

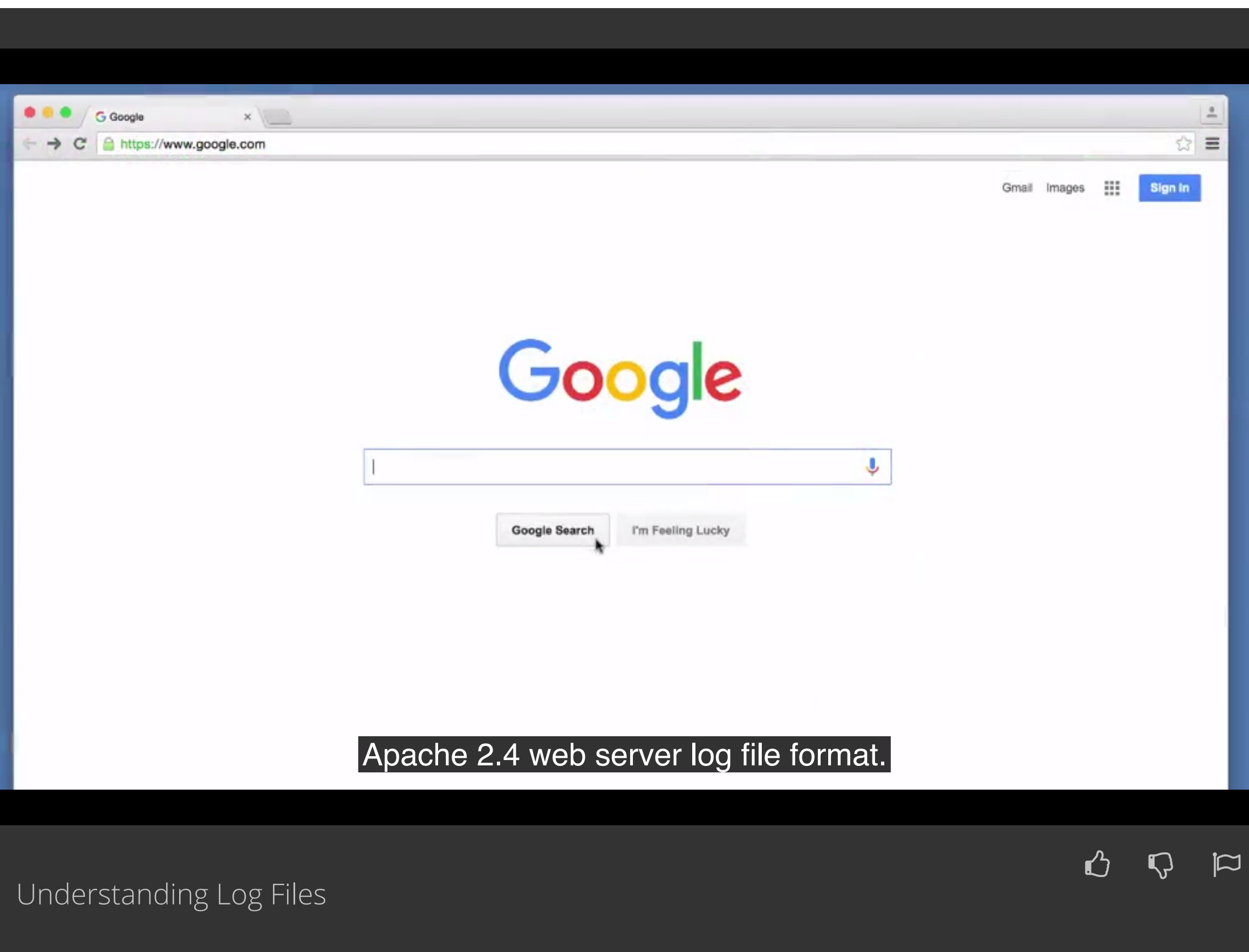
Reading Log Files

Module Learning Outcomes / Resources	10 min
Introduction	1 min
Understanding Log Files	6 min
LogEntry Class with toString	5 min
Parsing Log Files	3 min
Summary	45 sec
Programming Exercise: Reading Log Files	10 min
Practice Quiz: Reading Log Files	3 questions

Finding Unique IP Addresses

Counting Website Visits

Review



Understanding Log Files

Have a question? Discuss this lecture in the week forums.



