



Encrypt data in Application properties in Spring Boot

Declare dependencies for Jasypt Spring Boot and Jasypt Maven plugin

In order to use Jasypt library for a Spring Boot application, you need to declare the following dependency in the project's `pom.xml` file:

```
1<dependency>
2  <groupId>com.github.ulisesbocchio</groupId>
3  <artifactId>jasypt-spring-boot-starter</artifactId>
4  <version>3.0.3</version>
5</dependency>
```

This will add some JAR files to the project's classpath, which help Jasypt to decrypt the encrypted values in the application configuration file transparently.

Then you also need to declare Jasypt Maven plugin as follows:

```
1<plugin>
2  <groupId>com.github.ulisesbocchio</groupId>
3  <artifactId>jasypt-maven-plugin</artifactId>
4  <version>3.0.3</version>
5 </plugin>
```

This plugin is important as it allows use to use Maven commands for encryption and decryption, as described in the following sections.

Add `@EnableEncryptableProperties` in the main class

3. Encrypt and Decrypt a single String value

Open a new command prompt window. Change the current directory to the project directory where the `pom.xml` file is in. And type the following command:

```
1mvn jasypt:encrypt-value -Djasypt.encryptor.password=care4edu-
  Djasypt.plugin.value=n@mHm2020
```

This will run Jasypt Maven plugin to encrypt the string `n@mHm2020` using the default encryption configuration with the private key `care4edu`. In the output, you would see it prints something like this:

```
1ENC (MBTWfX8gqMevQe5CKW0pToMbajnpJk0zlb3yoooiSWPjkfYrE8TFNF6vDEMXTu/j)
```

Here, the encrypted value is wrapped inside `ENC ()`, then you can use replace a password in the configuration file by this value.

If you run the above command again, you will see a different encrypted value because the default encryptor uses a random generator. That means a string can be different encrypted value though the private key is the same.

The default encrypt algorithm is bidirectional, which means you can do decryption. Type the following command:

```
1mvn jasypt:decrypt-value -Djasypt.encryptor.password=care4edu-
  Djasypt.plugin.value=MBTWfX8gqMevQe5CKW0pToMbajnpJk0zlb3yoooiSWPjkfYrE8TF
```



NF6vDEMXTu/j

This will decrypt the specified value using the default encryption configuration with the private key cafe21. Then you would see it prints the original value n@mHm2020.

So these encrypt and decrypt commands are the very basic ones you should be familiar with.

4. Encrypt credentials in application.properties file

Suppose that you want to encrypt username and password of a Spring data source in the following `application.properties` file:

```
1spring.jpa.hibernate.ddl-auto=none
2spring.datasource.url=jdbc:mysql://localhost:3306/c4e
3spring.datasource.username=root
4spring.datasource.password=password
```

First, wrap the values of username and password inside `DEC ()` as shown below:

```
1spring.jpa.hibernate.ddl-auto=none
2spring.datasource.url=jdbc:mysql://localhost:3306/c4e
3spring.datasource.username=DEC (root)
4spring.datasource.password=DEC (password)
```

Here, `DEC ()` is a placeholder that tells Jasypt what to encrypt, and the remaining values are untouched.

And in the command prompt, type:

```
1mvn jasypt:encrypt -Djasypt.encryptor.password=care4edu
```

Then it will replace the `DEC ()` placeholders in the `application.properties` file with the encrypted value:

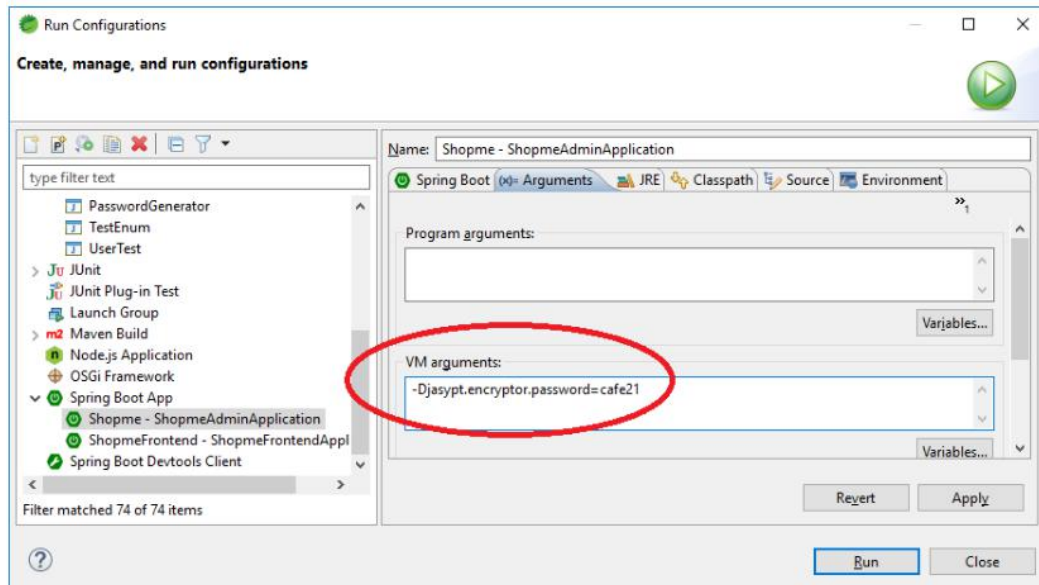
```
1spring.jpa.hibernate.ddl-auto=none
2spring.datasource.url=jdbc:mysql://localhost:3306/c4e
3spring.datasource.username=ENC (9t11aMX4Ije8n0+IcjyS...)
4spring.datasource.password=ENC (IQi6U2g7sz4pw6wL4GoY...)
```

5. Run a Spring Boot application with Jasypt

Now, to run the Spring Boot application you need to pass the private key password as VM arguments in the command prompt like this:

```
1java -Djasypt.encryptor.password=care4edu-jar yourapp.jar
```

To run the Spring Boot application in Eclipse or Spring Tool Suite IDE, you need to edit the run configuration by passing a VM argument like this:



Start the application, and it will run smoothly as Jasypt decrypts the encrypted credentials transparently.

6. Decrypt credentials in Spring application configuration file

In case you want to see the original values of encrypted ones in the Spring Boot configuration file, type the following Maven command:

```
1mvn jasypt:decrypt -Djasypt.encryptor.password=care4edu
```

Jasypt will print content of the `application.properties` file in the output, as it was before encryption. So this command would be useful for checking and verification purpose. Note that it doesn't update the configuration file.

7. Encrypt credentials in application.yml file

```
mvn jasypt:encrypt -Djasypt.encryptor.password=care4edu
```

By default, Jasypt will update the `application.properties` file. In case you're using `application.yml` in your project, specify the path of the file in the command like this:

```
1mvn jasypt:encrypt -Djasypt.encryptor.password=care4edu  
-Djasypt.plugin.path="file:src/main/resources/application.yml"
```

Using this syntax, you can encrypt credentials in any properties file you wish to.


8. Re-encryption with new encryption password

If you want to change the encryptor's private key (password), simply use this command:

```
1mvn jasypt:reencrypt -Djasypt.plugin.old.password=care4edu  
-Djasypt.encryptor.password=shikshakpro
```



CARE 4 EDU
Solutions Pvt Ltd.

+91 9535352376 
info@care4edu.com 
www.care4edu.com 

Then Jasypt Maven plugin will replace the values encrypted with the old password `care4edu` with the new ones encrypted with the new password `shikshakpro` – and you get the configuration file updated instantly. Very convenient.