**147. Protecting Child (Nested) Routes with canActivateChild**

* Instructor: In the last lecture, we added the can activate guard and it was working fine, but it was working for our whole servers path here.

Text

Description automatically generated

* Now we could grab it from here and add it to our child's to make sure that only the child's are protected, the children, and not our root path.
* But that is not the easiest way because if we add more child items, we have to add can activate to each of them.
* **canActivateChild**
* There is another guard we can use.
* It's pretty similar to can activate.
* It's called can activate child.
* So let's implement this interface two.
* You need to import it from at angler router.

Text

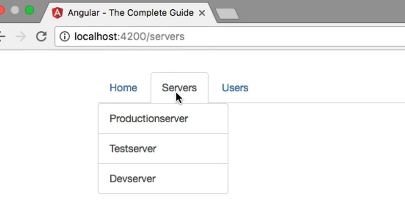
Description automatically generated

* And this interface requires you to provide a can activate child method in this class, which basically takes the same form as the can activate method.
* So, it has the route and state and it returns an observable promise or bullying.
* So this is the form of can activate child.
* Well, and since this is exactly the same form and we want to run the same logic, we can simply call, or excuse me, return.
* This can activate.
* So the other method we already included and here we simply pass our route and state as arguments, since can activate expects these two arguments.
* What is the advantage of adding this then? Well, since we added the can activate child interface, we can now use a different hook here in our routes.
* Instead of only using can activate, we can use a different one.
* And I will split this over multiple lines to make it really clear on what's happening.
* I will comment out can activate and instead add can activate child.
* This now also takes an array of services which act as guards, which implement the right interfaces.

Text

Description automatically generated

* And here we can still add the off-guard because the off-guard now is able to do both.
* Protect a single route, since we have can activate implemented, or all child routes, since we have can activate child implemented too.
* So if this in place, we should now be able to go to servers.
* This works.
* But if we try to load a single server, now we get redirected back because now only the child routes are protected.



* Something I can also prove by trying to go to slash one edit.
* This also navigates us back.
* So now this is defined grain control, you can implement to protect a whole route and all its child routes or just the child routes, depending on which behavior you need in your app.