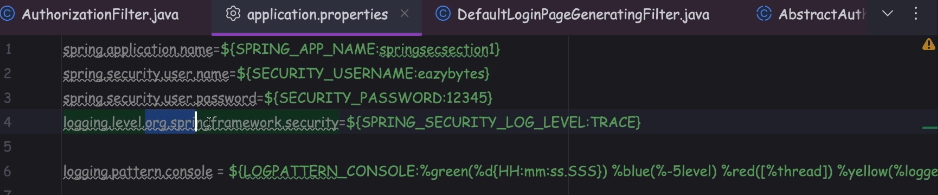
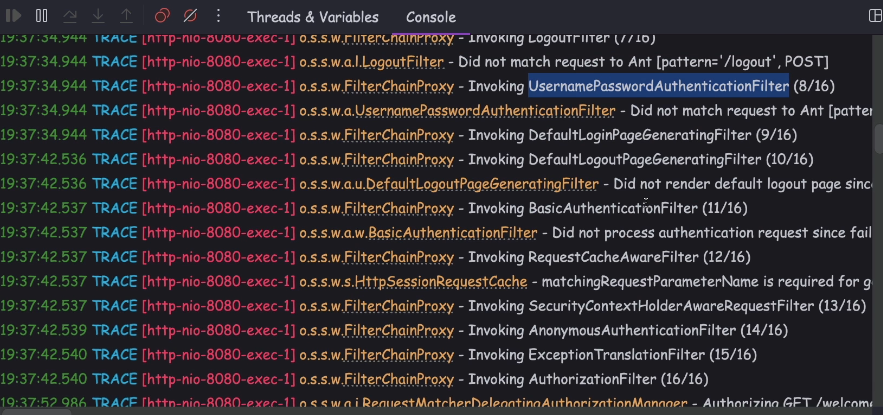
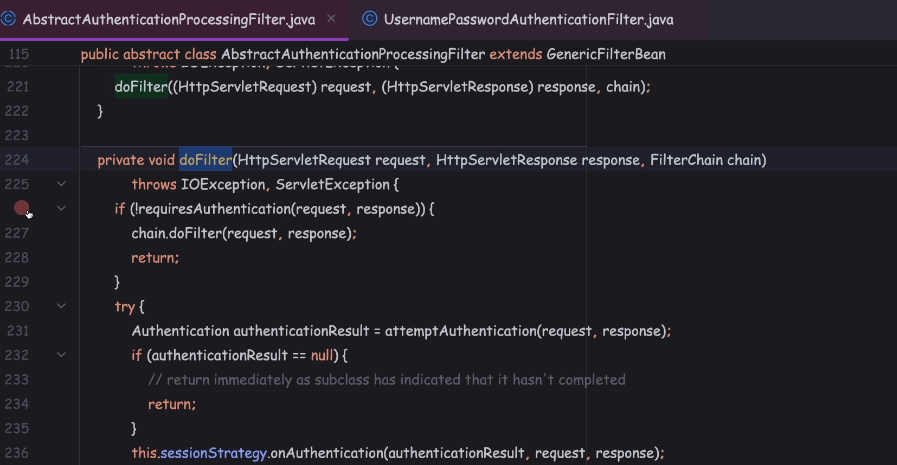
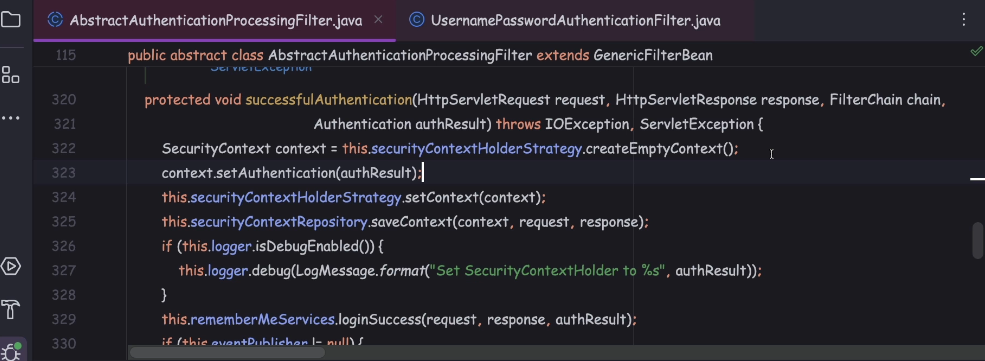
**1.spring security intro**

