Lab 6: Maintaining State in a Web Server

Objective:

In this lab you will complete a Java program to be an iterative web server that serves dynamic documents. You will examine two techniques for maintaining state. The first method maintains state in the Java program and the second uses cookies.

Instructions:

1. Open a new project in Eclipse. Create a simple web page that includes at least a title and your name(s). You may use Word or you may use any text editor (Notepad or the Eclipse text editor) to create a text file with HTML code. Store the web page in the project. If you need help here is a simple web page: [simplepage.htm](#)

2. WebServer.java is a simple web server that serves a single document to the requesting browser. Modify the web server to serve the web page you created in Step 1.

   Note the client-server interaction required by HTTP: Once the connection is made the server sends a header to the client consisting of the following lines:
   
   HTTP/1.0 200
   Content-Type: text/html

   followed by a blank line. Following the header the server sends the HTML text.

   Run the server then open a browser and request the page from the server. Record what is displayed in the browser and what the server displays in the console.

3. Modify the server to insert the current time/date (properly labeled) and the IP address (properly labeled) of the source of the request in the web page. Note: don’t generate the time-stamp until after the connection is made. Have the server display the current time/date (the one sent) in the console as well.

   Further modify the server to be an iterative server, that is, the server should accept connections one after another as they come in. To do this, put the block of statements starting with waiting for a connection inside an infinite loop (use

   ```
   while (true) {
   ... }
   ```

   Run the modified server then open a browser on at least two different lab machines, request the page from the server, and record the results obtained (in the browser). Also record what appears in the server’s console.

   Don’t forget to explicitly terminate your server.
4. To illustrate the techniques for maintaining state across connections consider the problem of having the server include the number of visits from that host in the page that is served.

The first technique will have the server maintain the number of connections from each IP address in a HashMap. A HashMap is a data structure available in Java that is well-suited for storing this data. Here is a “snippet” of code that checks whether an IP address is present in the map (it won’t be present if this is the first request from this host). If it is present then the corresponding count (Counter object) is incremented. If it is not present a new entry is placed in the map. You will need to add the file Counter.java to your project for this to work.

Modify the server to include the code snippet to track the number of connections from each IP address. Have the server insert a statement in the web page served that informs the client of the visit number from that host – e.g., visit 1, visit 2, etc.

Run the modified server then open a browser on at least two different lab machines and make several requests of the page from the server from each machine. Verify that the server is working correctly. Record the results obtained (in the browsers).

(Extra Credit: You will probably find that the number of visits goes up by 2 with each page refresh request. Why is this the case? Can you fix it?)

Print a copy of the server and hand it in with your lab.

5. Now you will modify the server to maintain the same sort of state information using cookies.

Cookies are small name/value pairs that are sent as part of the HTTP headers. The server is responsible for managing/updating the cookie. The browser stores the cookie and is responsible for including it in the header of any HTTP request of that server. Note in particular that the first request carries no cookie since the browser doesn’t yet know that the server requires one. When the server serves the page the first time it sends the cookie. In any subsequent requests the browser must provide the cookie. (This assumes the cookie is persistent. Each cookie has an expiration date and the browser stores the cookie until that date.)

We will use a cookie to track the number of visits from a browser. The cookie will be of the form

numvisits=n

where n is the number of visits.

The server needs to check each request for the presence of the cookie. If the cookie is present there will be a line of the form

Cookie: numvisits=n

in the header of the request. The server needs to include a line

Set-Cookie: numvisits=n; Expires=<expiration date>

in the header of the response (include it between the first and second lines of the header), where n is 1 if this is the first visit from the browser and n is one larger than the value in the cookie if it is any subsequent visit. <expiration date> is the expiration date of the cookie.

Here’s how to do this:
Modify the server by first removing or commenting out the code involving the HashMap. Now modify the code that displays the request from the browser to look for a line of the form

Cookie: numvisits=n
If there is such a line then parse the number of visits \((n)\) from the line. Modify the header sent to the browser to include a line of the form

\[
\text{Set-Cookie: numvisits}=n; \text{Expires}=\text{Sat, 30-Dec-2034 00:00:00 GMT}
\]

where \(n\) is 1 if there was no cookie in the request from the browser and 1 larger than the number of visits in the cookie otherwise.

Be sure to include the visit number in the web page delivered to the browser (as you did in the last step).

Run the modified server then open a browser on at least two different machines and make several requests of the page from the server from each machine. Verify that the server is working correctly. Record the results obtained (in the browsers). Also record the results displayed in the console by the server.

(Extra Credit: As in Step 4 does the visit number go up by 2? Why is this the case? How can you fix it?)

Print a copy of the server that uses cookies and hand it in with your lab.

6. As a final step of this lab, discuss the two approaches you have used to maintain the state of a web server. What is an advantage of each approach?

Help Policy:

Help Policy in Effect for This Assignment: Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 254.

You may use the following materials produced by other students: materials produced by members of your group.