Lab Submission Date on BRS – April 11th, 2021 (23:59)

Assignment 3: Who's the WebLogger???

The demonstration week for this assignment is the week of April (12-16th), 2021. **No demonstration will result in zero mark.** This lab assessment is worth 30 marks (16% weightage).

Objectives

Build a project to log access to your website. This project lets you log users who access your website by access time, authentication type, username (if they've logged in), user IP address, the URL they accessed on your site, their browser type, the milliseconds they were there for, and so on. All without their knowledge.

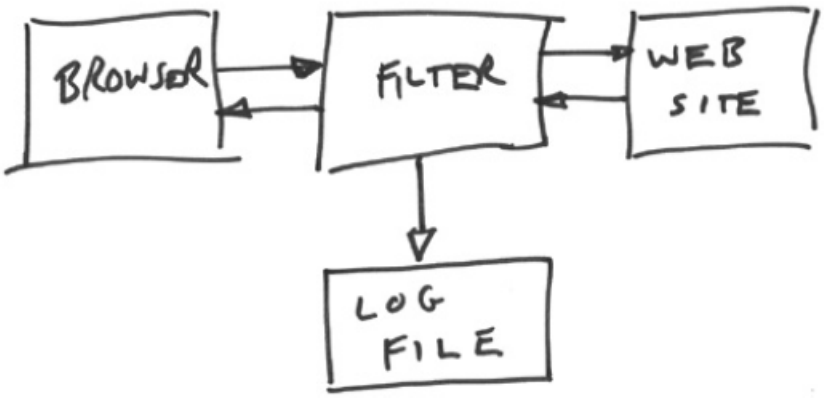

Fig 1: Block Diagram illustration

Using WebLogger, you can track users as they move around your site. In fact, you can even block access if you want. What does the user see? Nothing at all, as shown in the figure 2, where the user is accessing a page called **target.jspexcept** for the text WebLogger has added to the bottom of the page. Unless that added text informed the user that he was being tracked (and, of course, you can remove that text from WebLogger), he would have no way of knowing it.

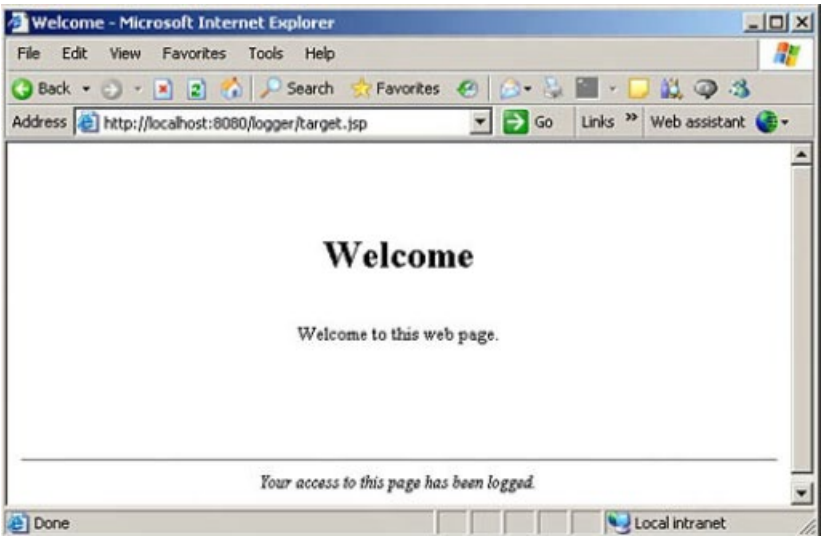

Fig 2: WebLogger on the go!!

Here's the information that's logged for each user:

- Access time
- Authentication type
- Username
- User IP address
- URL accessed
- Browser
- Milliseconds used

Here's what the report will look like. You can have this logged in the web server's logs or in a separate log file of your own **SelfCreatedWebLogger** lets you do either:

- User access at Mon Mar 01 02:46:52 EST 2021
- Authentication type: BASIC
- User name: Gud
- User IP: xyz.x.x.x
- Accessing: loggertarget.jsp
- Host: xyz.x.x.x
- Browser: Internet Explorer
- Milliseconds used: 111

To make what it does possible, WebLogger uses a filter. Creating a new filter is just a Java class that implements the **javax.servlet.Filter** interface. You can see all the methods of this interface in table1 below.

