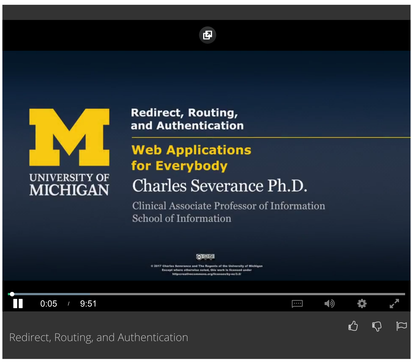
Lecture Content

Assignments

Bonus Materials



Redirect, Routing, and Authentication

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

Interactive Transcript

English

0:08
So up to now I've been talking about the model view and the controller and I'm saying we'll soon tell you what the routing part of the controller. Well this is when we are going to do that, we are going to talk about how your application can sort of push the browser around and make it do things that you want. In the last lecture we saw cookies as one of the things where you and the server can, sort of, dictate to the browser. You can set these cookies, and then they have to send it back. And so what we're going to talk about today centers a lot around what we call the concept of redirect. And so, in the simplest form, when you click on a web page

0:46
it comes into your PHP server, you can send a very special kind of a response called a redirect or a 403, you send a response back. And what happens is, is this response never gets to the DOM. It reads it and goes, oh hm, I'm in the wrong place. I'm supposed to do another GET request. And so it immediately calls and does another GET request, maybe to the same script or different script. And then it gets the thing, and then it makes the document and you see it. So there ends up being two request response cycles. So the first one says go get another different file and then the second one. And so that's what we're going to call, and then we call this pattern the redirect pattern. Because it's, you do a GET and you redirect it and you do another GET or you do a POST and you redirect it and you do another GET. And so we will see how all this works. And this is the way that we are in sort of our controller.

1:42
We run some code and we say, you know what, I want you to go somewhere else. And then you would send a command basically, to the browser, to go somewhere else, or to do something else.

1:51
It gives you a lot of power and turns out it's necessary in certain situations.

1:56
So one thing that we have been sort of glossing over and we got all these screen shots and all the little demos, this concept of an HTTP status code.

2:05
So most everything that you've seen is a 200 OK. 200 says, give me this page and then here's your page, 200 OK, that means I found it. Another one that everyone's kind of familiar with who's used the web, is this kind of concept that 404 Not Found. This is even a meme, or a joke, right. 404, what does that mean? Well, it's not found, it happens to be a numeric [LAUGH] status code for the hypertext transport protocol, HTTP. That is the code that the server sends back if you go to a page that wasn't found. Well those are the two that we kind of note the most about. The one we're going to talk about in this lecture is 302, there's also a 301, which means I know exactly what you're looking for, and it's somewhere else. And so I have one and you can see it at drchuck.com. So my normal post is dr-chuck.com, because I couldn't buy this. It took me awhile [LAUGH] to buy this one because someone got it before me. And so I had to do the one with a dash. But then, eight, ten years later, I finally [LAUGH] got to buy this one. But everyone knew the one with the dash. And so I said, you know what, go here, and if you go in your browser, you will see, you're sent to there. And the idea of a redirect originally was that we knew the web would be dynamic and would change. And some pages would be here and then they would go here and we have to throw this one away. And you put a little kind of breadcrumb that says, nah, this is all gone here, go over here. And so it's called a redirect, meaning that if you go to this, you must do two request response cycles. You request that drchuck.com with no dash. You will get back a 302 to dr-chuck.com and then you'll see my regular web page. And so that's a redirect and that's what were going to talk about a bunch today.

3:49
How is this done? Well, there is this header called the location header. It is in addition to that status code of 302. And the location header is a response header that your application sends. That says, this is not the page your looking for but go to this loop URL. So it's a way for you in effect in the code in the server to send back a URL to go up in [LAUGH] the bar and say retrieve that one instead of the one you just retrieved. Okay, so it's called the location header and we'll see these in a second.

