**OPERATING SYSTEM PROJECT**

**SYNOPSIS**

**[Multi-Threaded Web Server](http://pages.cs.wisc.edu/~dusseau/Classes/CS537-F07/Projects/P2/p2.html)**

**Server-Side Program:** When a new client is connected, and he sends the message to the server.

**1. Server class:** The steps involved on the server side are similar to the article [Socket Programming in Java](https://www.geeksforgeeks.org/socket-programming-in-java/) with a slight change to create the thread object after obtaining the streams and port number.

* **Establishing the Connection:** Server socket object is initialized and inside a while loop a socket object continuously accepts an incoming connection.
* **Obtaining the Streams:** The [inputstream](https://www.geeksforgeeks.org/java-io-inputstream-class-in-java/) object and [outputstream](https://www.geeksforgeeks.org/java-io-outputstream-class-java/) object is extracted from the current requests’ socket object.
* **Creating a handler object:** After obtaining the streams and port number, a new clientHandler object (the above class) is created with these parameters.
* **Invoking the [start()](https://www.geeksforgeeks.org/start-function-multithreading-java/) method:** The start() method is invoked on this newly created thread object.

**2. ClientHandler class:** As we will be using separate threads for each request, let’s understand the working and implementation of the ClientHandler class implementing Runnable. An object of this class acts as a Runnable target for a new thread.

First, this class implements Runnable interface so that it can be passed as a [Runnable](https://www.geeksforgeeks.org/runnable-interface-in-java/" \l ":~:text=java.-,lang.,run() method of Runnable .) target while creating a new [Thread](https://www.geeksforgeeks.org/java-lang-thread-class-java/).

Secondly, the constructor of this class takes a parameter, which can uniquely identify any incoming request, i.e. a **Socket.**

Inside the **run()** method of this class, it reads the client’s message and replies.