Apunte PBKDF2 Java,

con y sin Salt

PBKDF2 es un método estándar para derivar claves, y tanto en OpenSSL como en Java se puede utilizar de manera similar. Aquí dejo una guía de cómo implementar PBKDF2 en Java, haciendo un paralelo con el uso en OpenSSL.

OpenSSL

En OpenSSL, usar PBKDF2 se puede hacer con el siguiente comando:

```
> openssl enc -des3 -pbkdf2 -iter 1000 -salt -in plain.txt -out
key.bin
```

Este comando usa PBKDF2 para derivar una clave a partir de una contraseña en `plain.txt`, usando 1000 iteraciones y un salt.

Java

En Java, se puede usar la clase `SecretKeyFactory` junto con `PBEKeySpec` para implementar PBKDF2. Aquí dejo un ejemplo de cómo hacerlo:

```
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import java.util.Base64;

public class PBKDF2Example {

   public static void main(String[] args) throws NoSuchAlgorithmException,
InvalidKeySpecException {
       String password = "password";
       byte[] salt = "12345678".getBytes(); // Salt de 8 bytes
       int iterations = 10000;
       int keyLength = 256; // Longitud de la clave en bits

       // Crear PBEKeySpec con la contraseña, salt, número de iteraciones y longitud
de la clave
```

```
        PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt, iterations,
keyLength);
        SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
        byte[] hash = skf.generateSecret(spec).getEncoded();

        // Codificar la clave derivada en base64 para una representación más amigable
        String base64Hash = Base64.getEncoder().encodeToString(hash);

        System.out.println("Clave derivada: " + base64Hash);
    }
}
```

Ejecución

Para ejecutar el mismo debes guardar esto en un archivo, compilarlo y ejecutarlo:

```
> javac PBKDF2Example.java
> java PBKDF2Example
```

Sin Salt

Si deseas hacer la derivación de clave sin usar un salt (aunque no es recomendable por razones de seguridad), simplemente puedes pasar un array vacío para el salt en ambas implementaciones.

OpenSSL sin Salt

```
> openssl enc -des3 -pbkdf2 -iter 1000 -nosalt -in plain.txt -out
key.bin
```

Java sin Salt

java

```
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import java.util.Base64;

public class PBKDF2ExampleNoSalt {
```

```
   public static void main(String[] args) throws NoSuchAlgorithmException,
InvalidKeySpecException {
       String password = "password";
       byte[] salt = new byte[] {0}; // Sin salt
       int iterations = 10000;
       int keyLength = 256; // Longitud de la clave en bits

       // Crear PBEKeySpec con la contraseña, salt, número de iteraciones y longitud
de la clave
       PBEKeySpec spec = new PBEKeySpec(password.toCharArray(), salt, iterations,
keyLength);
       SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
       byte[] hash = skf.generateSecret(spec).getEncoded();

       // Codificar la clave derivada en base64 para una representación más amigable
       String base64Hash = Base64.getEncoder().encodeToString(hash);

       System.out.println("Clave derivada: " + base64Hash);
   }
}
```

Ejecución

Para ejecutar el mismo debes guardar esto en un archivo, compilarlo y ejecutarlo:

```
> javac PBKDF2ExampleNoSalt.java
> java PBKDF2ExampleNoSalt
```

Espero que esto te sea útil y puedas ver claramente cómo hacer la correspondencia entre OpenSSL y Java.