# Remember Me with Persistence

**00:00:** Welcome to the third lesson of Module Three, Remember Me with Persistence. All right. So in this lesson, we are going to move from our previous Remember Me implementation, which was cookie based to an implementation that is backed by persistence. This is the second option that the framework provides. And this is what we're going to explore now. All right. So let's go right to the code. All right. So the very first thing that we're going to do here is we are going to start using MySQL. So just for this lesson, we are going to switch from the in-memory database that we've been using to using MySQL. And we're only doing that for this lesson because it's easier to look into MySQL and show what's happening there than to look into H2 or the HSQLDB database.

**00:48:** And let's now also update our Spring Boot configuration to make sure that we are actually using MySQL. We're going to do that by simply adding some configuration properties right here into the application properties of the project. So as you can see, we are configuring the data source. We are of course configuring the user name and a password, and we are specifying the driver. So with the help of these four configuration properties, we are now using MySQL as our main persistence. All right. So on to the first thing that we're going to do here. And that is going to be changing our security configuration and more specifically, the Remember Me configuration to stop using the default cookie-based solution, and to start using persistence. So the first thing we need to do is we need to wire in the data source.

**01:39:** And now that we have the data source available, we'll need to define a persistent repository for our Remember Me cookies for our tokens. So let's do that. So let's have a look at this bin. First of all, you can see that we're defining this persistent token repository, which is an alternative for the in-memory token repository. If we have a look at the hierarchy of this interface, we can see that it has two implementations. One of them is the in-memory one and this other one, the JDBC one, is the one that we're going to start using to get these tokens persisted. And of course, the next detail that you'll notice here is that we are making use of the data source that we have just wired in. That is why we needed the data source in order to define this JDBC token repository and allow that to connect to the database.

**02:31:** And let's now finish the configuration by connecting all of these pieces together. So what that looks like is that for the Remember Me configuration here, we are going to replace this part with the persistence implementation, and we're basically going to start using this new token repository. And that wraps up the security configuration for Remember Me. Now, the Remember Me solution is no longer just using the default, the cookie-based solution, and it's now using persistence to store the tokens. However, we are not done. We also need to create the database structure that this whole token persistence will need in order to actually be able to persist the tokens. There are of course, a few good ways to do that. In this case, we're going to be using data.sql. That's the file that Spring Boot makes available for this exact type of use case. However, just keep in mind that there are multiple ways to write and run SQL code on the startup of the project. We're using Boot now, but again, there are many other ways to do it.

**03:36:** All right. So we're now going to add our script into this data.sql file. And notice that this file is only creating the test user for now. So let's do that. Let's create our table structure for the new

token persistence. Okay. So pretty straightforward, but let's still go through the structure and understand what's going on. So any of these persistent tokens, any of the Remember Me tokens will have a reference to the username right here, then to a series field, which we're gonna talk about later. Then of course, the actual token value. And finally, a last used timestamp. This is, of course, going to track when did that last persistent login happen. So again, pretty straightforward, but that's basically going to be enough in order to get our token stored and of course, in order for the Spring Security implementation to actually leverage the underlying structure.

**04:31:** All right. So now that that's out of the way, let's start up the system and let's see this in action. So what we're going to do here is we're going to make use of the MySQL work bench. This is why we chose MySQL. And we are going to connect to the local instance. We are going to check out this table structure right here. And then, we're going to see how the token gets created and of course, how the token gets removed when we login and then when we logout. So let's first check out the table structure. Here's our persistent login table. And of course, the table is empty. So let's now switch to the running application and let's login with Remember Me enabled. Okay. So we are gonna login. We are going to enable Remember Me. And we are logged in. So let's now see that the token actually exists in the database.

**05:25:** And here we go. We are getting the username, the series identifier, the token itself and the last used timestamp. So everything looks in order. Let's now switch back to the application and logout. But actually, before we do that, before we logout, let's see the cookies. So as expected, the Remember Me cookie is of course still there. The value of the cookie, this content value, is different because it's using the new algorithm, but the cookie itself is still there as expected. All right. Let's now close this off and let's log out and see the persistent login being deleted from the database. Alright, we are now logged out. Let's switch back to the work bench and let's see what data we have in this table. And we can immediately see that the persistent login, the token, has been removed correctly once we've logged out of the application. Alright, so we now have a fully working Remember Me solution, no longer using the default implementation in Spring Security, but this time using the persistent implementation.

**06:31:** Alright, before we wrap up, let's go through the logical flow of this persistent Remember Me mechanism and let's understand exactly what's going on behind the scenes. So first of all, a request comes in, and the Spring Security filter chain starts executing. And out of this filter chain, we will focus on the Remember Me authentication filter. That is, of course, going to be the main filter that drives the Remember Me process, so that's what we're going to focus on. The first execution step in this filter is to check if the Remember Me cookie even exists on that request. If the cookie does not exist, then the entire Remember Me filter will simply stop running and basically the execution will go to the next filter.

**07:14:** If, however, the cookie does exist, then we're moving on to decoding that cookie. Now, if the decoding of the cookie fails, then we're going to throw an exception. And of course, if the decoding succeeds, we are going to move on to the next step, which is going to be validating the cookie, and more specifically, checking the MD5 signature of that cookie. If that check fails, we're of course going to end up throwing the exception again, and if that check succeeds, we are going to move to the next step, which is going to be checking the user account and essentially performing authentication. As expected, if that last check fails, we'll also throw an exception, and

if it succeeds, we're going to move on to creating and of course persisting the authentication token in the database as we've just seen. And after that point, the next filter in the chain starts running. Alright, hope you're excited. See you in the next one.