**Authentication**

The way to configure authentication in Spring security is by affecting what’s known as AuthenticationManager.  
  
An AuthenticationManager is **the main strategy interface for authentication**. If the principal of the input authentication is valid and verified, AuthenticationManager has a method authenticate which returns an Authentication instance with the authenticated flag set to true and throws an Exception if the principal is not verified.

We directly don’t interact with the AuthenticationManager but we work with a class known as AuthenticationManagerBuilder.  
We use the AuthenticationManagerBuilder to configure what the authentication should actually do. For e.g. if we want a inMemoryAuthentication, then the AuthenticationManagerBuilder will ask you the username, password and role. And once you provide these it actually creates a AuthenticationManager with your values.

Now the question is how do you get hold of the AuthenticationManagerBuilder?

* By extending a class WebSecurityConfigureAdapter which has a method configure which takes in as a parameter as a AuthenticationManagerBuilder, ***configure(AuthenticationManagerBuilder auth)***we could override this method and give our own details.

PROGRAM to configure the AuthenticationManagerBuilder:

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter {

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception {

auth.inMemoryAuthentication()

.withUser("Nilim")

.password("nilim@123")

.roles("USER");

}

***Encoding the passwords:***

Use is to not save the passwords in plain text format.  
PROGRAM: Just the basic password encoder.

@Bean

**public** PasswordEncoder getPasswordEncoder() {

**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

***HOW Spring Security Authentication works?***

The inputs to an Authentication are the credentials of the user who is trying to get authenticated.  
  
And the outputs could be a Boolean. But in Spring Security if the input has been authenticated it doesn’t return a Boolean, rather it returns a Principal (i.e. the information of the logged in user).

When Spring Security performs authentication it takes care of both the input and the output using an object of type ***AUTHENTICATION*** which is an interface which holds the principal after the input gets authenticated.  
So, this ***Authentication*** interface can be referred to as the data transfer object for the authentication credentials and the holder of the principal after the authentication is successful.  
  
Now, who actually does the authentication?

* Common way is by using Providers:

There is an interface known

***AuthenticationProvider*** which has a method ***authenticate()***