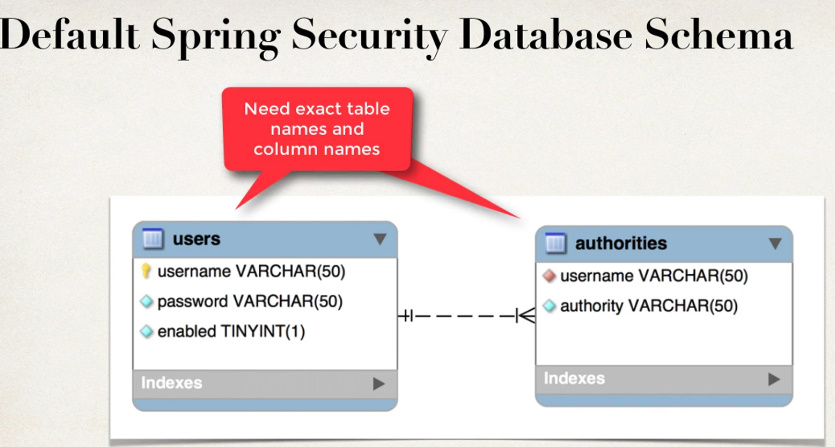
* Adaugarea userilor in memorie nu este o idee de loc buna.
* E mult mai practic sa-i punem intr-o baza de date si de acolo sa si extragem toate conturile lor.
* Spring Security include tot codul necesar JDBC pentru a citit datele din baza de date, deci noi vom scriem putin JDBC cod.
* Spring security ne permite sa cream custom table schemas, dar trebuie sa facem noi tot codul
* Aici vom folosi Spring Security predefined schemas, adica ale lui default

**Pasi**

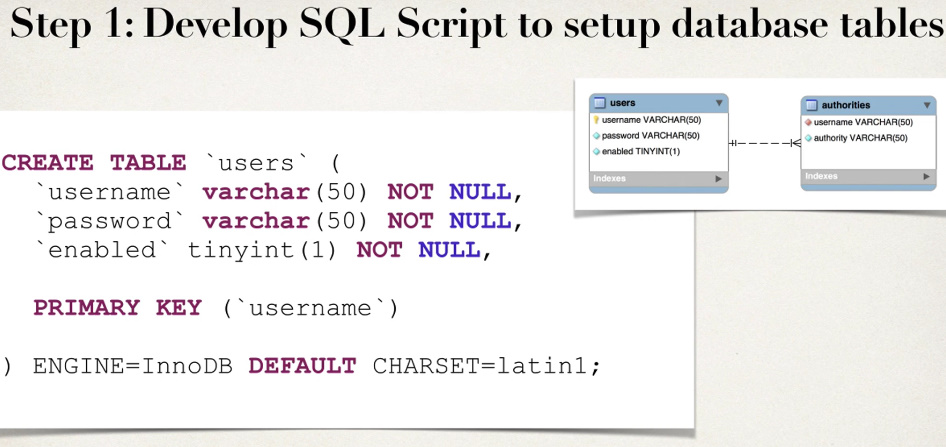
1. Cream tabelele in baza de date exact ca mai jos(acelasi nume, aceleasi coloane):

Spring Securiti foloseste ca default schema:



Deci, va trebui sa cream clase care sa incapsuleze aceste 2 tabele, anume cu numele si coloanele de mai sus dar va trebui sa cream si tabele exact ca acestea.