4:19
Now, we set these headers and other headers using in PHP using this thing called the header, it's a function called header. And you give it a string and if you look at [LAUGH] the documentation, this is older documentation because it doesn't show PHP 7. But basically it's header and then a string. And then there is a serious of legitimate headers you're allowed to put. Location is one of them, you say Location: and then some URL.

4:46
That URL can have GET parameters. And basically, what this is showing is you have to call header before any actual output is set. So, you want it to be up top. And if you remember, in our code, there's this magic line where the view happens.

5:00
And so, HTML, that HTML tag is right there. This is the model code, the point is you can do things like header and redirect up here. But you can not do it down here because it won't work because once the output is set, then the headers have already been sent. So the PHP does not send the headers it accumulates the headers until you write your first line of output. When you write you first line of output and that can be as simple as, on your first line is a blank line and then you say <?php. That starts output, so you gotta be really really careful. And it's not the output that the user sees, it's the output that the browser sees. So the HTML tag is not something the user sees, but it is something that the browser sees, and still, this is going to cause an error. So these headers have gotta be called, in our parlance, in the model, in the top part of our document above the line where no output is allowed. No echo statements, no nothing. Just inside PHP run code, make decisions, update databases, and then you pass down the line into the view. And you can't redirect after that, so you have to redirect at this top part. But this controller is your codes decision that I'm going to redirect, I want you to go somewhere else.

6:18
And so this is just a simple bit of code and again, it's model view controller and that there is a magic line. Down here is the view, this is the model, and this part is kind of the controller. So model is kind of updating the parts of this that update the database. But, the controller is also the routing and the what are we going to do next and how is this all going to work. And so this is where the controller really starts to sort of, you can see lines that you go like, that's controller. So this part is controller, so what we're going to have is a little tiny form, right. 'where', and we're going to put a number into it, and if it's a 1, we're going to redirect to redir1.php, this file is redir1.php. If it's a 2, we're going to redirect to redir2.php, we can put get parameters on here. And you don't have to be the same server, you can redirect to anywhere that you want to redirect. You'll notice that I come in, there are no lines, I do a header, and then I return. Meaning there's no point in following through and doing this output because I said that retrieve but it's not put in the DOM. All right, it doesn't go into the DOM, it just is retrieved and then it goes and gets the next thing. So you return, so that is the end of this script. We get out, we get out, we get out. So you'll see and you'll type this a lot, time to redirect header return, header return, header return. So pretty much almost every time you say header shortly within a line or two you're going to say, return. So, on a 1 it redirects back to itself, on a 2 it redirects to redir2.php with a parameter and on 3 it redirects to dr-chuck.com. So if you're to run this code and you type in a 1, you'll see it goes right back to itself, redir1.php. If you type in a 2 to redir1.php it redirects to this one, and the next page you'll see is redir2. And you can put get parameters on here, this'll turn out to be useful later and if you type a 3, it posts to this and then it gets told to redirect to dr-chuck.com. And so that's the three things that this code can do. Just demonstrating how the location header and the header function works.

8:28
Now, if you'd look at your browser console and you look at your network, some browsers have made it so they collapse these two lines into one. You can kind of dig into it, I took this with a older version of chrome and you could actually see that you do the hit the redir1.php and it sends you back a 302 and then that then does a GET which sends back a 200. So this is a return of 302, if you don't see this in your browser don't be concerned. They're trying to make your life easy and they some how have combined these together. If you watch it, you can see it's one line and it kind of jiggles and you see a little thing and you can look at a little more detail. But rest assured, even though your browser doesn't always show it as two request, it's two requests. It does a GET request, which page it gives an empty document and a redirect and then it follows the redirect and gets another document and that's the document that you see.

9:22
So the second page, of course, is the page that comes back. And in this case, we typed a 2 and so it goes to redir2.php. So, that's a real request. The first one, redir1 is an empty response, all right, the empty response, and the second one is a normal response. And that's a 200 and it has a text body that goes along with it.

9:43
So up next we'll talk about how we use these when we're dealing with POST data in order to deal with POST data properly.

Downloads

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

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