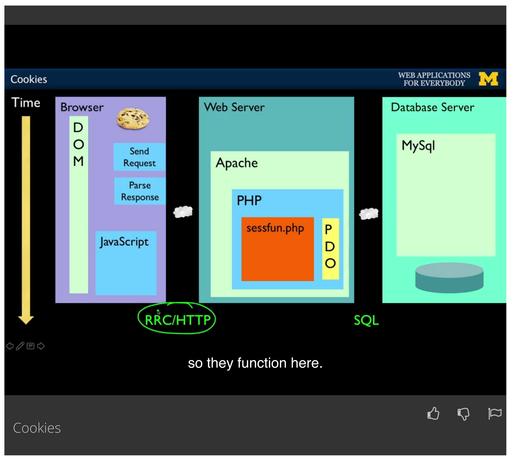
Lecture Materials

Assignment

Bonus Materials



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

Interactive Transcript

English

0:07
So now we're going to talk a little bit about what we call cookies and sessions. The idea of cookies and sessions is that web servers need to know which of many browsers they are communicating with. What we've done so far is we really have a web server, a database server and a browser, all on the same computer and so it's all you. But in reality, you have a database server and a web server and like a thousand users and which of those browsers is it talking to? And we do this so much when we log in and it knows who we are and it knows what our shopping cart is. We take all this for granted. But in this lecture, we are going to dig in to the mechanisms that make all that possible. It just so automatic. It's another one of those things that if you don't know what it is, it's like wow, I didn't realize it was that complex. And then once you understand, it's like of course, it's no big deal. Okay. So the first thing we're going to talk about are what are called cookies, HTTP cookies. Cookies are part of the HTTP protocol, so they function here. They're not stored in the server, they're not stored in the database. And so the idea of a cookie is it's a bit of data that's stored in your browser. It's key value paired. It's almost like a PHP key value array that is stored in the browser. So, you can say X equals 2, except it's the server's data. So this is a bit of server data that's stored in the browser. Those are called cookies. And so the idea is that you start with nothing in the cookie and then it comes in, the request comes in. And then what happens is added onto this request is like X=3 and then it stores that X=3 and then that comes out. Then the next request comes in and then it adds X=3 to that request. Just like get data and post data, cookie data is added to request. The difference is unlike get data and post data, once that request is done, the $_get and the $_post are gone. The cookie data comes in on every request and so you could have like a thousand requests and they will all have this cookie X set to three. And then if later, we set that to X=4, it overwrites it and then the next thousand requests will say X=4. So it's just a way to say here's a key value bit of data and please send it back to me on every request. Okay. So this is a really simple concept and it solves the problem, as I mentioned of when you have a web server, that's talking literally to thousands of different people, all with different identities. Some have credit cards that might be a shopping cart, we might be ordering stuff and we need to mark the browsers. So that's what we're doing, we're marking the browsers and we're picking a random number and we're going to give them a mark. In the earliest of days, 1990, the request response cycle had no state and all browsers looked identical. But now, when we're going to log in, we're going to have identities. Facebook knows who you are. YouTube knows who you are. Cookies are the way that we do all that stuff. So, cookies are a piece of data that's chosen by the web server, the key and the value are set by the web server. Sent to the browser. Stored in the browser and then sent back. Now, cookies are not the same as sessions and they're not the same as being logged in. You will find that it's quite often that you visit a web page and they're going to set a cookie, long before you log in. It just often on the very first request, it's going to set a cookie. It's going to say do I have this X cookies set? If not, I'll set it every time. And so and again, I've drawn this picture a couple of times. You go in, it sets a cookie. It sends a page back and a cookie X=2. And that cookie is X stored over here and then on all the requests, the cookie comes back in. That's what this picture is trying to show you. So they're marked as to which web addresses they come from. And so, you only get the cookies, if you're a server, you only get the cookies back from the server that came from that server. They're kind of stove, stove piped and isolated, siloed in the browser, so that you can't have one server looking at a different server's cookies. They also have an expiration date. There's also a maximum size of them but usually we don't put very much in and we don't run into the maximum. So you can have a cookie that lasts for a couple of years or we can have what's called a session cookie which is a cookie that goes away as soon as the browser is closed. We tend to use session cookies for things like logging in and logging out, although some systems use long term cookies to do log in and log out. Those tend to be like signed and more complex. So basically, PHP has really excellent support for cookies. Just like when we have $_get, $_post. These are the things where it's coming in. You got get data that ends up in $_get. You've got post data that ends up in $_post and cookie data ends up in $_COOKIE and it's super global just like all those other things and so that's pretty cool. We can say what were the cookies that were sent to us and they're all sitting there in $_COOKIE and it's a key value array. Like what could be easier? So, we can ask and again we don't do any work, $_COOKIE is defined. And $_COOKIE is defined if we have previously set a cookie. And so what we're going to do here in this particular bit of code is we're going to check to see if there is a zap key inside of cookie. We would have had to set it. So, it'll start by being false the first time through and if it is, we'll set zap to be 42 with an expiration. So that's an hour from right now. So that's time()+3600 is an hour from right now. And then we'll print the contents of this cookie array and then we'll click this over and over and over again. And so, the first time that this happens is you'll notice that there is no cookie. We're going into a fresh browser. We hit the cookie thing. There is no cookie. But in the response, we see a set cookie zap= 42 with an expiration date and away we go. And so this is coming back from the server to the browser. And so now the browser has to set this cookie and the browser has to send us back this cookie on every future request. So, if you go into application cookies and here you can see this zap. Now there's other cookies that you're going to find, log in cookies, et cetera. But this is storage in the browser. This is in the browser storage, basically, and the cookie has been set. When it receive that, it set this and so there's a little database in your browser that is the cookie values. Then, the next time you hit the statement, it's going to send the cookie in. So the other thing I showed you was response headers. This is the request header. So the cookie says oh hey I just took those two key value pairs, I'm going to send them to you on this cookie. And I got zap=42 and this could be hours later. Well not hours later because I only had an hour but it could be minutes later. And so it sends it in. So then PHP receives this and it puts that into the $_COOKIE. So the first time, we do a get request and that get request detects that there is nothing in $_COOKIE. So it sends back a set cookie, zap=42. So now, the browser has zap=42. Then we do a get request and then that shows up in this one. Then it does another get request and zap=42 keeps coming in. So this is time passing, right? Time passing. So, whatever we put, it stays there. And unlike $_get and $_post, $_get and $_post only last one thing and they don't come in the next one, right? Unless, you put them out somehow and cause them to come out. So up next, we're going to talk about sessions which is an addition to cookies. It's an in server data structure that is driven by cookies.

Downloads

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

Would you like to help us translate the transcript and subtitles into additional languages?