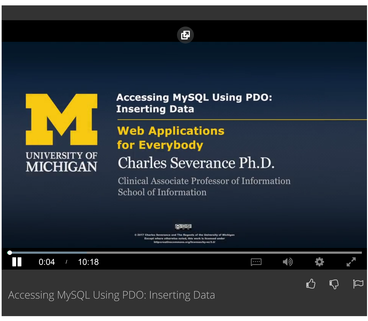
Lecture Content

Assignments

Bonus Materials



Accessing MySQL Using PDO: Inserting Data

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

Interactive Transcript

English

0:08
So now we're going to figure out how to actually insert data from our form on a web page into a database, we're going to go all the way through. So here's the code, it's not all that complex. Let's take a look at what it's doing. And so, here we have a form. Right? And this form, it's down here if we take a look at the form. Again, we got a model view controller where we kind of have the code that's doing all the thinking up here. This is the model, this is the view and the controller does kind of its thing. Right? Controllers the thing that decides it. We'll learn more about controller later but this is the model code, This is the view code. So will start looking at the view code because on the get request, when you do a get request, it just skips all this because there's no postdata. Right? Now I think you should know that. So it skips through and so it prints this out as a form. Remember the name is the key value pairs that are going to go into the post array, name, email, password and then a button. So we got this. So we type all this stuff in type, type, type. We press the add new button and that sends a method=post. Once we typed that, once we cut this button, then it runs back in except now there's postdata. There's the name, the email and the passwords. So these variables get pulled in. And now we're going to do is we're going to write some SQL, we're going to say insert into users, that's the database name, email and password that's the column names. Values= and then we have these little things called placeholders. Now you can name these anything you want, but I named them the same as the postdata but which happens to be the same as the column. This needs to be named right but these placeholder could be x and this placeholder could be x. This name here comes from the post form and this name here comes from the data. But these placeholders are arbitrary things that you make up but you keep your life a little saner if you just name them the same thing. So :name is a placeholder, :email is a placeholder, :password is a placeholder which will be replaced by the actual strings. So, we can print out this SQL. So that's the SQL and you see that the SQL just has these placeholders sticking in it. And then, we're going to do what's called a prepared statement. So we're going to prepare it which prepares kind of looking at it, seeing if you've got the syntax right, etc, etc. You know what's going on. So prepare can blow up. But then we're going to actually execute the statement. So it sort of parses it and make sense of it. You might prepare it once and then in a loop use it over and over again. But for now, we're just going to prepare it and then execute. So this is actually sending it to the database server. And what we're basically giving it is an array execute the parameter so execute we pass in an array. This is the array, pass in with the string with the placeholder :name mapped to the actual string we want. So this here's going to be Fred, this here is going to be Fred@. And this is going to be one, two, three or whatever it is. So what that says is take these values, stick them in the corresponding place in the query and then run the query. Okay? So that's what that's doing, to prepare execute. So it's what's called a prepared statement. And if you're going to use the placeholders, you'll notice we didn't do this before we just ran the query and that's because we didn't have placeholders, we didn't have values that we wanted to substitute. So they prepare execute is the pattern we use when we have values that we want to substitute. And so, we can do that and we can insert a bunch of users over and over and over again. Okay. So we can combine kind of the last two things by having sort of the add new user form that makes this user. And we can have the model up here. This is the code from the previous one where we just checking to see if we're in the middle of a post that comes from the add new and puts those things in and then we can also just put a little while loop so we can see this, so we don't have to kind of hit the insert button and then go look in phpMyAdmin to see if it showed up. We can actually just have an SQL statement that is going to select and this of course came from like a couple of sample bits of code ago where we do selecting print all this stuff out. So you type something, hit submit and then, poof! It shows up at the bottom and that's quite easy and quite natural. I'll show you in another video the exact demo of how this works. So now let's talk about deleting users, so that's how you do inserts and selects member doing CRUD create, read, update and delete. So we have to do delete. So the first thing about delete is you do not want to do a delete on a get, you always want to do a delete on a post. And so this is where you see and from a user experience perspective, you see how often if you hit a delete button like a little trash