to client is in the form of socket which contain the stream for input
            //and output
            Socket s = serverSocket.accept();
            ConnectionHandler ch = new ConnectionHandler(s);

            //ch is the thread, so
            ch.start(); // this will call the run method automatically
        }
    }
}
```

Problems | Javadoc | Declaration | Console | LogCat

No consoles to display at this time.

Writable | Smart Insert | 4 : 33

# HTTP & Multithreading

- Handling the connection we established

```java
//this class basically handles all the connection which contains the requests
    public class ConnectionHandler extends Thread {

        Socket s;

        //for sending the output to client
        PrintWriter pw;

        //for getting the input from client
        BufferedReader br;

        //constructor
        //which accepts a socket
        public ConnectionHandler(Socket s) throws Exception{

            this.s = s;
            br = new BufferedReader(new InputStreamReader(s.getInputStream()));
            pw = new PrintWriter(s.getOutputStream());
        }

        //thread class contains a method run which is called automatically when we start the
        //thread
        //in this method we have to read the request and give the response

        @Override
        public void run() {
            try {
            //here we get the request string and give this string to
            //HttpRequest class

            String reqS = "";

            //from br we have to read our request
                while(br.ready() || reqS.length() == 0){

                    reqS += (char) br.read();
                } //check this afterwards

            System.out.println(reqS); //for display

            HTTPRequest req = new HTTPRequest(reqS);

            //now we pass the httpReq object to httpresponse class for getting the response

            HTTPResponse res = new HTTPResponse(req);

            //write the final output to pw
            pw.write(res.response.toCharArray());

            pw.close();
            br.close();
            s.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
```

# HTTP Request

- Handling the Request

main.xml | HTTPResponse.java | ConnectionHandler.java | HTTPRequest.java | Main.java

```java
public class HTTPRequest {

    //first line contain 3  parts
    // 1-request type 2-file name 3-http version
    // now we just make use of the file name
    String filename;

    //we have to create a constructor that accepts a string
    public HTTPRequest( String request ){
        //now we have the request only 1at line is important to us
        String lines[] = request.split("\n"); // get all the lines of request separatly
        filename = lines[0].split(" ")[1]; //get the filename
    }


}
```

# HTTP Response

- Handling the Response

main.xml | HTTPResponse.java | ConnectionHandler.java | HTTPRequest.java | Main.java

```java
import java.io.File;


public class HTTPResponse {

    HTTPRequest req;

    // this is the final response
    String response;

    // root path of the server
    String root = "/Users/pjin/Dropbox/Course Files/Sem6/CSE489/Final Project/root";

    public HTTPResponse(HTTPRequest request) {
        req = request;
        // now we have to open the file mentioned in request
        File f = new File(root + req.filename);

        try {

            response = "HTTP/1.1 200 \r\n"; // version of http + status code 200 means it's all good

            response += "Server: Emir & Peng's Java Server/1.0 \r\n"; // identity of server
            response += "Content-Type: text/html \r\n"; // response is in html format
            response += "Connection: close \r\n"; //
            response += "Content-Length: " + f.length() + " \r\n"; // length of response file
            response += "\r\n"; // after blank line we have to append file data

            //to read this file
            FileInputStream fis = new FileInputStream(f);
            int s;
            while ((s = fis.read()) != -1) { // -1 means end of file
                response += (char) s;
            }
            fis.close();
        } catch (FileNotFoundException e) {
            // if we don't get file then error 404
            response = response.replace("200", "404");
        } catch (Exception e) {
            // if other error the 500 internal server error
            response = response.replace("200", "500");
        }
    }

}
```

# 3. Logging

- Keep track of the processing of the server.
- Record the status of crucial steps, including INFO, WARNING, etc.
- Save the logging information as a log file for future usage.

Main.java | ConnectionHandler.java | HTTPRequest.java | HTTPResponse.java

```java
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.FileHandler;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

public class Main {

    private final static Logger logger = Logger.getLogger(Main.class.getName());
    private static FileHandler fh = null;

    ServerSocket serverSocket;

    public static void init(){
        try {
            fh = new FileHandler("logger.log", false);
        } catch (SecurityException | IOException e) {
            e.printStackTrace();
        }
        Logger l = Logger.getLogger("");
        fh.setFormatter(new SimpleFormatter());
        l.addHandler(fh);
        l.setLevel(Level.CONFIG);
    }

    //entry point
    public static void main(String[] args) throws Exception {
        Main.init();
        new Main().runServer(); //to avoid any problem with static fields
    }

    public void runServer() throws Exception{
        logger.info("Server is started...");
        serverSocket = new ServerSocket(7777);

        //for accepting requests
```

```java
    // root path of the server
    String root = "/Users/pjin/Dropbox/Course Files/Sem6/CSE489/Final Project/root";

    public HTTPResponse(HTTPRequest request) {
        req = request;
        // now we have to open the file mentioned in request
        File f = new File(root + req.filename);

        try {

            response = "HTTP/1.1 200 \r\n"; // version of http + status code 200 means it's all good

            response += "Server: Emir & Peng's Java Server/1.0 \r\n"; // identity of server
            response += "Content-Type: text/html \r\n"; // response is in html format
            response += "Connection: close \r\n"; //
            response += "Content-Length: " + f.length() + " \r\n"; // length of response file
            response += "\r\n"; // after blank line we have to append file data

            //to read this file
            FileInputStream fis = new FileInputStream(f);
            int s;
            while ((s = fis.read()) != -1) { // -1 means end of file
                response += (char) s;
            }
            fis.close();
            logger.info("Successfully created response for user.");
        } catch (FileNotFoundException e) {
            // if we don't get file then error 404
            response = response.replace("200", "404");
            logger.warning("Error 404: Can't find the requested file.");
        } catch (Exception e) {
            // if other error the 500 internal server error
            response = response.replace("200", "500");
            logger.warning("Server error 500.");
        }
    }

}
```

# Demo

# More to Add

- Authentication:
- Server-side Scripting:

# Questions?

# Thank You