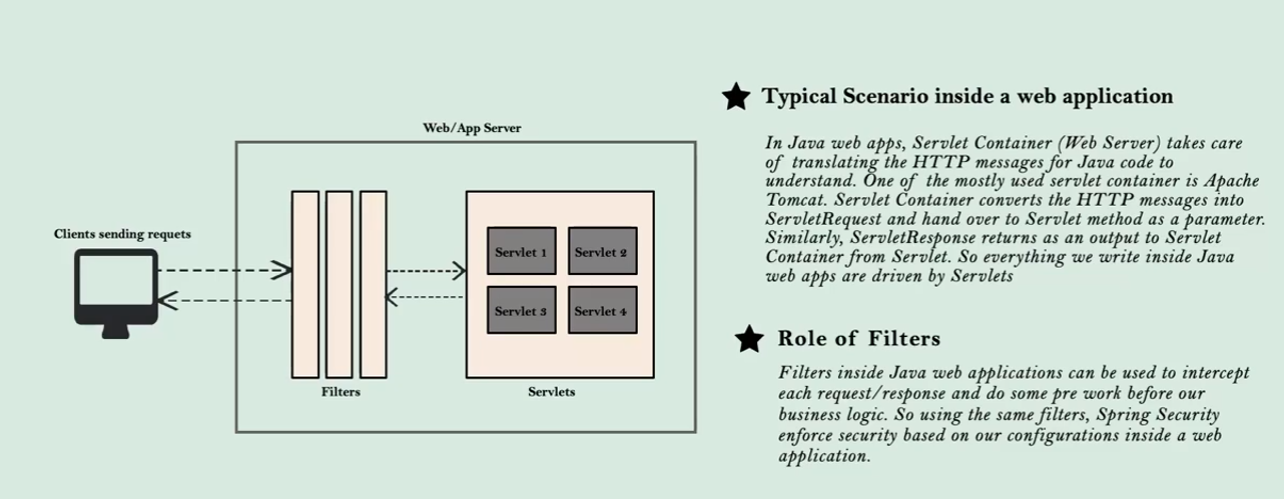
Springboot filter and security

Spring security internal flow

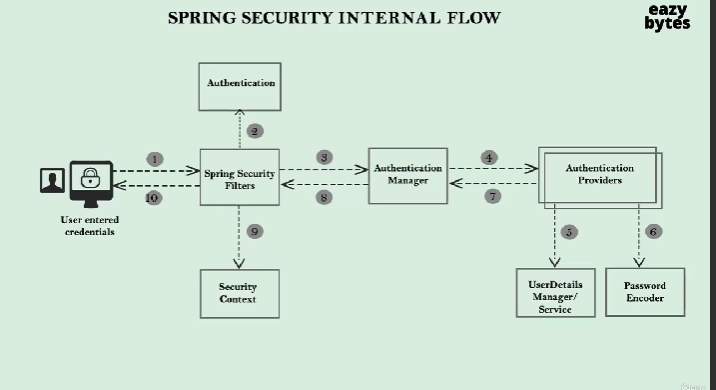
Servlets and filters

As you know client and server communicate with the help of http requests but our java code doesn’t understand it. Hence to over come this there is a layer added between java code and client that contains servlets and filters. Servlets are responsible to convert http requests into understandable java related objects. Hence servlets are major components of web application.Spring boot internally uses servlets.



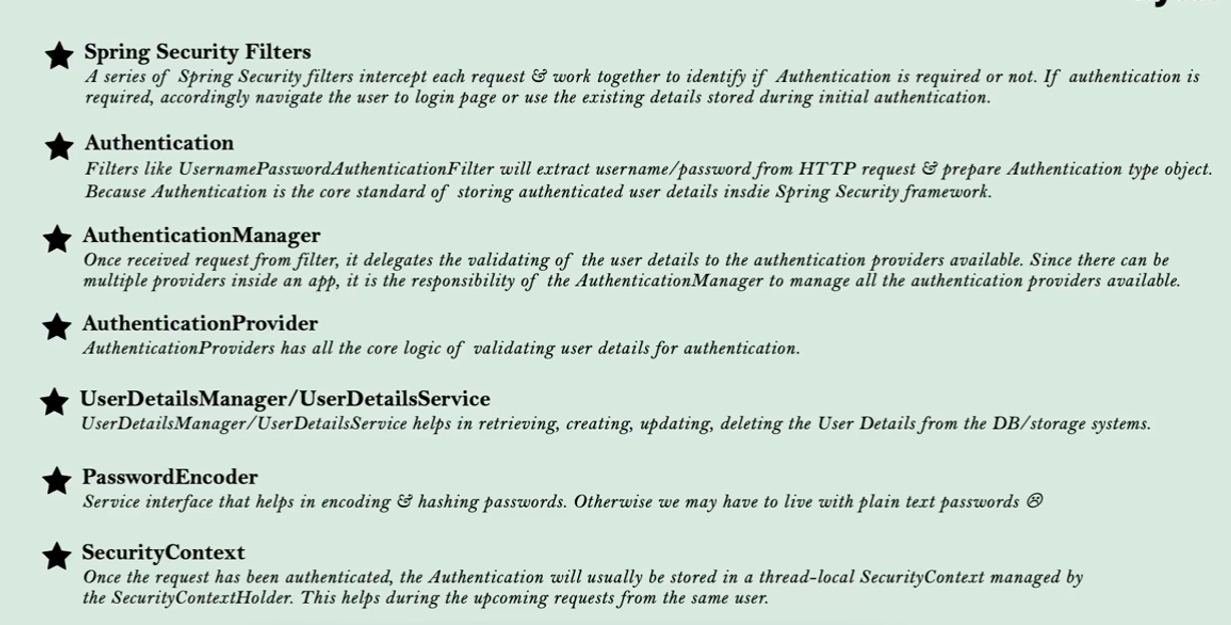
Filters are used to manipulate request / response from or to the client and spring security makes use of these filters and enforce security.