authority = role

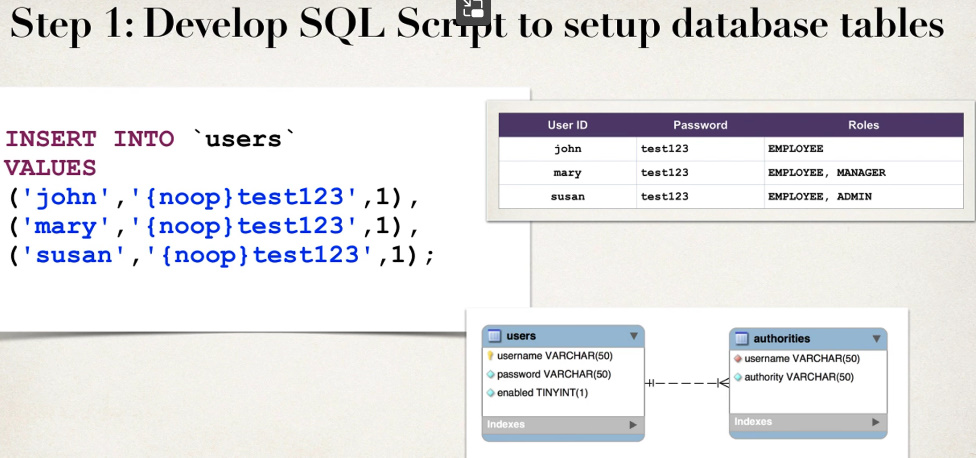


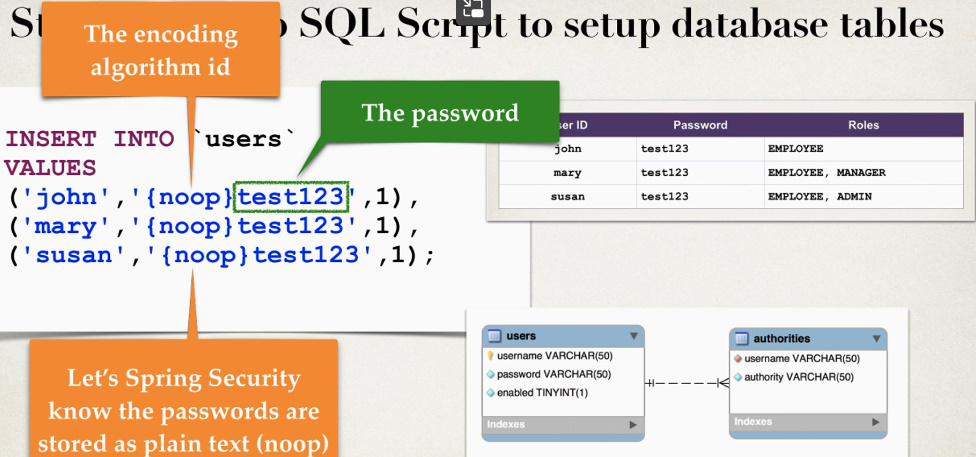


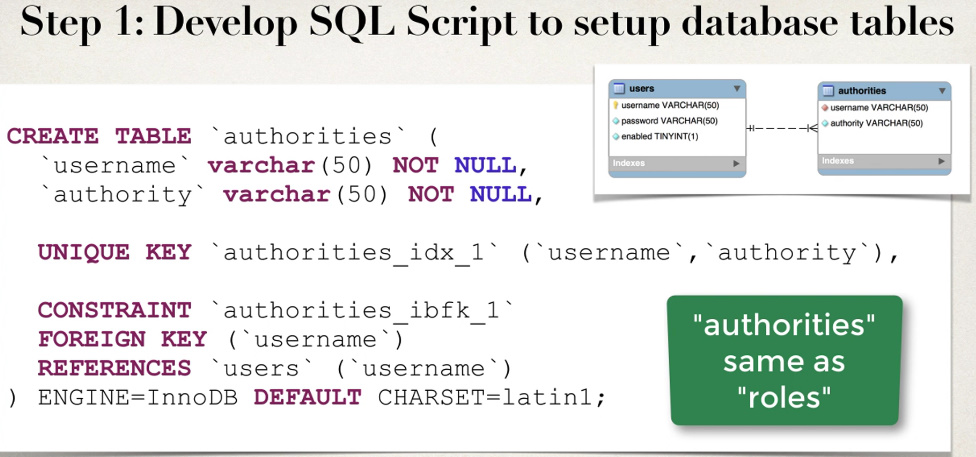
Deci, parola la user mereu va fi introdusa sub forma {id}password/password criptata, de ex {noop}test123

Spring trebuie sa stie daca parola stocata e criptata sau e direct pusa in baza de date, si anume acest {} ii spune

