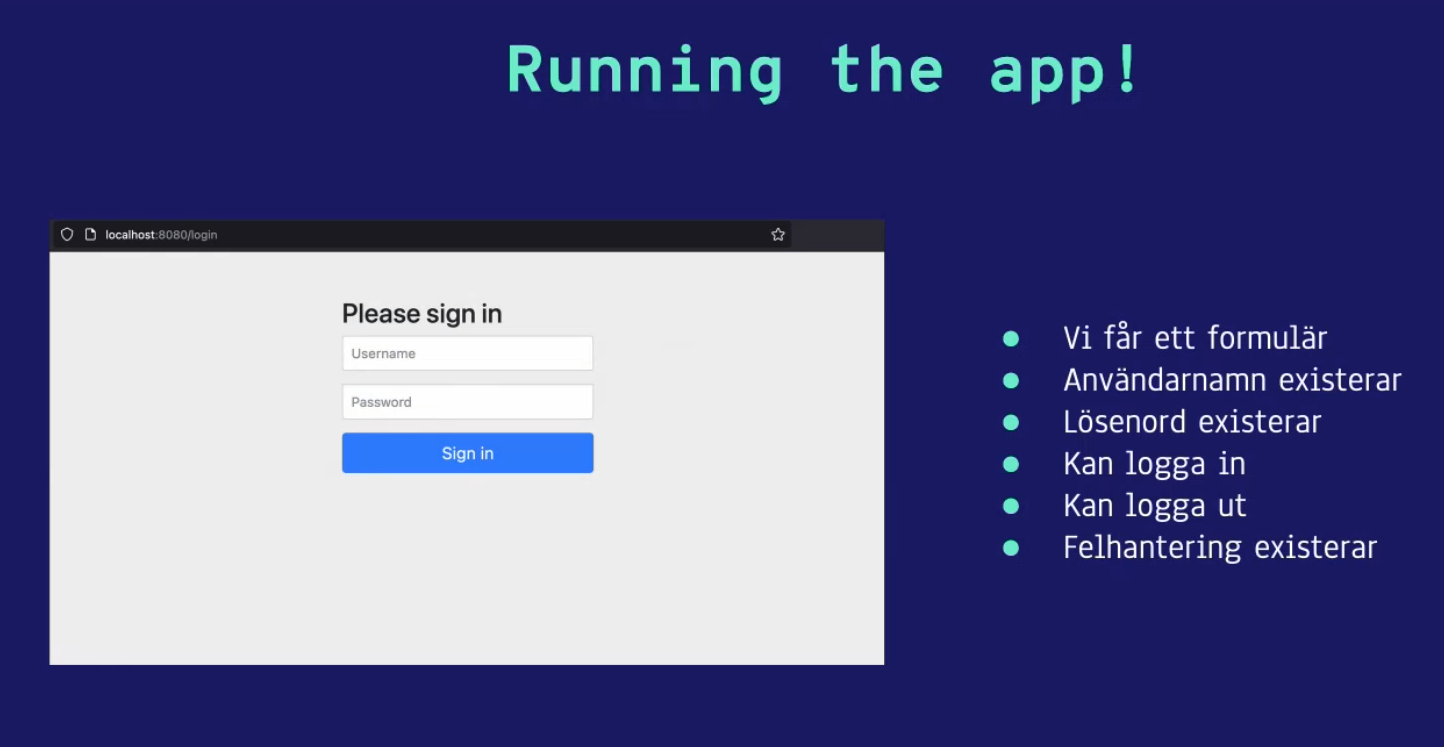
Steg 1.

Sätta upp spring security för login på sidan.



Om jag skriver /logout i endpoint så kan jag välja att logga ut.

Blir det login?logout så är det för att logout är fallout.

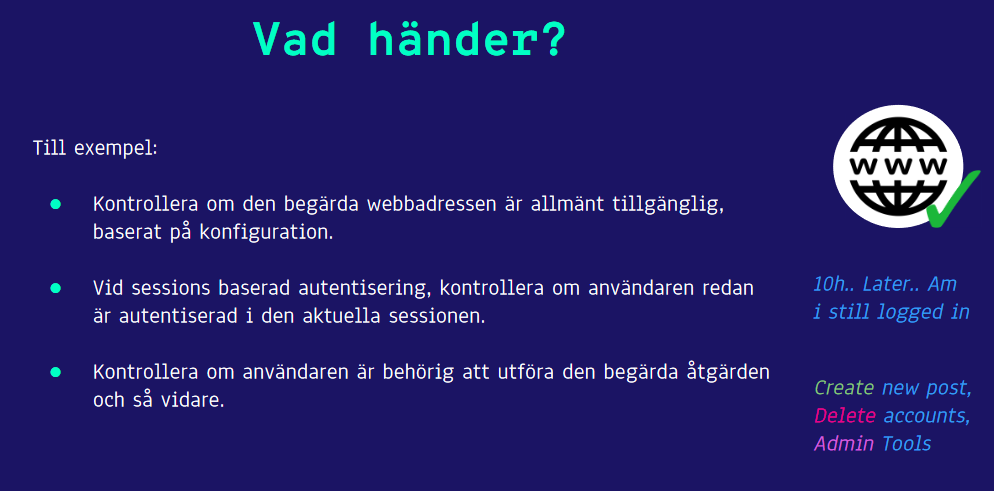
Spring security filter introducerar filter chains om fångar upp alla requests genom olika lager för att tex testa behörighet.



