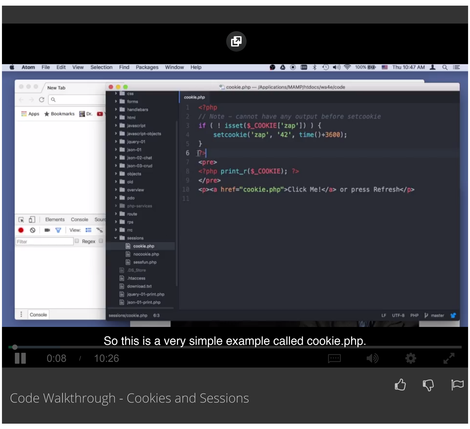
Lecture Materials

Assignment

Bonus Materials



Code Walkthrough - Cookies and Sessions

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

Interactive Transcript

English

0:00
Hello and welcome to Web Applications For Everybody. Today, we're going through some of the session and cookie examples. So this is a very simple example called cookie.php. Basically, what it does is cookies are key-value pairs that are stored in the browser. They're a bit of data that the servers can store on the browser and we store it using the setcookie. We can say, it's almost like a dictionary or associative array that lives on the browser. We say, zap equals 42. And then we give it an expiration time of now +3,600 seconds, which is an hour from now, but we're only going to do this if the cookie is not set. Now, this turns into a header on the response and the first time we do this, we'll notice that it's not set and then we'll set it. Okay? So you set it up. I got a fresh brand new browser and I will go straight to that bit of- straight to that bit of code and so we can see on our developer console. Hopefully, the cookie is going to be set. Here we come. Now, I can take a look so I look at the request. The request is a get to that URL. What happens here is now, because of this, this is a header, this is a response header that set the cookie. What that basically says now, is that every other request from this point forward, that cookie has to come in. Now, we can actually see these cookies under application, cookies, HTTP. Now, these other things have to do with automated thingies that are gathering data, whatever, but this is the zap cookie. Now I can hit delete here and get rid of it. Matter of fact, I can delete all these guys too. I just get rid of them so I just got rid of those cookies. These cookies belong to me. They're my browser. In the old days, you could actually change these. I don't know how you change these as easily as you'd- but you can change these. Cookies are not something- they're something the application sets on the browser, but they're not something that you'd think of as reliably owned by the browser. There is a cookie setting here and it set this cookie on that request response cycle, and that's because the current values of the cookies are copied into this super global $_COOKIE in PHP, and, because I was hitting it for the first time, there was nothing there, so we set it. But from now on, if I ask for this, again, with a refresh. It is not going to give me the cookie because it's going to notice that I've sent it. So, if I take a look at this, the cookie array has zap equals 42. And the way that was populated is, in the request header. The request header that the browser sent to the server, it sent a cookie header that says zap equals 42 so what PHP does, is it parses the zap equals 42, and puts it in this variable $_COOKIE, a super global like post, like get, etc. So, because it saw that it was already set, it didn't feel the need to set it again. Right? You'll notice there is no set cookie at least not for zap. There's all this other crap that got set again because I deleted it. I don't even know what those things are for but whatever. If I go and I take a look at my application and I look here, this one got set again on that request, but if you hit refresh over and over and again now, every time, every time, there's going to be, the cookie is going to be there and it's not going to have to be set again. If I delete it, I delete it and then I hit refresh and I take a look at the cookies, it's back because this code detected that it wasn't there. Once I deleted it, it doesn't get sent on the next request. Okay? That's cookies. That's cookies. Cookies are key value pairs that come in the browser. Now, let's talk about sessions. Sessions are not the same as cookies although they make use of cookies and so, in PHP, if we want to use sessions we say session_start. What happens? This is called sessfun.php - I won't hit enter yet. Session_start basically does a whole bunch of stuff. Session_start, if there is no session cookie and there is no session cookie here, we'll see one in a second, it actually picks a large random number for the Session ID, then sends that out as a cookie, session cookie, and then creates this super global called $_SESSION. Now the data in $_SESSION is not stored in the browser. It's actually stored in a magic place in the server in such a way that if you modify session, in a request, it actually stores the data in a server. Once you've done session_start, you have this magic variable that doesn't come from the client, doesn't come from the browser. It comes from the server on every request response cycle so you can put data into it and then you can take data out of it, and so it's pretty cool from one request to the next. OK. Let's go here to network to start and I'm still on cookie.php. Let's clear that. When I hit enter here, it's going to retrieve sessfun.php and it's going to notice that there is no session cookie and it's going to pick and create a session cookie. Okay? If I look here, and I'll look at the response, you'll see a Set-Cookie. The name of this cookie is PHPSESSID. That's actually something you can change in PHP, and this is the large random number that was picked by it. Okay? The session is an array and, I set this array to be pizza equals zero so $_SESSION was created on this first request, but it's empty because it's the first request and so pizza is not set and so it says, session is empty. Look at that. Session is empty and then it sticks a zero in there and then down, at the bottom, it prints it out so we see the zero is in the session. This zero in pizza has not come back to our browser. If we look at the cookies, we have the Session ID, which unlocks the session on the next request, but it doesn't store pizza equals zero. Pizza equals zero is stored magically in the server somewhere. Let's move this down a bit. I moved it down a bit. If I hit refresh, it'll come back in session_start, we'll see a cookie this time. It'll send the cookie this time and it will say, oh, pizza is set because of this code, not because of the cookie. It will be the same session because it will send this back again. Session sum pizza equals zero. Then, it's going to say, oh, but it is set so this is going to be false. It's going to drop down here and it's going to look at the current value of sessions of pizza, which is zero right now. If it's less than three, we're going to retrieve it, add one to it and stick it back in the session and say added one and then print this stuff out. Pizza is now one. It's still on the server. The cookies haven't changed. If I click it again, that code around it will be two, all right, and then it will be three. Next time I click it, session_pizza, a server variable, the server variable that's going from request to request to request to request, the server variable is going to be greater than three. Oh, wait. Maybe I got to click it one more time. Click it one more time. Oh, no. It is three because it's only less than three so because it was equal to three, it ran this, which is session_destroy, which really takes out all the key value pairs from the session in the server and session_start. You'll notice that our Session ID didn't change, but then, when we print the $_SESSION variable, it is now empty. What session_destroy really does is empties out the $_SESSION. The next time this runs, the Session ID is still not going to change because that's still in the cookie, that's still being sent every request and it unlocks the session. The session just on the server happens to be empty but the next time we click it, it's going to say, oh, session is empty, and set it to zero. Session is empty. Set it to zero. If you really really really want to, you can go to your localhost:8888, and go into your MAMP, and go to phpInfo, and then look for the session.save_path. That is the session.save_path. Now, I'm going to go show you what happens inside the session.save_path. I'll open up a terminal and I'll change directory into the- oh, come back, session.save_path. So in this session, there are actual files. Let me just do this. Open dot, so you can see it here but I can actually open these. I'll use old school VI to take a look at one of these things and these have what's called a serialized versions of the basic stuff that is in a session. Okay? That is data. You're not supposed to mess with that, but just so that you can see that these sessions variables are stored on disk somewhere, and I don't particularly know why this one's not storing very well on disk but that's a different problem. So there is sessions and I hope you found this particular video on cookies and sessions useful.

Downloads

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

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