URL Authorization

00:00: Welcome to Lesson Three of Module One, URL Authorization. So we're going to go into the basics of URL authorization in this lesson and keep in mind that we're only going to explore the basic functionality because there is a separate lesson, where I focus on the more advanced use case.

00:18: So what are we going to focus on? Well, first of all, we're going to learn how to configure authorization. We had a quick look at authentication in the previous lesson. And now we're going to start looking at authorization. We are going to make use of some extension points from the base class to set up some custom authorization rules, and finally we're going to have a very quick look at all possible options for URL authorization. We are not going to implement all of them because as I mentioned there is a separate lesson for that, but we are definitely going to implement a couple.

00:52: Alright, so we are back in our security configuration here. And for the first time, we'll extend one of the methods out of the base class and we will leverage that extension point to configure authorization. So let's do that. And you can immediately see that this is actually a method that overrides the configure method out of the base class. The very first thing we need to do here is we need to understand the behaviour of that base configuration, of that base method. So let's step into that code and let's see exactly what the base configuration is doing because we are overriding that configuration so we need to know what we're changing.

01:32: Okay, so this has a simple implementation, let's have a look at what it's doing. The first thing we notice is this authorize request API, which is defining all of the requests in the application as needing authentication. So this basically defines that every request will require it to be authenticated. We then see the form login configuration, the one that we were benefiting from earlier when we saw that by default Spring security was autogenerating a form login for us. So what we're going to do now is we're basically going to override this entire configuration and we are going to provide our own. However, in order to do that and really not affect too many things at once, we'll simply copy paste this existing configuration so that we can continue building on top of it. Okay, so by doing that, at this point, we haven't changed anything. What we did is we have simply overridden the configure method but since we copy pasted that exact configuration we have not really changed any of the actual behaviour.

02:37: So now let's say that we want to change that behaviour. Let's say that we want to make sure that, for example, the delete operation in our application gets some different authorization requirements. And the reason for that in this example is because delete is really the kind of method that needs some extra authorization requirements, because it is a sensitive method that actually does delete data from our system. So let's set that up. Well, the way to do that is of course to keep using this API. You can already see just by looking at this configuration how fluent and how easily readable the configuration really is, so we're going to basically continue to build on that. Okay, so let's look at what we configured here. The first thing we did is we used the antMatchers API to have an ant expression matching the URL pattern. And as you can see, we are matching the slash delete pattern, meaning everything that has the delete prefix will now match this exact pattern.

03:42: Now after we match the pattern, we configure the authorization for that specific pattern and this is where things start becoming interesting because we're going to go over a few of the main types of authorizations, a few of the main API methods that we have available here, and then we're going to have a very quick pass over the rest of them. We are only going to focus on a few of them now because we do have a dedicated lesson focused specifically on more advanced authorization rules, but for the time being let's have a look at these ones.

04:14: Okay, so let's first discuss hasRole. HasRole is the simplest option you have when you're configuring the authorization of your URLs. And of course being that simple, it's very straightforward to understand. The whole point here is that your user, your principal needs to have the ROLE_ADMIN authority. It is important to understand that hasRole will check for a "ROLE_" prefix authority. So that means that in this case, if we use the hasRole admin, then the check is not actually going to be against an admin authority it's going to be against a ROLE_ADMIN authority. So slightly different than what you might expect. This was the old behaviour in Spring security. And if you don't have to do that, if you don't want your roles or your permissions to be prefixed with ROLE_, then what you need to do is you need to switch. So this would be the way to do it. If you just want to keep your authorities clean and simple and you don't want to have those prefixed, then hasAuthority is the API you're looking for. Alright, so these two are of course very similar, hasRole and hasAuthority and these are two of the four that we're going to look at now.

05:30: Now, what are the other two? Well, we have two alternative APIs that will allow us to check multiple authorities or multiple roles at once. So let's have a look at that. So the two new operations
are hasAnyAuthority or the corresponding hasAnyRole. So these are of course pretty straightforward. They allow you to check a single or multiple authority. So let's see how that would work. If for example, we would want to check two authorities here and either of those authorities would be okay. So this is pretty much it. Now of course Admin2 may not be something you want to go into production with but this is of course just an example. So in this case, any of these two authorities, Admin and Admin2 will be okay for allowing you access to delete operations. And the role API here is going to work exactly the same way but just keep in mind that if you're using hasAnyRole, then that prefix will still be involved.

06:30: So that means that basically the check and the name of your authority needs to be ROLE_ADMIN or in this case, ROLE_ADMIN2. So pretty simple, pretty straightforward but of course one of the best and more flexible ways to really check authorization in your system. Alright, so I mentioned that we're going to end by looking at other API operations. We're not gonna discuss them but we are going to have a very quick look at the entire API here. So let's have a look.

07:03: First of all, we have the ones that are prefixed with has. And we already know hasAnyAuthority, hasAnyRole, hasAuthority, and hasRole. The only one that we haven't discussed is hasIpAddress. This is super interesting although you're probably not going to use a lot of it in production. It is still interesting and sometimes very helpful to be able to pinpoint a specific IP address. But now of course the broader API is here so we have here access. This is going to allow you to use expressions. We have anonymous to basically signal that any type of access is okay for that URL. We have authenticated to just say, "Well, we need to be authenticated here." That's it. We don't need any special privileges or any special authorities, but we do need to be authenticated. We have of course denial which is going to basically restrict any kind of access. We have fully authenticated. This is an interesting one because this is going to tie into Remember-Me so we're gonna have a look at this in the Remember-Me module.

08:10: Then we have the ones that we already discussed, hasAnyAuthority, hasAnyRole, hasAuthority. And then we have hasIpAddress. We already had a look at that. HasRole, we already discussed that as well. We have not which is super interesting because it allows you to do some interesting training here. We have Remember-Me and obviously we're gonna discuss this one in the Remember-Me module. And this you can see, that is about it. So this is the entire API when it comes to authorization of URLs. Now of course, the access operation having support for expressions will open things up quite a bit. We have a lot of options when we are allowed to use expressions, but for the time being this should really give you a good understanding of what it means to secure a specific URL and a good idea of how to do that.

09:03: Alright. And now for the final part of this lesson, what I want to do is with this extra authorization on the delete operation, I want to go into the application and I want to log in with a user that does not have the admin authority and of course I want to trigger a delete. So the goal here is to basically see the authorization working and the delete not being available. Okay, so we are in the application, let's go ahead and create a user. And now with the user created, let's try to delete that user. And keep in mind that the delete operation now needs the currently authenticated user to have this admin role, which we do not have. So this operation should not work. And here we go. We are getting back the HTTP 403 Forbidden. As we were expecting, this operation is now unavailable for our currently authenticated user.

10:05: Alright. So what are the takeaways of this lesson? We started by extending our existing configuration in setting up some authorization rules and then we went through all of the possible options for URL authorization. We had the practical look at the few of them but we quickly went over all of the options just to get an idea of how flexible the framework really is. Alright. Hope you are excited. See you in the next one.