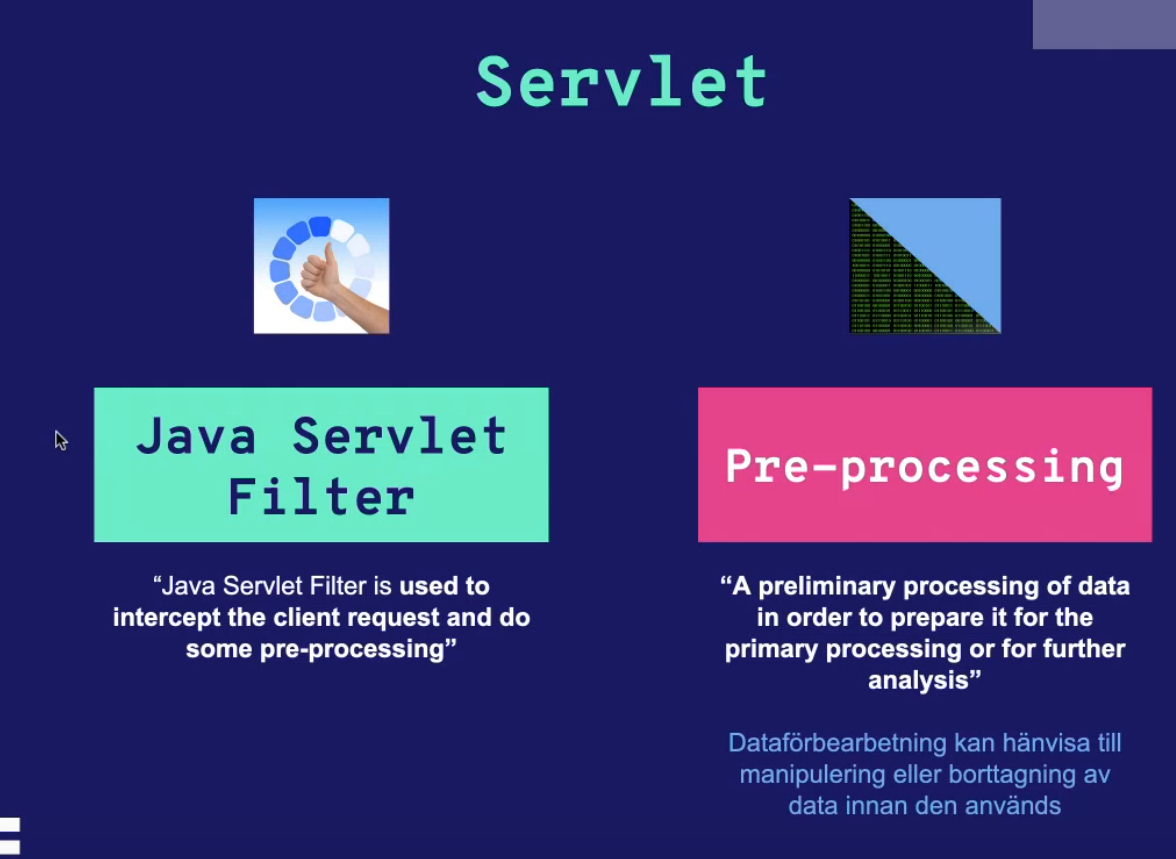
Kedjan består av olika filter och i bilden så ser man exempel på security filter A - B ända fram till N