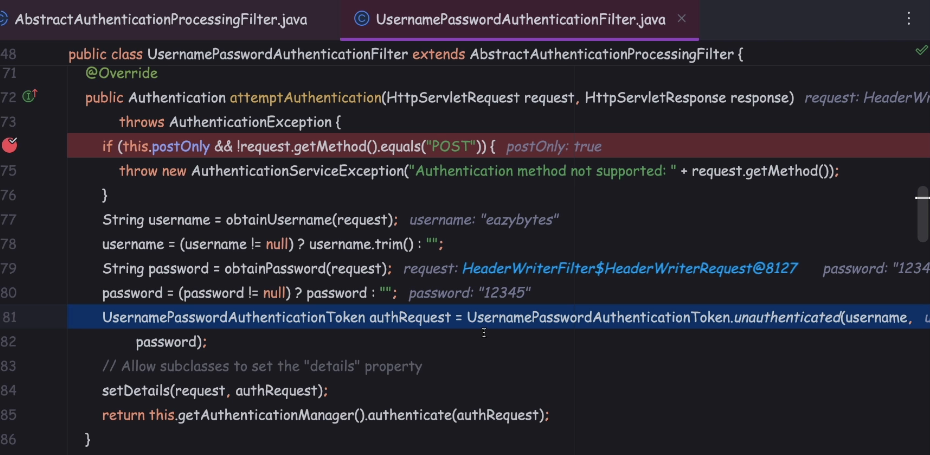
Demo of Spring Security internal flow

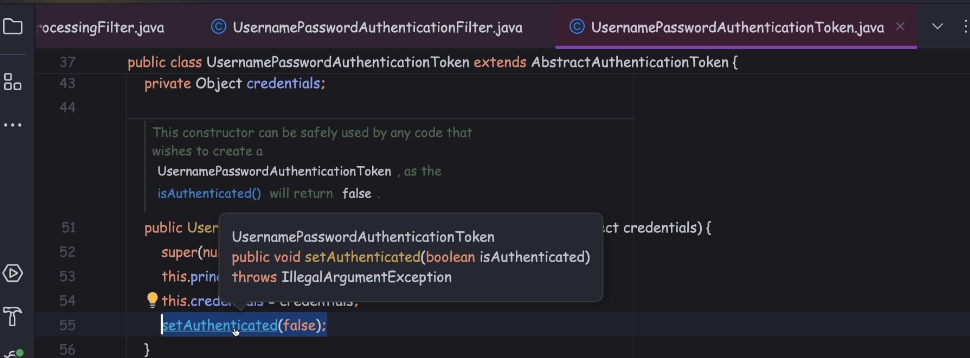


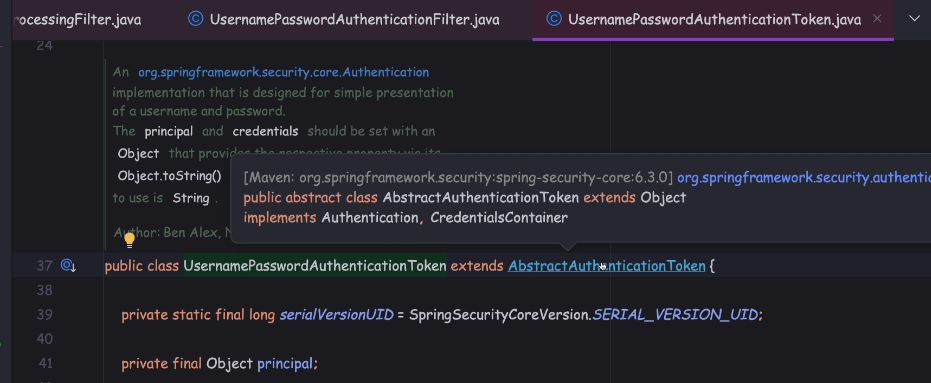




