**Encryption of Application Properties**

The encryption of application properties was done for the MongoDB demo project itself. We used Jasypt Encryption Algorithm

Jasypt (Java Simplified Encryption) is a Java library that provides simplified encryption and decryption of data. It aims to make it easy to integrate encryption capabilities into Java applications. Jasypt supports various encryption algorithms, including symmetric and asymmetric encryption.

**Pluggin used:**

<plugin>

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

<artifactId>jasypt-maven-plugin</artifactId>

<version>3.0.5</version>

</plugin>

We used the “superkey” as the key for the encryption. The ENC () is refers to encrypted value of the original parameter.

**Application Properties file.**

server.port=8089

jasypt.encryptor.algorithm=PBEWithMD5AndDES

jasypt.encryptor.password=superkey

jasypt. encryptor.iv-generator-classname=org.jasypt.iv.NoIvGenerator

dbname=ENC (49gWRWCkN/L8GxT3/YFCZQQ3f5L+ndOo)

my.param=ENC(gq5Uz6lAzAsN53aGdKNQPQ==)

spring.data.mongodb.uri=mongodb://localhost:27017/${dbname}

Controller class:

The value of my param which is defined in the application properties file is fetched through @Value and stored in secretparam.

The StandardPBEStringEncryptor in Jasypt is a class designed for password-based encryption of strings. It's part of the Jasypt library and is commonly used to encrypt and decrypt sensitive information like passwords, API keys, or other configuration settings in Java applications.

The password which should be encrypted is passed through the requestHeader for /testEncrypt endpoint. We will get the encrypted value of the password in the console. The encrypted value is passed in the ENC () in the application properties file. The application runs successfully.

To decrypt the password again, pass the encrypted value in the ENC () in the application properties file. Hit the /testDecrypt endpoint, you will get the given password.

@Value ("${my. param}")

private String secretparam;

@GetMapping("/testEncrypt")

private void test2(@RequestHeader (required = false) String string) {

StandardPBEStringEncryptor se = new StandardPBEStringEncryptor();

se.setPassword("superkey");

String str = se.encrypt(string);

System.out.println(str);

}

@GetMapping("/testDecrypt")

private String test3()

{

return secretparam;

}

**Encryption of Properties using AES Encryption Algorithm**

**Pluggin used:**

<plugin>

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

<artifactId>jasypt-maven-plugin</artifactId>

<version>3.0.5</version>

</plugin>

Controller class

The value of my param which is defined in the application properties file is fetched through @Value and stored in secretparam.

AES256TextEncryptor is a symmetric encryption algorithm widely used to secure sensitive data.

The password which should be encrypted is passed through the requestHeader for /testEncrypt endpoint. We will get the encrypted value of the password in the console. The encrypted value is passed in the ENC () in the application properties file. The application runs successfully.

To again decrypt the password, pass the encrypted value in the ENC () in the application properties file. Hit the /testDecrypt endpoint, you will get the given password.

Controller Class

@Value ("${my. param}")

private String secretparam;

@GetMapping("/testAesEncrypt")

private void test2(@RequestHeader (required = false) String string) {

AES256TextEncryptor aes = new AES256TextEncryptor ();

aes.setPassword("superkey");

String str = aes.encrypt(string);

System.out.println(str);

}

@GetMapping("/testAesDecrypt")

private String test3()

{

return secretparam;

}

**Application Properties file**

server.port=8089

#jasypt. encryptor.algorithm=PBEWithMD5AndDES

jasypt.encryptor.password=superkey

#jasypt. encryptor.iv-generator-classname=org.jasypt.iv.NoIvGenerator

my.param=ENC(6GBFJrwG4XsmowflGmxe64UXWDjERbW548CvtthuiUBKhjsa7zZHK/rLVo0Mm/aX)

spring.data.mongodb.uri=mongodb://localhost:27017/employeedb