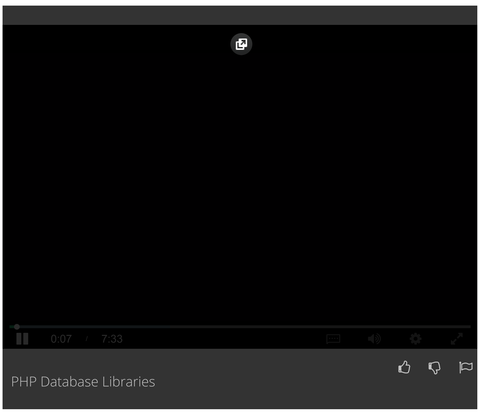
Lecture Content

Assignments

Bonus Materials



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

Interactive Transcript

English

0:08
So now we bring things together. We've talked about SQL, we've talked about PHP, and now we're going to connect them together. And this is where my favorite picture in the whole world really starts to come together, in that we've got our request response cycle that sort of comes back and forth, and all that stuff, and we'll get that figured out. But now, what we're going to do is we're going to actually build a request that goes into PHP, PHP is going to make a database connection, and send SQL commands. SQL is going back and forth crossed here. And we've been doing this, we've been talking SQL through PHP MyAdmin straight to this thing, but now we're going to make it. So PHP creates and sends the SQL. SQL is going to do the same thing: select, read, update, delete, all that stuff, and then send us back what's called a record set. And then we're going to read through that as if it were a file. It's a sequence of records, basically, and we're going to do something to that. We're going to write some HTML out, then we're going to produce the response and then from that point on, it goes into the DOM and then we see the DOM. So this is the part we're going to focus on today, and we're going to use a library called PDO, the Portable Data Objects, which is part of PHP 5. It was introduced in PHP 5. And at this point where, PHP 5 is kind of the older version, PHP 7 is the modern version. And like I said, we've been doing all along, we've had this database server, and we have been acting like the database administrator by using PHP MyAdmin, sending SQL, and then it just shows it back on our screen. And again now, we're going to use PDO to do this where the user talks to our PHP code. We, as the application developer, writes some SQL, send it, and it goes back to the end user. So, our job is to write some PHP to do database. We don't let the user talk to the database, because then they could see things they're not supposed to see. I mean, our job here is to sort of give them the view of this data that we're supposed to give them. Like I mentioned, we're currently in PHP 5, and we're going to PHP 7, but in the PHP 4 and earlier, there was another way that we did this. And one of the things that everybody absolutely loved with PHP, was how easy it was even like 2, 3, and 4 version to access SQL. It was just built in. You had these routines, these are the legacy routines, this is less than 5 in the legacy, there mysql_. As we went from PHP 4 to PHP 5, they decided, we can do all kinds of things. Now, it turned out that in PHP 4, you would have Oracle_, and SQLite_, and basically you'd have quite different routines for each of the different databases that you would talk to. SQLite was built-in, but the other ones weren't, and mysql was built-in, and a couple of things happened. They wanted to build this object-oriented version of this. So what happened was, as everyone had been using these for 15 years, and we really got used to them. So as PHP 5 came out, they went and did two things. First, they moved an object-oriented one. And it turned out you can only have one connection, you couldn't even connect to two databases unless you connected to one and disconnected from that one and the other. There was a lot not to like about this other than the fact that we knew them really well. So they wanted to do an object-oriented version, because then you could open a connection to one database, and open a connection to another database, and have an object for each of those connections. And then when you want to talk to this database, you'd use this object, and you have simultaneous connections. So that was really nice, right? And the idea was, is we didn't know if people were going to want to use the patterns from these MySql routines, and so they made a new thing called MySQLi, squelly I think is how I pronounce that, and it's an object-oriented version that has exactly the same calling sequences to our beloved mysql_ routines. Kind of like you see date_ add in the OO lecture which we did that stuff and you just say it, dot dot dot add. So it was like redo the non object-oriented routines as object-oriented routines. So they did that, and then they also said, let's build from scratch, looking at other languages, and how they had done some really pretty things. And they looked at all the different mysql and oracle_ and they're like, "What are some of the prettiest patterns?" And so, they made a whole new thing that had a new API, that didn't try to look like the old one. We didn't know which of these two we were all going to convert to, when we went to PHP 5. Now, the old ones would work, but they're tacky and not very well liked. I was teaching classes during this point in time and I for a little while, I thought to myself, okay. I've got all the source code, all the sample code, and I'll just use mysqli. But I quickly ruled that out, because there are so many wonderful features that make writing SQL much easier in PDO. And so there was a debate for a while, but I think at this point, we're pretty much PDO and in this class, I just use PDO. And given what I just said, all the stuff that I just said, you can debate for a while, but I think the debate is kind of over, and PDO has won the debate. And so I just want to show you some sample code, because you may run across code. The old classic mysql, you can almost see this because it always starts with mysql_. And it's like, oh okay, this is basically less than PHP 5. It's not object-oriented, and has all kinds of sort of flaws, but it's flawed from a architectural perspective, but it's beloved from the fact that we've been using it for over a decade, and got really good at it. And then there's the mysqli, you see that it's like new, it has this thing, and it only does MySQL and the mysql_ query becomes mysql arrow query. And this is really just a one-to-one translation, except it's OO and it's OO syntax. And it has the advantage of, you could make more than one database connection if you have an app, and you have connection to two databases, and then you can do that. But I think very few people have ever used that. What we end up with is PDO. PDO is an object-oriented pattern, has a new thing, it does this stuff, and it tells what database it's going to use, etc. And then we run queries, and then those are slightly different. And so the difference here is, not only is this PDO object-oriented. O stands for Portable Data Objects. It also has a different API pattern. Like I said, I thought it like this, but now I just love this and I don't like this. And so here we are, PDO. You might see this stuff. So be ready for it. And whatever it is, ultimately, the goal of all the PHP is to create this string of SQL, and send it to the database, and then get some records back, and then loop through those records. So up next, we're going to actually talk about how we insert data through PHP.

Downloads

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

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