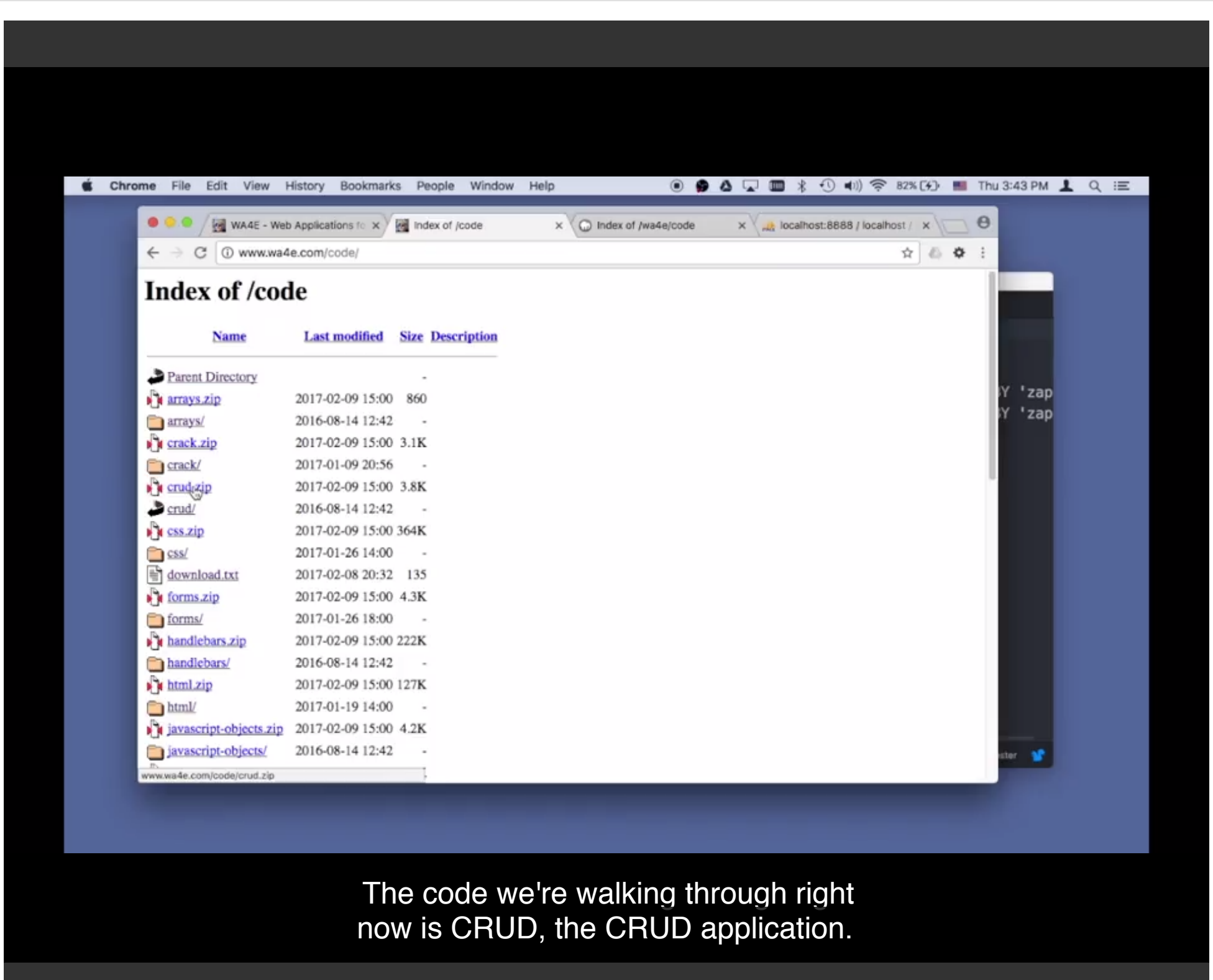
Lecture Content

- Create, Read, Update and Delete (CRUD) 20 min
- Code Walkthrough - CRUD in PHP 14 min