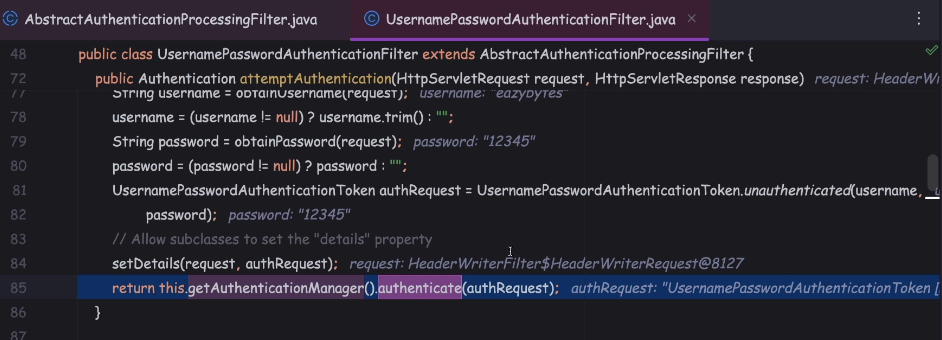


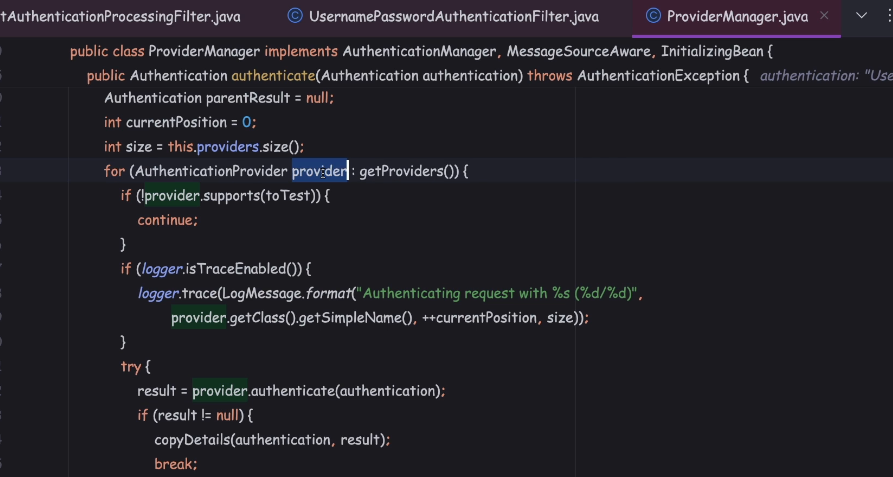


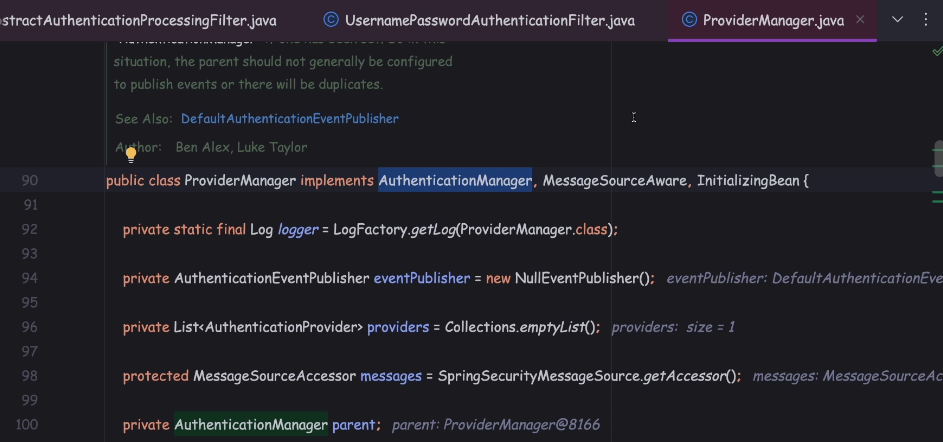


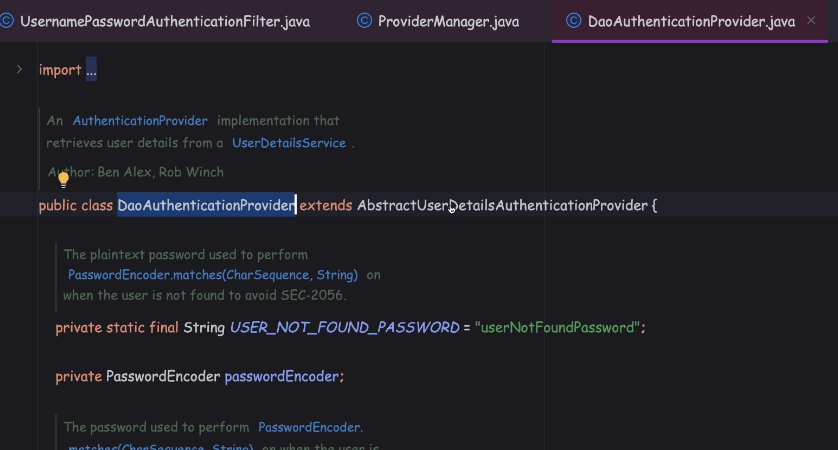