bcrypt e cel mai bun si recomandat



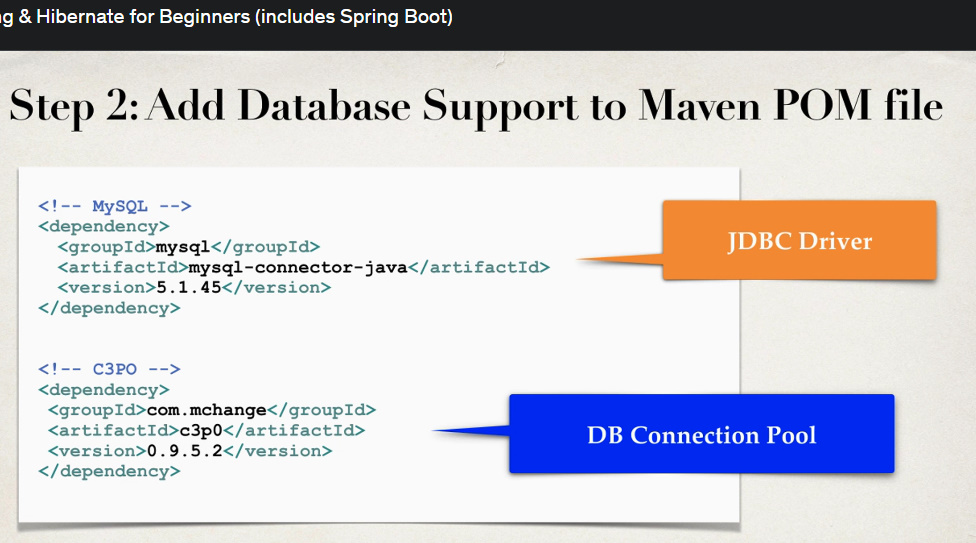






Atentie! La rol mereu Spring foloseste ROLE\_

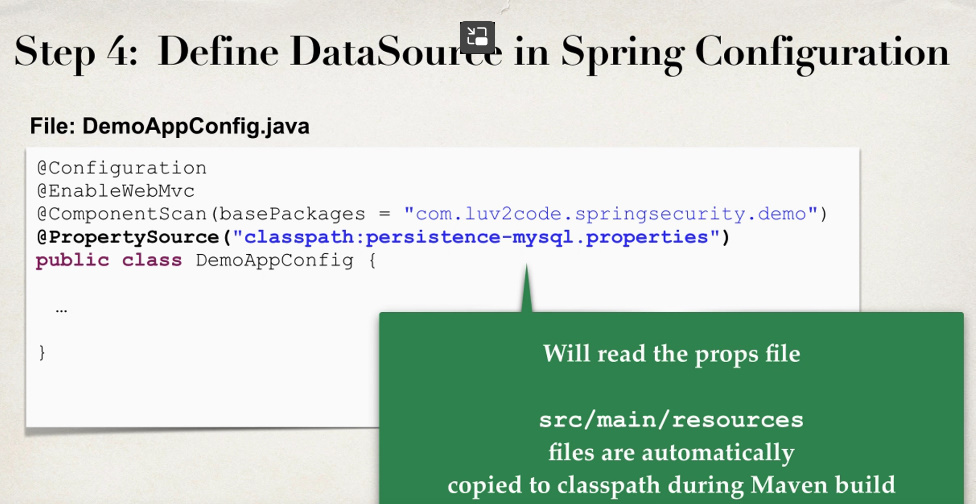
1. Adaugam dependentele

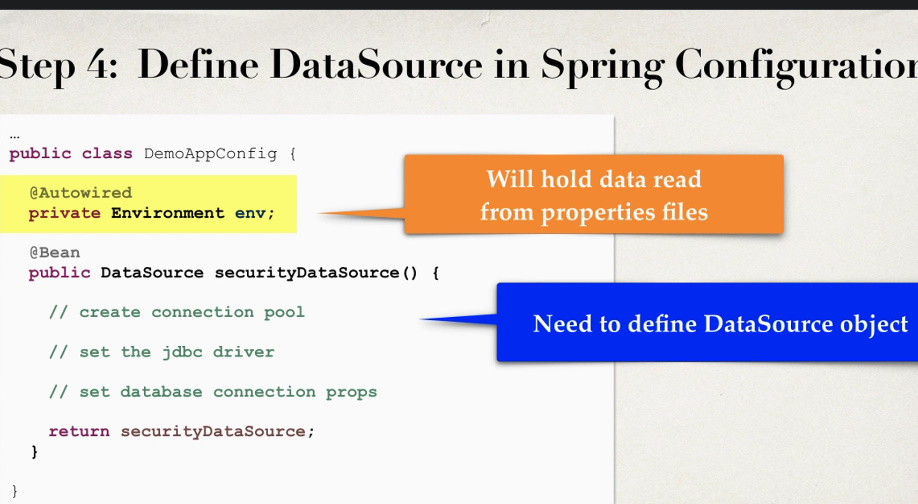


1. De data asta nu vom mai folosi xml pentru C3PO, ci un properties file



1. Adaugam acest property file in controller, in loc de <context:property-placeholder>:





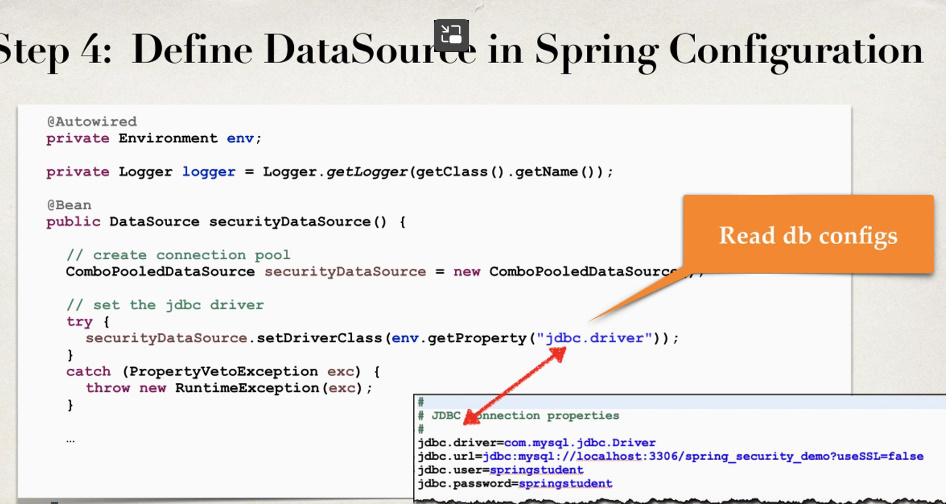
Deci, acest obiect de tip Environment va fi creat automat de