Assignments

Bonus Materials

Graduation Ceremony



Code Walkthrough - CRUD in PHP

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

Interactive Transcript

English

0:00
Hello, everybody. Welcome to Web Applications for Everything, yet another code walk-through. The code we're walking through right now is CRUD, the CRUD application. Now you'll notice if you click on the CRUD application, it won't let you run it on Web Applications for Everybody, and that's because it requires a database connection. So you're going to have to download it and run it somehow on your local host to sort of see what's going on. And so there we are, we've got CRUD running here, make a little bigger so you can see it. Because it has to be set up and I have this misc folder, we got a little smaller. This misc database, and if you're a little concerned or confused about how to get things set up in the CRUD folder, there is A SQL notes on how to set this database up, and then insert some users into it. And so, basically the idea of this CRUD application is it's not really that different than anything else we've done. We're really kind of moving code around and cleaning it up, and bringing it to sort of the best practices. So before we had done things like in the PDO file, we did things like user3.php, where we had a DELETE and an INSERT in a table all showing in the same thing. So in a way, this is a file from several lectures ago that's the closest to what we're going to do in CRUD. But what we're going to do is we're going to have a set of all. We're going to pull some of the functionality out and kind of clean it up out of user3. And move it around and then come up with a pattern where we have a file each for the INSERT, the DELETE, and the update in the list. And so this turns out like the index file is the thing that does the list. Let's just play with this application for a second. So here's our index, and it shows what's in the table, we can add a new thing.

1:57
Oops.

2:01
Catylyn, c-a-t-y-l-y-n @umich.edu, and then 987, so we have an add page that does the add and everything has a cancel to go back. And then it redirects itself back to index.php with a little success message. And then we if we got a Delete, We can delete this, and Delete it, now it's gone. And then we have like on Edit, and we can make changes.

2:38
And then update it, and then it changed it. So, we're kind of routing between these four main files. There's sometimes a view file, like a view.php that we click on, but we have so little data that we just kind of do it in the index.php. So this is not that different than anything that we've done before, so let's go ahead and start with index.php. One thing you'll notice about index.php is I'm using hrefs to go into edit with a get parameters user id = 1, and into delete. So there's no forms on this. And so there's no posting going on. There are just hrefs, these are hrefs and this is href. So if you look at the top, the kind of the model part of index, there's nothing there. All I do is pull in the PDO and I start the session. And then I have flash messages, and pretty soon this is going to, it won't take long. I'm going to stick this in a utility file and just include them. But basically, they check to see if there is a kind of residual from whatever previous thing was executing, if there's an error or success, printing it out, and then deleting it to implement the flash pattern. This table is pretty much code we've done before. We make a table, we make a row and then we make one, two, three, and then the fourth column is this edit/delete. And all we do is we put an anchor tag in, edit.php with a query parameter, user id equals the primary key, the user id, and that's the edit text. And then the delete.php and that's the id equals and the delete text. So that's how we end up with hrefs, and then we just go to add.php to go to add new. So let's take a look at the add code. This yellow stuff, this is just my browser. I wish I could convince the browser to stop doing that. That's the browser pre-filling stuff in. And the add code is kind of like the user3.php code.

4:28
It's got some post, intelligent post code. It requires pdo.php session start it gets things going. And then it's not doing a very good job, but here's the beginning of the view. It's not got a doc type or HTML, sorry about that, but it does check to see if there's the add flash. We'll use that in a second. Now we just have a simple form, and then we have a cancel. I'm just making the cancel be an href rather than a fancy button. Now, if we take a look at the post code, this post code does a bit of data validation. Now I'm finally doing some data validation. And this is the model code that says, if there is these three variables that are in the post, a, a@a.com and a is the password. If those are set, first we're going to check to see if the name or password is less than one in terms of length. And if that's not true, then I'm going to say missing data. So let's execute. And then I'm going to redirect, right, set an error. Redirect back to the script, and then, quit so we don't fall through. So we're going to be doing that all the time. So, let's Let's send bad data. And, here we go. So if I do a View >Developer > Console on this and take look at the Nnetwork tab, and let's make another mistake. Get rid of that, get rid of that. Blah, Blah, Blah, but we'll keep the rest of it blank, and we're going to say Add New. You'll see that it does a post to add.php and then it sets the error and then it does a redirect back to add.php, which immediately does a get to add.php, which is what then gives us back this page with missing data, right? So that comes out, so that's a post redirect, and this is a error checking a sent data validation error checking post redirect. And that's exactly what we want. We're going to check to see if there is an @ sign in the middle of the email. And so if we have non-blank stuff here, Right, but there's no @ sign here it's going to complain. It's going to come, it'll make it through these stir lens, but it won't make it. So this is my pattern data validation looking good, check this one, looking good. Anytime you quit, just quit, right? There are other ways to do this, but I try to make my data validation kind of in a protector pattern, where I'm not going to let you go any farther, I'm out of here if something's bad. So this one's going to say, bad data, [COUGH] with a post redirect.

7:00
And if everything is fine, it does an insert using pdo with the substitution parameters, name, email, and password.

7:10
We prepare the statement. One of the things is, you don't put debug print in here because we're still above the line and so we don't get to print the SQL out. I tend to, it's a little harder. You can print this stuff out but then you don't want to sort of comment out the header, because you want to be able to see the output if you're debugging in here. But we just do the insert, wWe've done this before, and then we put a success message. And then we go to index.php. So we're doing controller function to go there. So xx@y.com and some password and we hit New and so it redirects and goes back to index.php. And index.php has a success message. And because it's been wiped out at the end, we hit refresh or get and that message goes away. But the data doesn't go away. The data is still there. So that's the flow for insert. So let's do delete next. So delete, remember, it just is a href coming out right here. It does an href user_Id equals and then the primary key. Now let's take a look at the delete code.

