**107. Injecting the Logging Service into Components**

* -: In the last lecture, we created our service and I told you to not instantiate it on your own.
* I told you that Angular offers some great tool which will give us access to our services.
* It's ***Angular's dependency injector***.

***Angular's dependency injector***:

* What is a dependency injector? Well, a dependency is something a class of ours will depend on.
* For example, the new account component depends on that logging service because we want to call a method in that service.
* And the dependence injector simply injects this dependency, injects an instance off this class into our component automatically.
* All we need to do is, we need to inform Angular that we require such a instance.
* ***So how do we inform Angular that we require such a instance?***

1. ***We*** add a constructor to the class, to the component in this case, where we want to use our service.

* So there I can bind it to a property by using this TypeScript shortcut of adding an accessor in front of the name of the argument to instantly create a property with the same name and bind the value to it.
* So here I will name this logging service.
* This name is totally up to you.
* Now this is the important part here though.
* b) *Add a type assignment here*.
* This is not optional.
* You need to set a type, and the type has to be the class you want to get injected.
* Logging service in this case.
* Make sure to also add the import at the top.
* Now let me close the body of this constructor.

Text

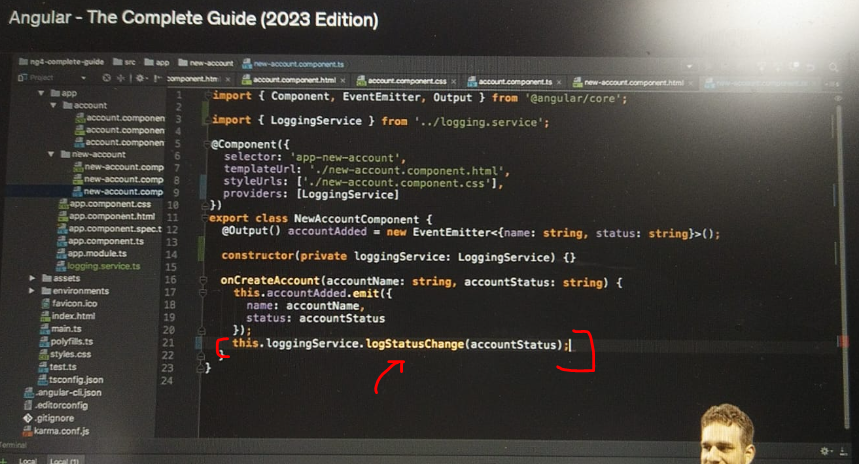
Description automatically generated

* And this simple task here informs Angular that we will need an instance of this logging service.
* Correctly you might ask, well, how do we inform Angular? Or why does this matter if we write this in the constructor? Think about who gives us the instance of this account component, this new account component here.
* This is a types of class in the end.
* So somewhere this needs to get instantiated so that something happens in our app.
* Well, who's responsible for creating our components? Angular is, of course, because we are placing selectors in our templates.
* And when Angular comes across these selectors it gives us instances of our components.
* Now, since Angular is responsible for instantiating our components, Angular will need to construct them correctly.
* So if we define in the constructor that we require some argument, Angular will recognize this.
* And now it tries to give us that argument.
* It tries to give us this, well type in this case.
* So it knows we want an instance of the logging service class because we define the type here.
* This is why this is important.
* Now that is almost enough, but not quite.
* Now Angular knows what we want, but it doesn't know how to give us such an instance.
* C) We need to do one additional step. *We need to provide a service.*
* Provide simply means we tell Angular how to create it.
* And that sounds very complicated and it is very simple.
* *All we have to do is add one extra property to the at component decorator.*
* *The* ***provider's*** *property here.*
* *This also takes an array like other properties we added before.*
* And here we again just have to specify the type of what we want to be able to get provided, you could say.

Text

Description automatically generated

* So logging service again.
* Now with that, Angular, when analyzing the component, recognizes that it should be able to give us such a logging service and it will set itself up to be able to do so.
* And when it then actually builds the component, constructs it, it sees that we want to have such an instance and it will know how to give us such an instance.
* And now we can simply, in our component, anywhere in this component, access our logging service property, which is created automatically since I used this type script shortcut here and call log Status change.



* So now I'm not creating that instance manually, Angular does it for us.
* And why is this better than creating it manually? You will see some other advantages later, but this basically lets us stay in the Angular ecosystem.
* And Angular knows how your app works.
* Again, some other advantages will come up later.
* So with this, we now have the same code as before.
* So if we have a look at our application again and create a new account, we still see the log here depending on which status we chose here.
* So this still works, but now we're injecting the service.
* Now, let's all do the same by copying that in the account component here.
* So here I will also add the constructor.
* And of course here we also need to add the import because TypeScript needs to know where this logging service comes from.
* Not Angular, TypeScript.
* And I will also add my provider of course, because for this component too, I need to inform Angular how to create this logging service.
* So if we do this and then go back to the application here, let it reload.
* Create a new service again.
* This works.
* Now let's change the status.
* And this also still works.
* Now of course, because I didn't replace this here.
* So let's call the logging service now and call log status change and pass the status.
* Now if we save this and try to change this, it still works.
* So now we got this same logging functionality outsource centralized in a service, and therefore our code here is a bit leaner.

Text

Description automatically generated

* And I guess you can imagine how in bigger applications where you have duplicate code, a service can really help you to get more dry, to don't repeat yourself all the time, but instead you really cleverly outsource your code into a service and have it there.