Method	Does This
`void destroy()`	Called to indicate that a filter is being destroyed
`void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)`	Called when control is being passed to the filter
`void init(FilterConfig filterConfig)`	Called to indicate that a filter is being initialized

*Table 1: The Methods of the **javax.servlet.Filter** Interface*

Creating a Simple Filter

WebLogger works with both the request object holding the data sent from the browser and the response object sending data back to the browser. So how do you work with, for example, the response object to send data back to the browser? Say that you want to filter access to a JSP page named ***simpleFilter.jsp*** and that you want a filter to add some text to the output of this JSP sent back to the browser. Here's what that JSP page looks like as it stands, all this page does is to display the text "Using a filter" in an <H1> HTML header:

```
<HTML>
<HEAD>
<TITLE>Using a filter</TITLE>
</HEAD>
<BODY>
<H1>Using a filter</H1>
<BR>
</BODY>
</HTML>
```

Because you have access to the data sent to and from this JSP page in a filter, you can display your own text in the web page, as you can see in Figure 3. In this figure, a filter attached to simpleFilter.jsp wrote the text "The filter wrote this," just as WebLogger added the warning about logging user access shown at the bottom of Figure 2. Note that the URL still points to simpleFilter.jsp.

Lab Submission Date on BRS – April 11th, 2021 (23:59)

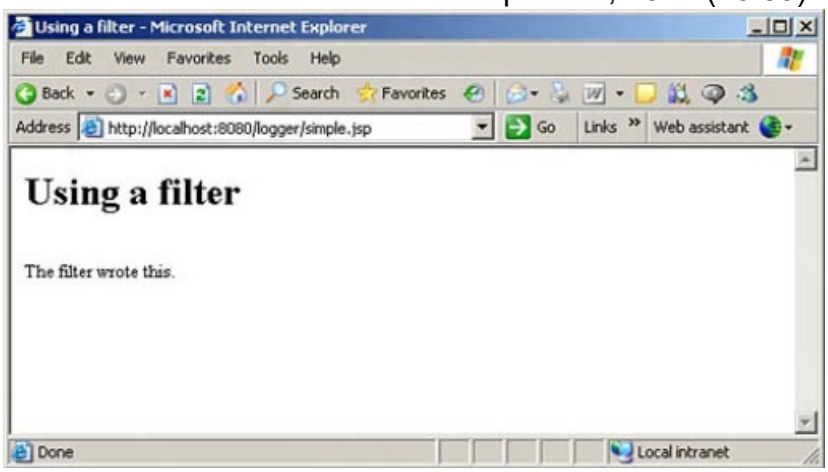

Figure 3: Adding text to a webpage using a filter

Tasks Involved

1) Call the filter **SimpleFilter.java** and implement the interface. For this implementation, you will need ***doFilter***, ***init*** and ***destroy*** methods.

2) Call the ***FilterChain*** object's ***doFilter*** method, the next filter, if there is one, is called, followed by any other filters, and then the filtered resource itself. Control returns in the reverse order back to your filter, where you get your hands on the response object again. And that lets you write text in the response sent back to the browser, as WebLogger does. To start, you configure the MIME type of text you're going to send to the browser, setting it to "text/html".

3) Add your own text to the webpage being sent back to the browser.

4) Configure the web server.

5) Restrict the access based on password. For instance, one of the most popular uses of filters is to restrict access to web resources based on password; take a look at Figure 4, where an HTML page, login.html, is asking the user for his password.