Interactive Transcript

Search Transcript

English

0:03

Welcome back. Now that you know the importance of web server logs, it's time for you to start thinking about writing code that deals with them. The first thing you're going to want to do is to be able to rename a web server log file and represent the information in it in Java objects. To do this, you need to think about two things. The first is what does all of the information in the web server log mean? And the second is how do you represent it in a Java class? Here, you can see one entry from a web server log file. It has a lot of information and it's not readily apparent what each piece means. To find out what it means, you would want to read documentation about this web server.

0:46

This particular data came from the web server log for an Apache 2.4 web server. So to find out more, you could go to Google and search for [Apache 2.4 web server log file format](#). If you do that, you'll end up with a lot of hits. And this first one here, gives you a link to the Apache documentation site. If you scroll down a bit, you'll find the information on the access log. That is the log of accesses to the web server, which is what this web log is and you will see that it has information about each of the pieces of this entry.

1:24

The first is the IP address, that is the address of the device on the internet, which made the web request that is logged here. The next two pieces are both dashes that indicates they are missing information. The first dash is for some information about who made this request, which you'll see the documentation says is unreliable. The users computer could lie about who they are. This second dash is for the user name if they're logged in with HTTPF authentication, if they typed in a username and password on the website.

2:00

The next piece of information is the date and time when the request was made.

2:05

Next, you have the request itself including what type of request it was. In this case, get, where they're asking for a particular web page and then what page they asked for. Next is the status. Here it says, 200, which indicates success. There are many other statuses, which indicate success or failures. You may be familiar with 404, which is the very well-known status code that indicates that the requested page was not found. Finally, is the number of bytes that the server replied with. How much data it sent back to fulfill this request?

2:43

Now that you've read the documentation and understand what each of these pieces of information mean, it's time to think about how to represent them in a java class. The first thing you need to think about is what type of information each of these is. For the IP address, you could use a string since you're interested in just the text of that field. Java does have a built-in class for IP addresses, which would give us some more features if we wanted to actually connect to that address, for example. But we don't need that functionality and we don't need to worry about the complexity that would introduce right now. We don't need to represent the two fields that we've omitted that have no useful information. We do, however, want to represent the date. You could use a string for that in which case, you would just have its text or you could use the built-in Java Date class, which understands what dates and times are and how they relate to each other. So you could check if one time is before or after another time.

3:43

For the request, you can just use a string. And for the status and number of bytes, those are both numbers, so you can use an int.

3:52

Now that we've gotten through these types, it's time to turn this into some Java code. Here you can see the start of a Java class for a LogEntry, we've declared a public class LogEntry and written fields based on the types that we just discussed.

4:10

You should now think, should each of these fields be public or private? Remember that if a field is public, any piece of code can access it. If a field is private, only code within this particular class can access it.

4:25

In this particular case, it makes sense to have each of these be private and to design your class to be immutable. Remember, from earlier when we learned about strings that immutable means, you cannot modify an object once you create it. So you're going to write this class, so that each of these fields will be set in its constructor, but can only be read. To make anything able to read these fields, you'll write a public getter or accesser method such as these, which will just return the value of that field, but there will be no way to set the value of the field once its constructed. Speaking of constructing, you need to write a constructor for this class. There are two ways you could do it. The first is to take in this entire string and then I have the constructor pull it apart into each of these individual pieces and fill in the fields or instance variables of the class.

5:23

The other is to have the constructor take each piece of information separately and simply initialize the fields of the object. We're going to have you do the second one, just making a constructor that looks like this, which is going to fill in the fields based on the information passed down with each piece being passed separately. Why would you do it this way? Well, this gives you a little more flexibility if you wanted to create one of these objects from some other source of information you could do so. It turns out that pulling this line apart is actually a little bit tricky, so we're gonna give you the code for that. It's a little bit ugly and we'll package it up into a nice method for you, so you can just use it to read the file.

6:07

Here is the final log entry class with all of the things you just learned about. You would use this to represent one of these log entries as you work with it. So your next task is going to be to use this class, the code we give you to split this up into each separate piece and put them together to make code that's going to read the entire log file.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt

Would you like to [help us translate](#) the transcript and subtitles into additional languages?