Spring Framework
- Java 기반 암호화 처리

CONTENTS

- Java 기반 암호화
- Spring Security를 이용한 암호화
- Mysql 함수

암호화 관련 용어

- **암호(Cryptography)**
  : 해독 불가능한 형태로 변환하거나 또는 암호화된 메시지를 해독 가능한 형태로 변환하는 기술
- **평문(Plaintext)**
  : 해독 가능한 형태의 메시지
- **암호문(Ciphertext)**
  : 해독 불가능한 형태의 메시지
- **암호화(Encryption)**
  : 평문을 암호문으로 변환하는 과정
- **복호화(Decryption)**
  : 암호문을 평문으로 변환하는 과정
- **대칭키 암호(또는 비밀키 암호)**
  : 암호화키와 복호화키가 같은 암호
- **비대칭키 암호(또는 공개키 암호)**
  : 암호화키와 복호화키가 다른 암호

- **단방향(일방향) 암호화 알고리즘**

  단방향 암호화는 평문을 암호화했을 때 다시 평문으로 되돌리는 것(복호화)을 할 수 없는 암호화.

  많이 사용하고 있는 알고리즘은 SHA-256.

  패스워드는 평문으로 만들 필요가 없는 정보로 분류해서 SHA-256 을 주로 사용합니다.

  - **SHA-256으로 해시**

    MessageDigest객체 생성시 알고리즘을 "SHA-256"으로 해서 생성하고, 해시된 데이터는 바이트 배열의 바이너리 데이터이므로 16진수 문자열로 변환해 준다.

    이 알고리즘을 이용해서 암호문으로 관리.

    복호화가 없으므로 별도의 키가 없다.

    키가 없으므로 SHA256 알고리즘을 돌려서는 항상 같은 결과가 나오게 된다.

    1과 같은 단순한 값을 sha256으로 암호화했을 경우 항상 같은 값이 나오기 때문에 비밀번호 입력을 적당한 길이와 복잡성을 가지도록 유도하여 패스 워드를 유추하는데 어렵도록 해야 한다. (대문자 + 소문자 + 특수문자 + 8자리 이상)

```
import java.security.MessageDigest;
public class Sha256 {
        public static String encrypt(String planText) {
                try {
                        MessageDigest md = MessageDigest.getInstance("SHA-256");
                        md.update(planText.getBytes());
                        byte byteData[] = md.digest();
                        StringBuffer sb = new StringBuffer();
                        for (int i = 0; i < byteData.length; i++) {
                                sb.append(Integer.toString((byteData[i] & 0xff) + 0x100,
                                                                16).substring(1));
                        }
                        StringBuffer hexString = new StringBuffer();
                        for (int i = 0; i < byteData.length; i++) {
                                String hex = Integer.toHexString(0xff & byteData[i]);
                                if (hex.length() == 1) {
                                        hexString.append('0');
                                }
                                hexString.append(hex);
                        }
                        return hexString.toString();
                } catch (Exception e) {
                        e.printStackTrace();
                        throw new RuntimeException();
                }
        }
}
```

- **양방향 알고리즘**

  양방향 암호화 알고리즘은 평문에서 암호문으로, 암호문에서 평문으로 변환하는 암호화와 복호화가 이루어지는 알고리즘이다.

  많이 사용하는 알고리즘은 AES-256.

  주로 이름, 주소, 연락처 등 복호화 하는데 필요한 정보를 이 알고리즘을 이용해서 암호문으로 관리.

  - AES 256(256비트 블록암호화) 으로 암호화
    - AES-256 는 org.apache.commons.codec 라이브러리에 의존성을 가지고 있어MAVEN 의존성 주입
    - AES256Util 객체를 생성할 때에는 암호화/복호화에 사용될 키를 입력
    - 키의 길이가 16자리 이하일 경우 오류가 발생
    - local_policy.jar 파일과 US_export_policy.jar 추가 다운로드한 라이브러리가 없을 경우에는 16자리의 키를 입력하더라고 java.security.InvalidKeyException: Illegal key size이 발생
    - 블록암호화를 진행하기 위해서는 패딩기법이 필요
    - 데이터를 특정크기로 맞추기 위해서, 특정크기보다 부족한 부분의 공간을 의미 없는 문자들로 채워서 비트 수를 맞추는 것. 암호화 시에는 반드시 필요한 방법.

    Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");

```java
import java.io.UnsupportedEncodingException;
import java.security.GeneralSecurityException;
import java.security.Key;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;
public class AES256Util {
	private String iv;
	private Key keySpec;
	/**
	 * 16자리의 키값을 입력하여 객체를 생성한다.
	 * @param key 암/복호화를 위한 키값
	 * @throws UnsupportedEncodingException 키값의 길이가 16이하일 경우 발생
	 */
	public AES256Util(String key) throws UnsupportedEncodingException {
		this.iv = key.substring(0, 16);
		byte[] keyBytes = new byte[16];
		byte[] b = key.getBytes("UTF-8");
		int len = b.length;
		if (len > keyBytes.length) {
			len = keyBytes.length;
		}
		System.arraycopy(b, 0, keyBytes, 0, len);
		SecretKeySpec keySpec = new SecretKeySpec(keyBytes, "AES"); // 공통 키 생성
		this.keySpec = keySpec;
	}
```

```java
/**
 * AES256 으로 암호화 한다.
 *
 * @param str 암호화할 문자열
 * @return
 * @throws NoSuchAlgorithmException
 * @throws GeneralSecurityException
 * @throws UnsupportedEncodingException
 */
public String encrypt(String str)
                        throws NoSuchAlgorithmException, GeneralSecurityException,
                               UnsupportedEncodingException {
            Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding"); // 암호화 패딩 기법 설정
            c.init(Cipher.ENCRYPT_MODE, keySpec, new IvParameterSpec(iv.getBytes()));
            byte[] encrypted = c.doFinal(str.getBytes("UTF-8"));
            String enStr = new String(Base64.encodeBase64(encrypted));
            return enStr;
}
```

```java
    /**
     * AES256으로 암호화된 txt 를 복호화한다.
     *
     * @param str 복호화할 문자열
     * @return
     * @throws NoSuchAlgorithmException
     * @throws GeneralSecurityException
     * @throws UnsupportedEncodingException
     */
    public String decrypt(String str)
                                throws NoSuchAlgorithmException, GeneralSecurityException,
                                       UnsupportedEncodingException {
                Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
                c.init(Cipher.DECRYPT_MODE, keySpec, new IvParameterSpec(iv.getBytes()));
                byte[] byteStr = Base64.decodeBase64(str.getBytes());
                return new String(c.doFinal(byteStr), "UTF-8");
    }
}
```

그 외의 알고리즘

- 한국인터넷진흥원 암호화이용활성화 사이트에서 자체적으로 개발한 암호 기능들을 다운로드 받을 수 있었다.
- KISA 암호이용활성화 : http://seed.kisa.or.kr/

■스프링 시큐리티로 비밀번호 암호화하기

```
<!--스프링시큐리티 web 라이브러리-->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>${org.springframework-version}</version>
  </dependency>
  <!--스프링시큐리티 core 라이브러리-->
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>${org.springframework-version}</version>
  </dependency>
  <!--스프링시큐리티 config 라이브러리-->
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>${org.springframework-version}</version>
</dependency>
```

스프링 시큐리티로 비밀번호 암호화하기

- **spring-security.xml 생성**

  : 해당 xml 문서에는 비밀번호 암호화인코딩을 위한 bean을 추가.

  스프링 시큐리티 라이브러리에서 제공하는 BcryptPasswordEncoder는

  - 비밀번호 암호화 메서드

  - 인코딩된 비밀번호와 Raw 형태의 비밀번호를 비교해주는 메서드를 제공.

  하지만 디코딩하는 메서드는 지원하지 않는다.

```
<beans:bean id="bcryptPasswordEncoder"
class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder" />
```

- web.xml 에 로드시 spring-security.xml 을 읽어오도록 파라미터를 추가

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
     /WEB-INF/spring/root-context.xml
     /WEB-INF/spring/spring-security.xml
  </param-value>
```

■ 스프링 시큐리티로 비밀번호 암호화하기

- BCryptPasswordEncoder 객체 주입 후 암호화 메서드 이용

  passwordEncoder.encode(memberInfo.getPassword());

- BCryptPasswordEncoder 빈의 메서드를 활용하면 로우 데이터와 인코딩 데이터를 비교할 수 있다.

  passwordEncoder.matches(rawPW , memberInfo.getPassword());

Mysql 암호화 함수

- **Mysql에서 제공하는 암호화 함수**

	암호화	복호화
단방향	MD5	-
	PASSWORD, OLD_PASSWORD	
	SHA1, SHA2, SHA	
쌍방향	AES_ENCRYPT	AES_DECRYPT
	DES_ENCRYPT	DES_DECRYPT

Mysql 암호화 함수

- 단방향
  - MD5

    SELECT MD5('컬럼' or '문자열')

  - PASSWORD

    SELECT PASSWORD('컬럼' or '문자열')

  - SHA1

    SELECT SHA1('컬럼' or '문자열')

Mysql 암호화 함수

- **쌍방향**
  - **AES 암호화**

    SELECT HEX(AES_ENCRYPT('컬럼' or '문자열', '암호화키'))

  - **AES 복호화**

    SELECT AES_DECRYPT(UNHEX('컬럼' or '문자열'), '암호화키')

```
insert into private_data values(
        '유영진', AES_ENCRYPT('010-7777-9999', SHA2('key_value', 512)),
        AES_ENCRYPT('서울시 종로구', SHA2('key_value', 512))
);
```

```
select name,
        CONVERT(AES_DECRYPT(phone,SHA2('key_value',512)) using utf8) phone,
        CONVERT(AES_DECRYPT(address,SHA2('key_value',512)) using utf8) address
from member ;
```