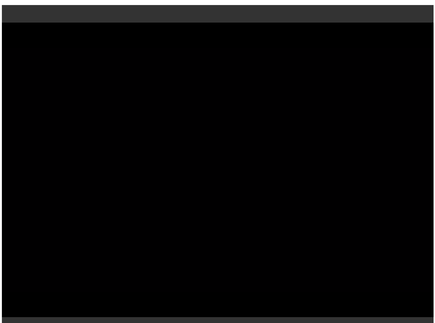
Lecture Content

Assignments

Bonus Materials



Error Handling with PDO

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

Interactive Transcript

English

0:08
Okay. So, one of the weird things about PDO is how error handling happens. Meaning that there's all kinds of things. Everything I've done so far other than like when you made your PDO.PHP work. Everything I've done so far has pretty much been syntactically correct. So I want to talk a little bit about the kind of things that can go wrong. And so this is kind of going back to where I was saying that you can have different error modes. And so this is error mode warning, it can give you a warning, it can blow up, I prefer the one during development that it blows up. And so here is a perfectly legit thing. It's going to take a user ID from the GET, you know, right here to GET parameter. It's going to take the user ID. It's got a substitution variable xyz, it runs. So, this is all good. Right? It doesn't blow up. But now, let's make a mistake. So this is a kind of a mistake here. Okay. This is a mistake where I've got this in my SQL under my execute, I tell it that. So, this is going to blow up. But because I've got error mode warning, it doesn't actually stop. This is failing. This statement right here, right? It says PDO execute invalid parameter number do not defined. Invalid parameter on error line nine, blah blah blah. These aren't the best of error messages. But this line right here is blowing up. But the problem is is just that the fetch fails, this is a kind of a non-fatal error, this should blow up but it keeps on going. So it comes down here still says user_id not found. So if something so badly wrong that this blew up, putting out a warning is really a bad strategy. Okay. And so this is why I prefer error mode exception and you'll see in my PDO.PHP Code that I'm always doing error mode exception. Error mode warning is one way, error mode silent is the default mode. That is really bad. That's why I want you to set it every time and error mode warning, is almost useless. You can check with an if statement at the end, but really if you've got a syntax error in your SQL or something that bad, you might as well just blow up because you can't really recover from it, you just go back and fix it. So that's why I suggest you always use error mode exception. And so if I turn error mode exception on which is my recommendation for my database, so it's still gets the same thing, invalid parameter not defined. It blows up and it dies. Which means it quits right there. This is a die. It stops and it doesn't run the rest of this code. And so that's what I like because really this is a syntax error, the SQL. PHP can't know because it's valid PHP but it's not valid SQL. And so that's why I want you to just set this in your PDO.PHP all the time. Okay, just set it and forget it. Now, you can do a Try and Except and you can Catch these things and sometimes it's really important for you to do that because sometimes these things can print out data that you don't want to show to the end-user. And so you can put a Try Except around these things. So now I've got to Try Except, Try Catch around this thing. So I told it I want it to blow up, but I've also told it that I want to catch the exception. So this runs, this is okay, this blows up because this is where it detects that there is invalid parameter and instead of coming down here, it comes to our catch spot and I get an exception variable because it passes the exception that caused the explosion inside here, blew up but it sent us back a little bit of a breadcrumb as to what went wrong and we can say, get the message, and that gives us this message right here. That message comes out and I print that out and then I return. So I don't, this is a way to get out of this script. And if it worked, which is not going to because it's totally broken, it continues and runs. So sometimes you want to do a Try and a Catch of those exceptions and then print your own thing out. Now the interesting thing is is when you're in production, it really doesn't do much good for your user to see an error message like this because you are the developer in here. This is you. And then your end-user is seeing this, right? So the user is using your application someday. We're not just writing applications for ourselves, when we're developing, we're seeing the application and error messages are really helpful. But when we're showing it to users, you don't necessarily want to show these because like this can have sensitive information. So in production you sort of want a different pattern. So what you do in production is you turn on exceptions. You do a Try of course this is constructed to blow up. So it runs the Catch, okay? And the exception comes in and the thing that we show our user, we send this to our user. So we say internal error please contact support. But then what we do is we put an error log message and we've talked about error logs before, an error log message is sticking a message in the error and the log file we can go look at later. And what it does it's like I'm saying, oh am coming from this file, error4.php, and then I take this message and I print it out and I stick it in the log. So this does not go to the user, this goes to the log. And then I quit. So now I, you know, you'll see this all the time when you run applications it's like, unknown internal error. Hopefully it's logging a bunch of stuff. So the developers can say wow, I can see what's going on. So this is way better to log the errors than it is to actually display them to users. Unless you're the developer and developer you can be watching log anyways. And then, remember where the logs are at. Well, the logs are at, you can do the PHPInfo and you can look for the string, error_log underscore log and then you can find the absolute path. I have been running the app and this is a Macintosh and it says, that is where the logs are stored. /Applications/MAMP/logs/php_error.log. So then I can go take a look at that file and that's where I will see those errors. So under different scenarios, you have different places for them. There's this really cool thing that you can do if you have a window, and this you'll see me when I'm coding, I have a tail running of the log. And there's this thing called tail, that's on Linux and Mac, tail -f takes and looks at that error log file and it just sits there and if a new line gets shown, it adds to the end. And so you can dynamically watch it. You don't have to go keep opening it in a text editor, you can use tail. On Windows you might find it interesting. I don't know if this is for you or not but you can find a Windows tail and that's super useful, so that you don't have to keep every time something goes wrong, I just like to keep the tail of the error log going in a window somewhere off to the screen or if I got two screens, I have the tail on one side of the screen. And so this is what it looks like sort of in my screen, the way I do it is, you know, I've got, I mean the logs file and tail -f php_error.log and then it just sits here. And then when something goes wrong, right? And when something goes wrong, error2 is blowing up, where is error4, yeah error2. So this is just showing me the errors and- but this is showing the stack trace. Showing me all kinds of great stuff. Oh and here's error4, sorry. Here's error4.php. Right there error4.php. And so it blows up, tells me I'm in error4.php and then it gives me that error message from it but never shows it to the user. That's the cool thing about logs, is that you never show this to the user but you can come in later or while you're developing just watch it. So developing and watching the log at the same time is not a bad strategy at all. Okay, so we've sort of zoomed through a whole bunch of things where we have got to the point where we make stuff in a database, we figured out how to use PDO to make a connection and send data into the database, retrieve data from the database. And here at the end, we sort of looked at some of the more subtle details of dealing with database errors in the PHP code that we write.

Downloads

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

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