**DB password Encryption Using JASYPT API**

Let’s understand this through an example use case. We have a spring boot application. In our application passwords and some other fields are written as plain text in the configuration file.The client demands we can’t expose all our password fields **“JUST LIKE THIS”**.

The next question is in fact not just a question, there will be several questions arise in our mind.

How to implement the encryption strategy?

Which framework/library should we use?

What are the complications and learning curve? etc etc.

Well. a good news for you. Remember it’s a spring boot app and how can you forget the starter dependencies?

We have “jasypt encryption library” with spring boot starter dependency. Just add the below dependency to your pom.xml file and do a maven update so that it will download the required binaries.

<dependency>

<groupId>com.github.ulisesbocchio</groupId>

<artifactId>jasypt-spring-boot-starter</artifactId> <version>1.7</version>

</dependency>

Yes, you heard it right. We are going to use jasypt as  the underlying library to encrypt our plain text password fields. Okay so far so good.

Lets see how jasypt encryption works. Let’s see we have key with the name “spring.datasource.password” in our application.properties file with the value “root123”.

Ex:- spring.datasource.password=root123

To encrypt the above field first of all we need to have a master password and the value of the field. Using them we will generate a encryption password.

java -cp jasypt-1.9.2.jar org.jasypt.intf.cli.JasyptPBEStringEncryptionCLI input=root123 password=jasyptPassword



Once we get the encrypted password and the master key we can put them in the properties file.

Note that in the properties file you have to enclose the encrypted password with ENC(..). Also we need to pass the master password in the properties file.  So for our case our properties file looks like:-

spring.datasource.password=ENC(RlVrVl5FnDsIs/PlpbVfHg==)

jasypt.encryptor.password=jasyptPassword

The next step is to annotate the main class with @EnableEncryptableProperties annotation. Now we are done with all the configurations.

We can start our spring boot application and if we have logs enabled then in the logs we can see the decrypted password value. Note that we don’t have to write any special decryption strategy here. Only thing is we are providing the master password. During run time the master password is used internally to decrypt the values of the fields which are enclosed with ENC()

In some cases, putting the master password in the properties file is also not a good approach. We can pass it from the command line while running the fat jar or set to the environment.

Apart from application.properties file you can also put the encrypted password in any other project specific properties file.