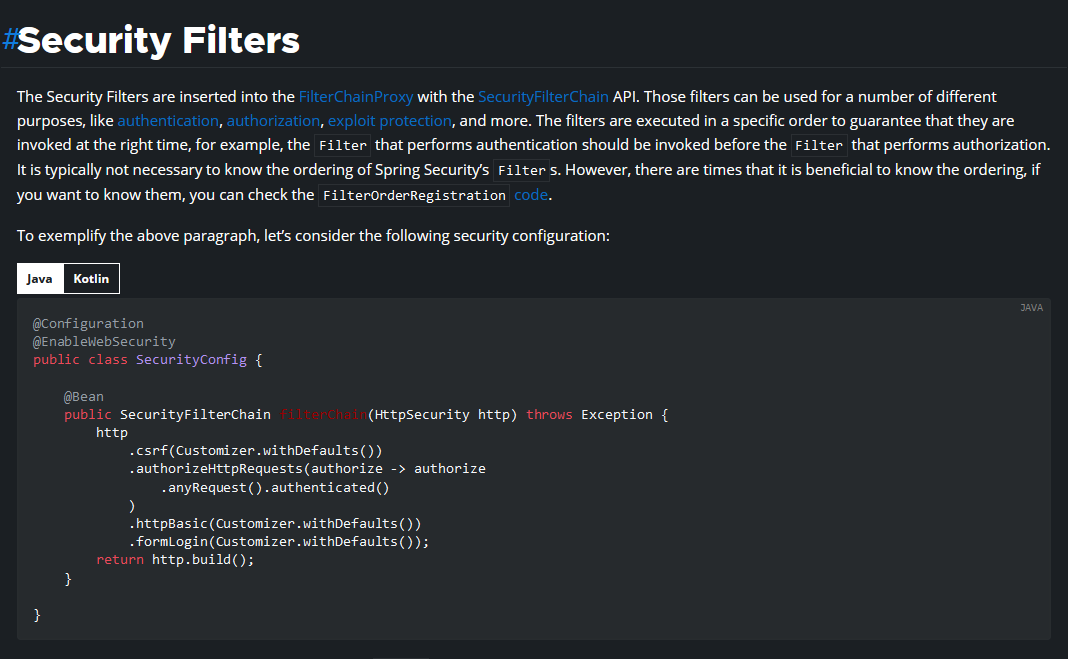
Så man kommer oftast till B för att skicka in request. Men mna såg att spring securityn drog igång så fort vi försökte komma in på local host (och med login screenen) detta för att varjue gång vi gör en request, så kollar den om vi är inloggade, har vi behörigheter? Finns tillåtelser att visa upp en html fil?

Kommer du in på företag med äldre versionen, var inte förvånad att det står ”override” på app security config. Men numera med 6.0 säger man att det här är en konfiguration och markerar det som en böna (beans) – beans är o kan vara ALLT i stort sett, objects, komponenter, services, repositories, autowired, den letar efter ”bönor”. Mna kan säga att den vill identifiera annotationer för att kunna överskrida.



Java servlet filter:





I exemplet ovan:

Vi har en class SecurityConfig mde två annotationer, en config annotation och en enablewebsdcurity, där vi slår på klassen med reqeusts

Sen annoterar vi en metod med @bean, en äldre version hade då visat @override istället.

Den tar in http security som en klass inom sin parameter och döps till http (man vet ofrtfarande inte riktigt om vad det här fakttiskt är?)  
Men sen ser man att man i httpn kan lägga till detta

http

.csrf

,authorizehttprequests osv.authenticated = innebär att man måste vara inloggad.

Sen i httpbasic med form login, är det som ska autentisera, så här styr man med andra ord vad som är tillåtet att göra med appen.

Saker som remember me, roles cookies, och vilka endpoints man får tillåtelse till kommmer nman kunna lägga till inom den här lilla koden.

Vad är det här http

.test

.authorize

. osv

Det är ett pattern och kallas för ”builder pattern” då den returnar http.build();

Mest till för att göra det enklare för ögonen. Med builder pattern sätter man det som är nöldvändigast, men också för at det ibland finns optinal parameters. Tänk att om det här var en konstruktor med typ 50+ metoder, då hade man behövt skriva alla 50 metoderna, med builder pattern, sätter man bara det som är absolut nödvändigast. Därför ser det annorlunda ut än vad man kanske är van vid.d

Authorize -> authorize .anyRequest(/).authenticated() har också lambda, hade I http basic där man bara gör en sak, så behövs inte det men hade du gjort två saker där nedan hade det också varit bra att köra lambda då.

Docs för projekt:

Layers och dess funktioner:

Model: data model, entities och repos

Order:

Product:

User:

Entity mappar den till en table i database, JPA creatar ett table som ocrrespondar till den här klassen i databasen @Id = primary keyn @generated value specificerar hur primary keyn genereras som i det här fallet genereras själv.

När koden körs och den är konnectad till databas med JPA configgat, så skapar spring boot en user i table med fälten id username password o email.

Repository: Lagret interfacear med databasen för att utöva CRUD

Service: Business logik

Controller: Hanterar http requests, invokar service layer