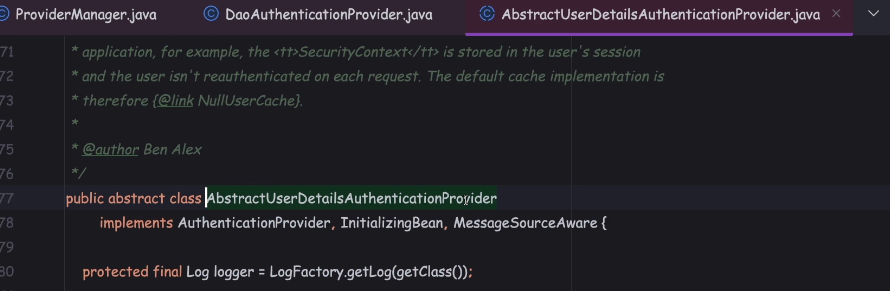




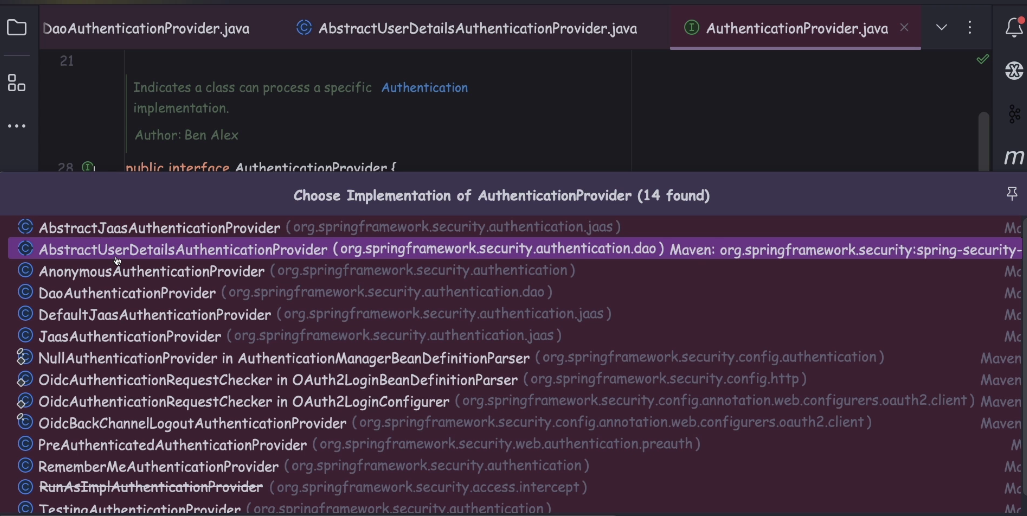




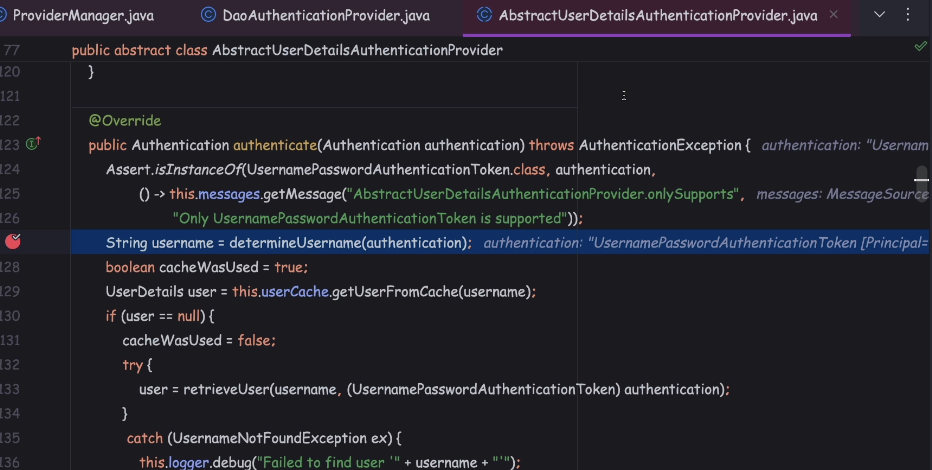


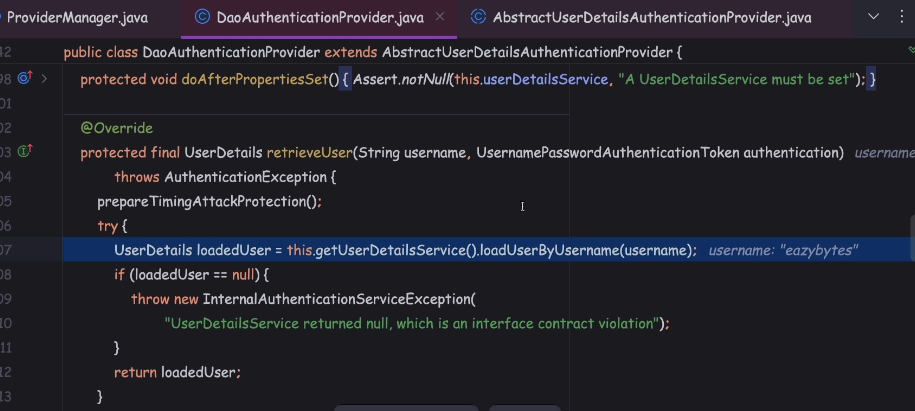


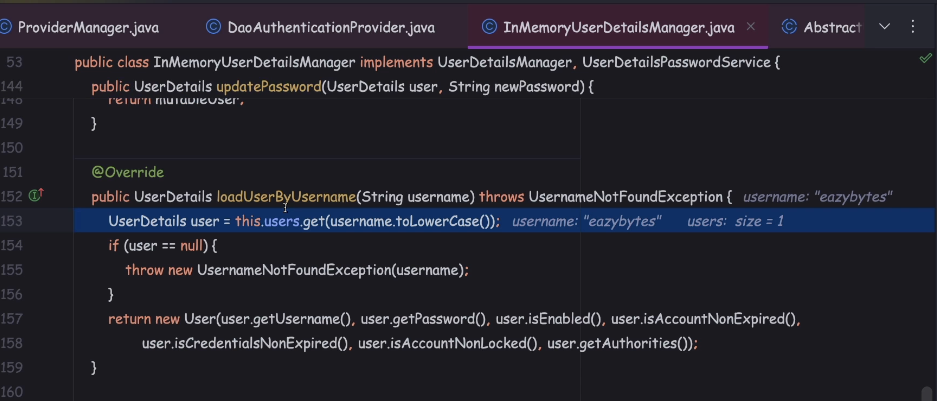