After adding spring security you know that application will provide us with login page and also when we login then we don’t have to re enter username and password again and again. This is done by default configuration.

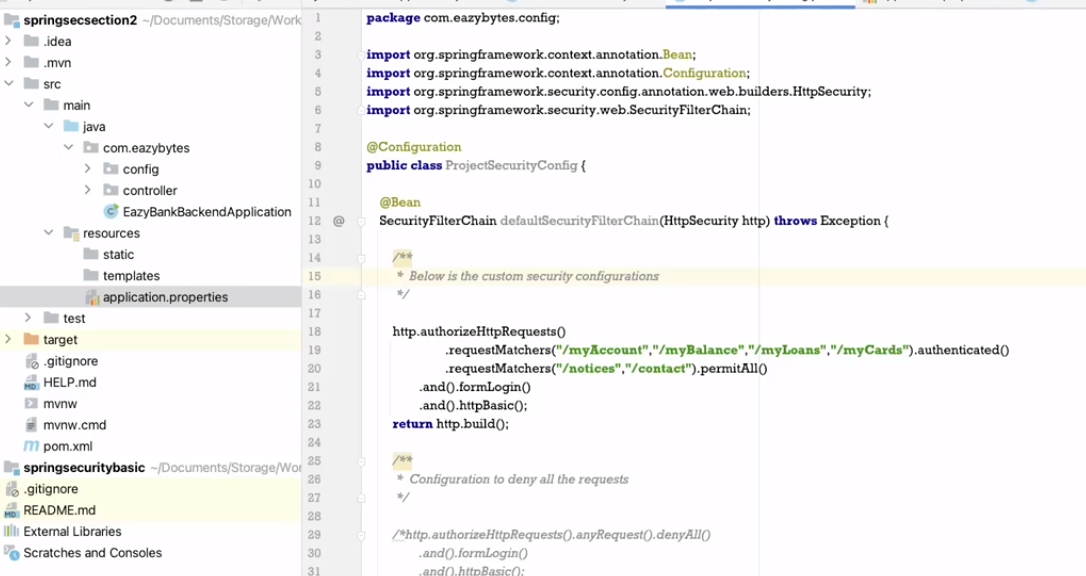


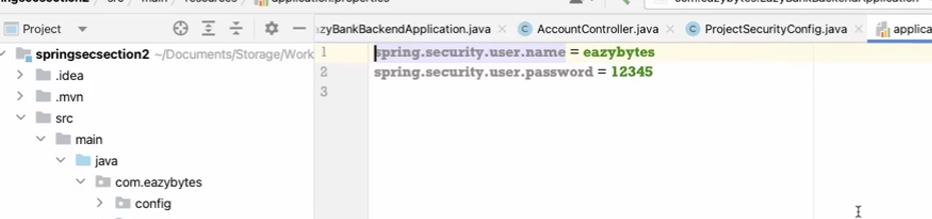
In the above

1. User will send the credentials to the backend server where our application is siting and implemented spring security.
2. As soon as our tomcat receives the request then it is intercepted by the spring security filters to monitor the request. Based on the configuration provided spring security allows the user to access the web page else it enforces authentication. Spring filters also will extract username and password and convert into authentication object as in step 2. Then it is handed over to AuthenticManager.
3. AuthenticManager is class that manages the actual authentication logic. It checks what all Authentication providers inside the web application ie whether validate user from DB / Inmemory / oAuth authentication as in step 4. If any authentication provider is failed then it tries with all the available providers.
4. Authentication providers can be configured and can be multiple providers. Also we can use spring security provided userdetails manager / userdeailsservice. Whenever we are authenticating then its obvious that user details from the DB should be managed and stored somewhere so that we can compare with input. So userdetailsmanager takes care of it.
5. Password encoder If we donot want to handle plain text passwords then password encoder encodes the password and store in the cache. So original password will not be revealed.
6. Next once the user is validated by authentication providers then security filters before sending reponse to the client will store the authentication object information in security context like session id, status hence next time it wont ask credentials.



