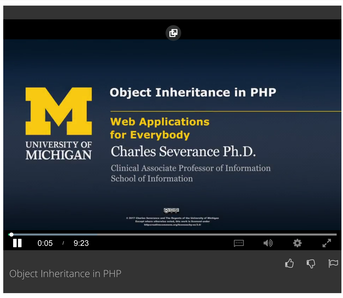
Course Materials

Lecture Content

Bonus Materials



Object Inheritance in PHP

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

Interactive Transcript

English

0:09
So now we're going to talk about inheritance. Inheritance is another basic object oriented concept. We're going to talk about it in the scope of PHP. But if you're in any other language, we use inheritance in PHP. It's a powerful notion that really makes it so that we don't have to repeat ourselves. And we have these class hierarchies. And again, I'm not asking you to write all this stuff I just want to listen and so I can use the word inheritance later, okay? So it's another form of store and reuse. It's going to like a function. We do it once, we're going to do it over and over. But if we have a class, then we're going to make many children classes. So a way to reuse class capability. Another word for this is a concept called subclasses. There's a class and then another class that subclasses it, which adds value. You can think of this as a parent class and a child class. All of these ideas are the same. One is sort of the original, and the other is sort of the extended copy.

0:59
And so we use the word extends. And so we'll take this very simple example where we have this class Hello that we used from the previous example. It has a value inside of it called $lang. We take a constructor that copies that value in, it's the stuff that we did all before. And we have a function in there. So we have a class called Hello.

1:18
Now, if we want to make a new class called Social that extends Hello. So this extends Hello basically says, take all this stuff and put it in. That pulls it all in. And so Social doesn't start as an empty class. Hello started here kind of with nothing, then we added stuff to it. But Social doesn't. Social has all this stuff pulled in.

1:39
It's as if we've typed it all there. And it's a form of not repeating yourself., with these things are all in there. So there's a $lang, and there's an auto construct, all this stuff. And now we can add a thing, we're going to add the bye function.

1:51
And it's just another bit of code that looks at the current language and away we go. So that we've made a new class, so we have two templates, we have a Hello template and a Social template. So now we say, let's go ahead and make a Social and pass in 'es'. So that calls this constructor. It's a Social, but this is part of Social as well. So they're merged together in Social.

2:14
Runs the constructor. We can then call the greet method which comes out of here. Runs that code, which was the code we used before. But then, because it's a Social, it also in addition has the bye method, which is now in here. So it's a way to take and pull all this capability in to a class, so you don't have to repeat yourself. So these are often called base classes, and these are called extensions or class, subclass.

2:43
But you can see it's a powerful notion. You still don't know why you want to make classes, but trust me when you get around to it, we will build these things and you'll find them to be quite useful. Mostly in the name of not repeating yourself and doing the same thing twice. So inheritance is an important aspect of object orientation.

3:02
So another thing I've sort of been glossing over is the notion of the visibility of an attribute. So attributes are the data, so they're variables that we put in there. And then methods are the code that we put in there. There is a concept of visibility. And part of the idea of a class is to hide complexity from the outside world. And and so if you think about a class, it's got data. And it's got methods. And up to now, we can play with the data inside the class. And we've been able to play with the data outside the class. And that's because we have marked all these things as public. Public means it's equivalent to access them inside and outside the class.

3:42
Private is the strictest, which basically says it can only be accessed inside the class and it stops this access. And there's kind of an in between state. If you make a derived class, which is a subclass. This is a parent, this is a child.

3:58
So this a child that inherits all this stuff.

4:02
This class can always talk to its own member variables, but this class cannot. So if it's protected, it can be accessed only inside the class, not from a subclass. If it's protected, you can access it in the class and the subclass, private you can't access it from the subclass. And public accessed outside the class, inside the class, and in a derived class. Protected inside the class and in derived classes but not outside. And the private is only inside the class, and no derived classes. And the reason is that sometimes you want to have a variable that's really just for this use only. And you don't want anybody messing with it because maybe two variables have to be set a certain way. And so you put a really bright line around them and say these are private. And if you really don't care if someone's messing with it, you just say I'll open the door up and I'll let you in to come in and change those things. So I'll mark it as public. So here's just a really simple example of showing how this works. There are three keywords, public, protected, and private. You'll notice I sort of slid that in the very beginning of the public, and that I was playing with it. So, in this class, you can see that this is outside. And this is inside.

5:09
So inside, we can talk to the public, to the protected, and the private. That all works. You can always talk to your member variables inside. But outside, this is way outside because it's not a derived class. It's way outside. The only thing we can talk to is the public. We can't talk to the private or the protected. So if you say private or protected these are not accessible so we've got a bright line and the only thing that's allowed outside this line is that and the function comes out as well. But you can make private functions if you want, as well. So that you get control when you're building it. because you're the class maker at this point and you're saying these things are internal. I'm going to use them for my own purpose inside of this little space. These things are external meaning that I'm going to let someone else mess with them.

5:51
And then to understand how protected works, we just take a new class that extends it. So MyClass2 extends MyClass from the previous slide. And all we can do is the public, we can't see anything there. We can call the printHello function from outside because that's just a member function that's public by default.

6:09
It can't talk to private from that other class, right? That was the private variable from the previous class, it can talk to the protected variable from the previous class and the public variable from the previous class but just not the private, okay? And so that kind of summarizes it, public outside the class, inside the class and in the derived classes. Protected inside and in derived and private is only in the class. Okay, so one last thing that I want to show you is sometimes programmers will want to not actually use the class keyword to make a new object. And so you can sort of make a class and dynamically put stuff into it at runtime. And so you will see this code sometimes, sometimes I use this. I'm not sure it's the greatest thing to do, but there is a way to say, make a class that has nothing in it. Has no code, no nothing. And so, that's an empty class. So it's a template for a class that's empty, but it is a class.

7:09
And so you can, in code now, say, put a variable name inside here. And then put a variable score inside there. And it does it, it sort of [SOUND] opens it up and puts these variables in and assigns some data into them. They're kind of by definition, at this point, public because we're outside the class and we're putting data in. But you can do that and so don't, some languages you have to predefine the class and have everything defined in a class-like thing. But in PHP, you just kind of extend it, and you can do all kinds of things. I basically created an access to public variables there. And so this basically is a standard class with key and value. It's not an array, but it's still kind of showing you the attribute name, name, and score. And in a print_r statement. But sort of a more elegant way to do this would be to make a class which is then a template. That says there are two things in players in general, a name and a score, and you can set it to some value. And you can say new player which creates now, right off the bat, a class, an object with two things in it, with names. And then you can, add one to the score. And if you print it out, you see that this is the object that we defined, so it knows that that's a different thing than a standard class. This was a standard class when we first created it, but it's a pattern. If you see this code, I don't want you to freak out when you see that code. because that's how some of us old school people made objects in the first place. We kind of treated them

8:38
kind of like arrays that were kind of prettier in a way. There was another way to do arrays without putting quotes in the arrays. So, the whole purpose of this lecture and these series of object oriented lectures is not necessarily to make you a solid object oriented programmer, because that takes a lot of sophistication. Hopefully, maybe you've taken my Python course and you learned a bit about object orientation there and we get to see it again. So for me, the idea of object orientated programming is something that comes to you more slowly than programming in general. So I hope that when the time comes that you're facing more complex problems. And now you really have to use OO in a more significant way. This kind of prepares you to understand that better. And so, that's the whole goal of this and we'll see you in the next lecture.

Downloads

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

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