8:18
So here's our model, so we're going to ignore that. Now we are going to check to make sure that the user id we've been given is correct. So if I somehow can find my way to delete.php, it says missing user id.

8:33
Sets an error and goes right back in index.php. So somehow that didn't work well. But if it's a URL that we constructed the user id probably is in there. So then it's going to demand that, then it's going to do a select where the user id is whatever this number is, 8 in this case. And if we don't get a row, then we're going to say bad value for user id. So if I do user id =, come on, 800, it sends a bad value for user id, and then redirect index.php. See how easy this is? You just chuck a problem, boom, set a flash variable, send it back.

9:14
Okay, so now [LAUGH] finally, we're going to get to the point where we're going to put out confirmation dialog and we're use htmlentities. We're using the short print this variable out. What we call htmlentities just in case the user has named themself Mr"input type equals button or whatever or little Johnny Tables for all we know. And so we're just putting that out and that comes up here. And so x is the person's name.

9:41
We have a little form with a hidden variable which is the user id, the primary key of this user. And then we have a little Delete button or Cancel button that's really kind of routing us back to index.php. But then the post comes in. And we post the delete, we're checking that's the delete that comes from the Submit button, all right, post, sub, delete the name. So if set, post, sub-delete, that really means that delete button was pressed and there's a user id. You want to double check these, don't skimp on these. Check for everything you're going to use, because the last thing you want to some kind of trace back in here. So, just double check stuff. This kind of guardian pattern is what it's called, to make sure that we don't run code when we're not sure the post is set. So we're not just decided when to run this code but we're also making sure that all the preconditions necessary to run it are okay. So we're going to run a delete statement, delete from user id, whatever. We're using prepared statement and all our PDO goodness. We say Record deleted and we redirect to index.php and we return. These three lines and these three lines are like, yep, you better get it, because it's what you do. I want to go here, and I will please display a green message record deleted. And so when I say delete, it does that, right, and goes back to index.php, and has a nice little success message. And I keep saying this, if I refresh, the success message goes away but the data doesn't change. Okay, so that's a flash.

11:19
So we're zooming through this. So the only thing that's a little new, actually, most of that was code we'd already had it's just a lot cleaner now, right? So now let's go into Edit. Let's take a look at edit, edit is a little bit different, it's kind of like an add. [COUGH] Here we go again in the model part. We're going to say if these things are set in the post, I'll come back to that.

11:41
Again, we're going to be clever, and we're going to check to make sure that we know the data. But now we're going to select all of this stuff, right? So select and we wait, if there's no row. So if someone had somehow gone to a user id number that doesn't exist up here, but we won't do that. It works just the same as the delete code. Just grab this code right here from the delete. Looks like it's exactly the same as the delete code.

12:09
And then I'm going to start printing the data out. And so I'm I'm going to do HTML into these rows. So name, email, and password and put them in the variable and n, e, and p and pull out the user id, that's just to make this a little more concise. Here's some HTML with some little bits of PHP variables being dropped in for the existing old values. Now I've already called HTML entities on these three, so they're okay, we're safe. And so that's how these old values that were pulled from a database in row number one get there. Now I'm sitting here waiting, and I'm going to say, Chuck Right, Chuck, and I'm going to do a post. And then the post is going to come in here. [SOUND] So this is the first time we've seen an update, but it works just the same as anything else. It's exactly as you would type UPDATE users SET, column = placeholder, column = placeholder, column = placeholder, WHERE user_id = placeholder. Then we prepare that statement, and then we execute with the data from the post. And of course, this is a solving our problem with SQL injection because we're using prepared statement in PDO. And, we should in time will put error messages here, etc. But for now, this is mostly going to work, unless it blows up because we have a syntax error in the SQL, which we don't. We sent a success message, record updated, and then we redirect, aAnd then we stop. Pretty much as soon as you say header location, you're going to quit. So I've changed this to Chuck, I say Update, and there is my data. And of course, it's supposed to redirect. I don't have to tell you that, by now, I think you kind of got it. Edit, we can do a post to edit, get no response, but then it redirects, and then it gets this thing back. So, that's pretty much it, right? That's pretty much the CRUD code. It's not that compared to everything else we've done so far, it's not all that new. It's just coming up with a nice organization for the code so that we can make sense of it, and not make these applications too complex. So, I hope this was helpful. And I hope the rest of the code walk-throughs have been helpful as well. [MUSIC]

Downloads

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

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