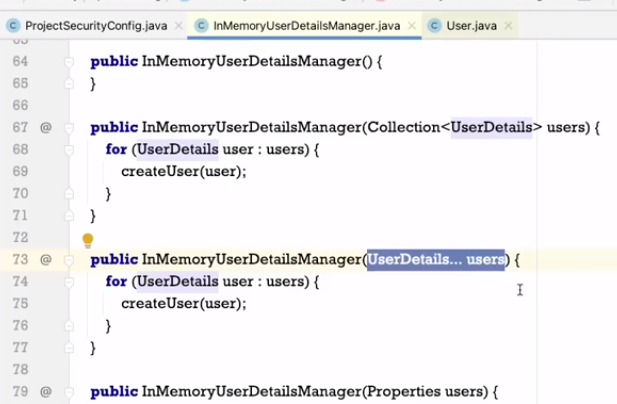
Below is the very basic way of providing security to our web pages it shows the login page for authenticated requests it was done with in memory username and password now will see user from DB authentication

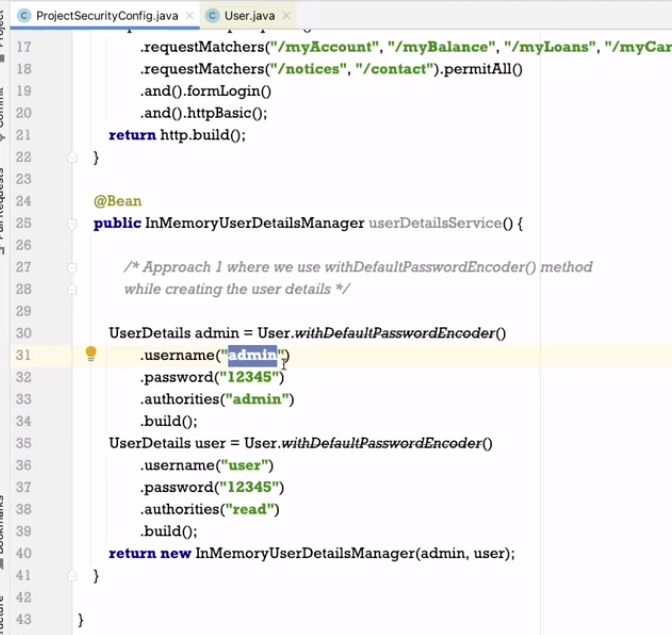
****

****

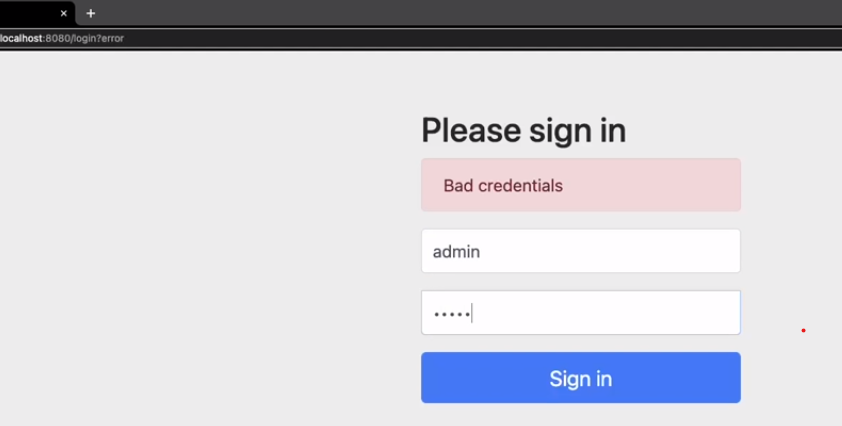
**We can** also define multiple users for in memory authentication but this is not recommended for production

InMemoryUserDetailsManager is a class that implements interface UserDetailsManager. It has constructors that can accept any number of arguments.





User is a class provided by spring security framework will know more about it later when you build the project and run you will see below page add the admin credentials you will be logged in



If we don’t want to perform password encoder then we can go with below approach please check code on github section3