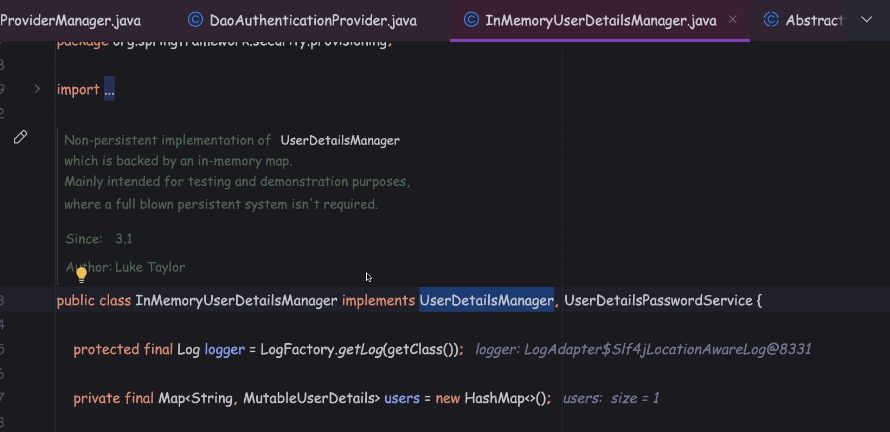


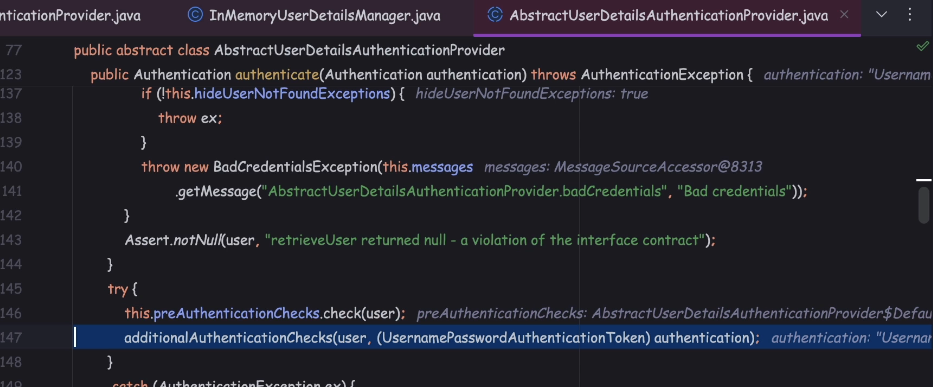
but in the default scenario it is going to invoke the DaoAuthenticationProvider.