can or something or a little trash can you hit the thing. You hit it then it goes to a page and says, are you sure? So often what they're doing is this is a get request that's going to show you a thing and then this is going to make when you hit this button that's going to be a post request, because if you remember getting post in the rules of getting post, you're not supposed to change anything on a get. You're only supposed to change it on a post because crawlers don't follow posts etc, etc, or the browsers don't let you double post etc, etc. Okay? So, we're going to put up a screen that's going to give us a key that we want deleted in a field, then we're going to hit the button and it's a post, so then, when we hit this button it comes in and runs this stuff. And so, if it's a get request it just comes through here and displays the screen. If it's a post request it comes in here and runs the stuff. So it's really simple. Delete from users where user ID= and then you just have a placeholder. In this case I'm using :zip because you can make the placeholder be anything you want. Okay. So then I'll print that out, I'll prepare the SQL and then I say I would like to execute that SQL and I would like the :zip to be replaced by whatever that user ID is, in this case the 4 is going to go in there. So this runs SQL, delete from users where user ID equals 4 and that actually deletes it and away it goes. Okay? So you start to see the pattern, it's like, you make a SQL, you put in your placeholders, you do prepare, you do execute and execute you tell what you want to put in each of the placeholders. Right? So it deletes it, and it's gone. And so I can actually in this user3, I sort of combine all this stuff together and I put a little delete button. And so eventually you'll have prettier user experience but I want to make it so that if I press this delete button this guy gets wiped out, if I press the add button this guy gets added, there's a new one, so I can have add and delete now combined on to the same screen. So let me show you how that ultimately happens. Everything is the same in this user3.php. I have the controller code, the view code up here. Right? The view codes up here and that's the delete code from that other one. And then what I do, everything else is the same except this little bit right here. What I have here is I have, this as a form and when I put this submit button, that data gets sent. This is a form, this is a form, this is a form and that's a form. So let me show you how I construct that form. Okay? So, if you look this, the table cells, the name is one cell, the email is another cell, the password is another cell and then this is that last cell, this is kind of like the action column. And in that cell, I print out a little form, form method=post. Input type=hidden. So an input type=hidden is like a text but the user never sees it. And I say name=user id value=. And then the primary key of the row. So I've added that I want to also fetch the primary key of each row and I put that in quotes and I got it with arrows and less thans and new lines and all that stuff. Now I have a little delete button which you do see and has a little value of Del on it and then I have the end of the form. And so that's how I end up with a form over and over and over again. Right? So if you look at the source code of this particular form, you know, it's going to be like, yo that's there and then this action column is, there's a form input type=hidden, name=user id, value=five that came from the row user ID right there. And then I have a submit button and that shows the submit button and then I have it in the form. So like I said, each one of these things is a form, the difference is they have a different primary key, whatever the one, two, three, five, whatever they are. That primary key is embedded in a hidden field in each of those forms. Then, when you press this button, it says, oh, the name is delete, so if I have isset($_posts['delete']) then I know that it was the delete form that was pressed. Right? And I have isset($_post['user ID']) so that triggers this model code to run the delete. And that's just the code from there. So that's how I end up with one, two, three, four, five little forms on this page. Now, later we'll make the user experience prettier as we go forward. And so, this again is kind of the model view controller, the whole model up here, we're going to require the pdo, we've got the add model, model. We got the delete model. And so this is our magic line between the model and then below that, the view. And so then inside the view, we're going to do a database call and we're going to loop through all this stuff and away we go. That's really from the other thing. And then we have the bit that has to do with user add form and in a way we go on. I'll walk through all of this in a text editor in another video. So up next, I want to talk a little bit about some security holes that I may or may not have just introduced into that last bit of code.

Downloads

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

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