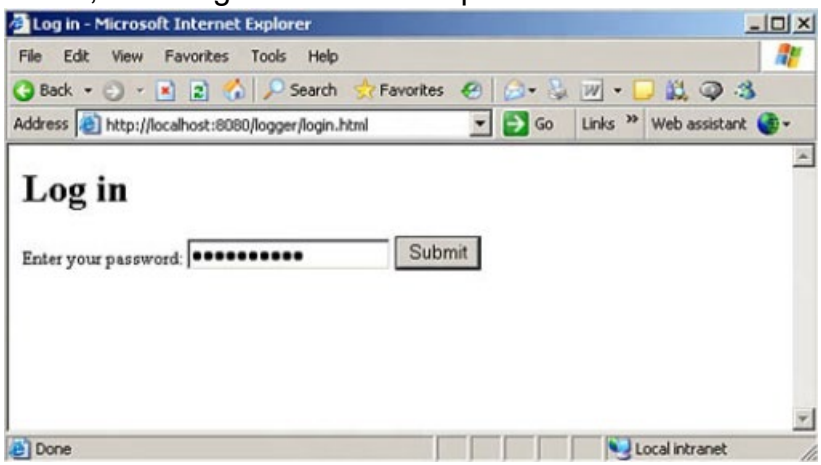

Figure 4: Restricting access based passwords using a filter

If the user enters the correct password (Figure 5):

Figure 5: Gaining access with correct passwords

6) If the user enters the wrong password, the filter denies access and displays the error page you see in Figure 6.

Figure 6: Denying access due to an incorrect password

7) Having absorbed all the filter technology, it's time to build WebLogger.

Figure 7: Reading a password

If the user enters the right password, he'll see **target.jspas** and his information will be logged. The **target.jsp** file will display a message, like this:

```
<HTML>
<HEAD>
<TITLE>Welcome</TITLE>
</HEAD>
<BODY>
<CENTER>
<BR>
<BR>
<BR>
<H1>Welcome</H1>
<BR>
Welcome to this web page.
</CENTER>
</BODY>
</HTML>
```

8) After successful login; collect and log user data.

Goals

To successfully complete this assessment and obtain all the marks, you must:

a) Assure that filter that grabs data about users who access your website without their knowledge, unless you tell them they're being logged.

b) To log accesses to a web resource, you don't have to change that web resource; you can use a filter in Java-enabled web servers. To create a filter, you have to create a Java class that implements the Filter interface, which includes the init, doFilter, and destroy methods. When the user accesses the resource you're filtering, the doFilter method gets control, gaining access to the request object sent to the resource and the response object the resource sends back to the browser.

c) WebLogger uses the request object to get its information about the user, reading the URL the user is accessing, the user's IP address, hostname, and browser type. If the user is logged in using HTTP authentication, WebLogger records the type of authentication used and records the username. It also logs the time of the user's access, as well as the time he spent accessing the web resource you're filtering.

d) Be able to compile and execute this code from the command line. You will be called upon to do so, *so have your command prompt ready during your lab session.*

Marking Scheme

Item	Penalty
Authentication rules doesn't confirm	5 marks
Filter methods implementation is unsatisfactory	10 marks
No message displayed by the .jsp files	10 marks
No access data logs	15 marks
Other specification violations	3 marks
Code violates Java formatting, variable naming conventions, comment headers	2 marks
Missed demonstration	30 marks
Ran in Netbeans, but not from the command line.	5 marks
Code does not compile or crashes	up to 30
Total	**____ Out of 30.**

Lab Deliverables

- You must submit your .java file in the assigned submission folder on BRS section 300 (activities → Assignments). We don't need your entire project folder, a java file and your image file (you may compress them with .zip, but .zip only) is what we need.
- You will need to demo this code in your lab session, the week of April 12-16th, 2021. During demonstration, you must be able to answer any questions related to the assessment.
- Create a folder called algonquinUserID1_algonquinUserID2_A1 (example, "mynam00123_yourna45678_A1") that represents the two members of your team. **Only one submission per group is required.** Both needs to be present during the demonstration. Absentees will not receive any grade.

Helpful Tips

- There are a few edge cases in the standard assignment; don't wait until the last minute to get started on it.
- Errors may be possible. Be sure your code handles those errors and exceptions without crashing.
- Your code will be tested against standard UTF-8 plain text files. Be sure to account for line breaks in the original file if required.