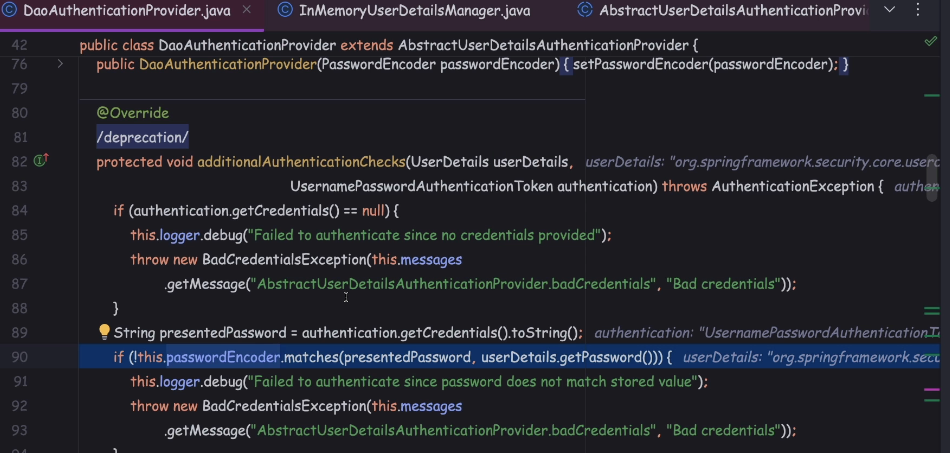






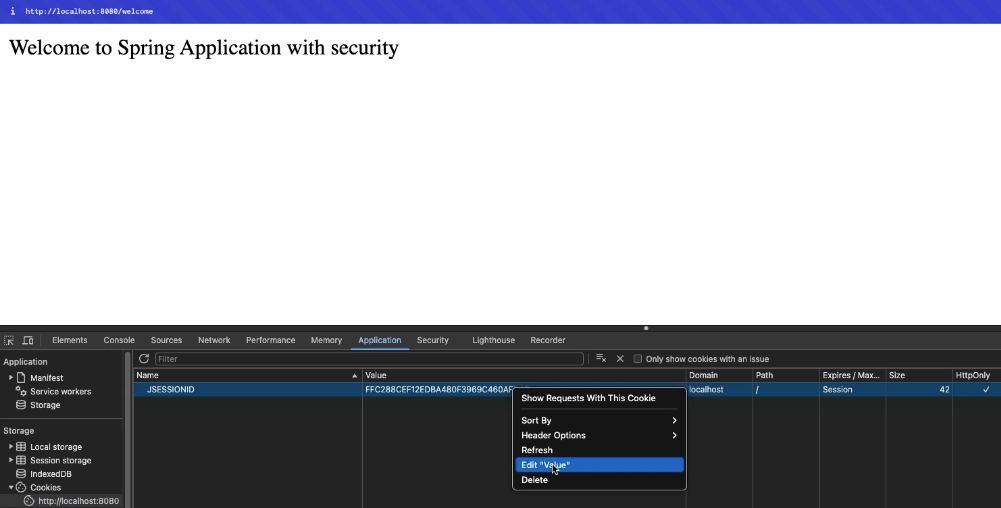


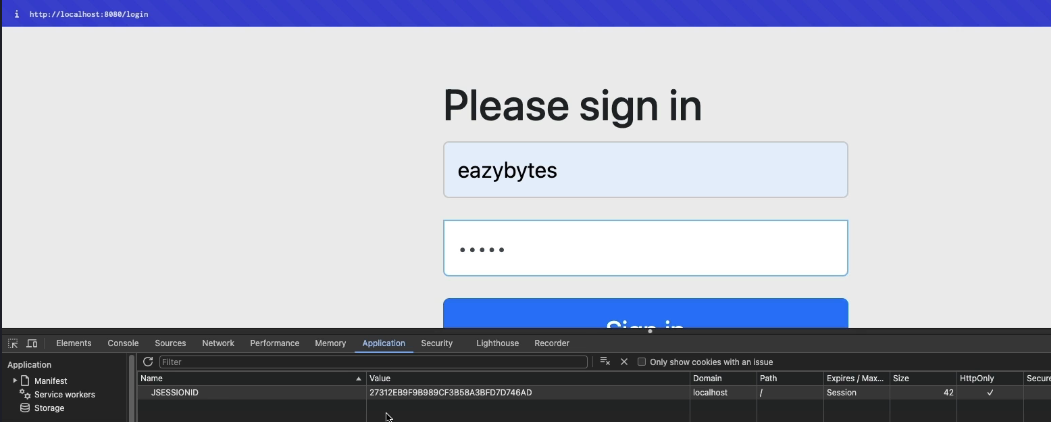


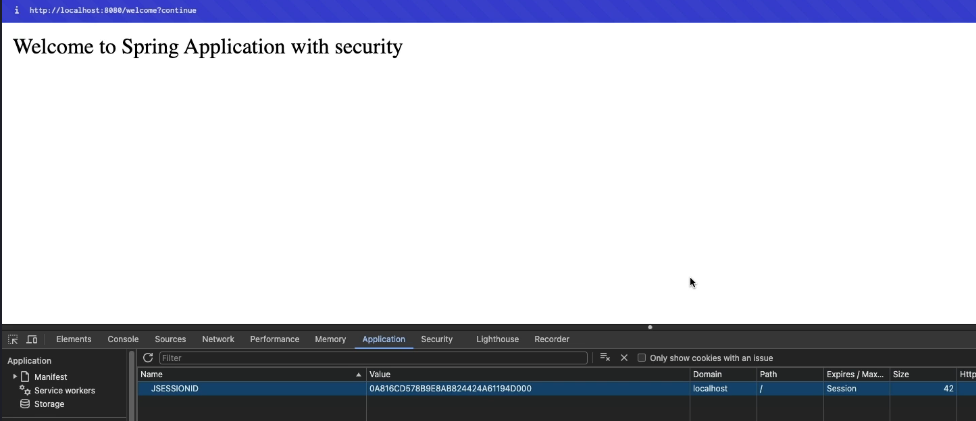


 Understanding on how multiple requests work with out credentials

Tamper jsession id







Spring Security will map to a successful authenticated session where my authentication details, they're going to be stored and they'll be reloaded and reused

every time when I try to access a protected API.

This way, Spring Security framework, also doing a smart job of not invoking the actual authentication every time.

If you try to invoke the actual authentication for each and every request,

then definitely it is going to have some performance impact.

But by using this JSESSIONID cookie, the Spring Security framework is doing a smart job here.

off course, I also want to highlight, this is the default behavior of the Spring Security framework.

There are better and advanced approaches also available, like JWT tokens, OAuth2.0 framework, OpenID framework.