Spring si va contine toate datele din fisierul(fiierele) .properties declarat de noi cu anotatia

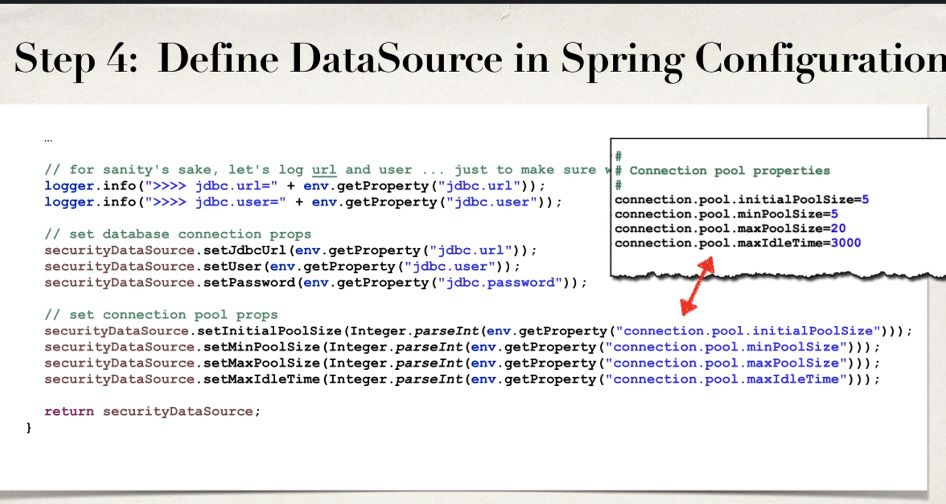
@PropertySource

Import:

import org.springframework.core.env.Environment;

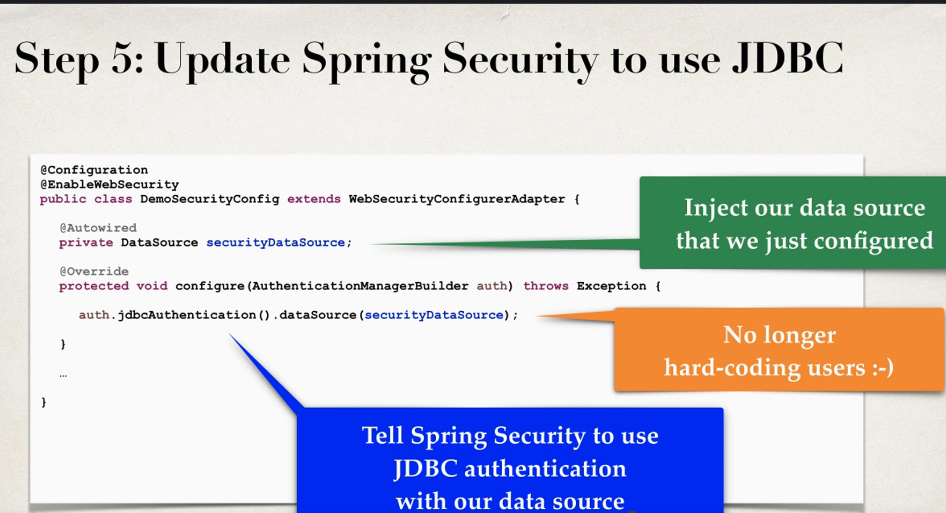


In mod normal, @Autowired nu e permis in configuration, dar exista unele exceptii. Putem pune @Autowired chiar si in Configuration, asa cum oricum Spring va intelege ca va crea Bean pe baza la anotatia @PropertySource



obiectul Environment mereu returneaza String, de aceea convertim in int

1. Acum trebuie sa lucram cu Spring Security Configuration.



* Spring Security se va ocupa deja de obtinerea userilor din baza de date.
* Nu avem nevoie de transactii sau beanfactory.
* @Autowired este aici permis mereu, spre deosebire de configuration pentru servlet
* Nu mai avem nevoie de obiectul UserBuilder
* Important! Pentru fiecare login, Spring Security mereu face un query pentru a extrage userii si parolele lor