Password Encoding

Kriptolu Şifrelerin Kullanılması

- Spring Security **şifreler**in DB'de salt metin değilde, **kriptolu olarak** saklanmasını sağlar
- Şifreleri kriptolamak için **değişik algoritmalar** desteklenir
- Bunlar **tek yönlü** algoritmalardır
- Kimliklendirme sırasında kullanıcının girdiği şifre algoritmaya göre kriptolanarak **DB'deki kriptolu değer** ile karşılaştırılır

Kriptolu Şifrelerin Kullanılması

- Kriptolu şifre desteği aynı zamanda **tuzlama** (**salting**) kabiliyetine de sahiptir
- Tuzlama yöntemi ile **sözlük saldırıları**na (dictionary attacks) karşı tedbir alınabilir
- Tuzlamada kullanılacak **gizli bilgi** sistem genelinde **ortak bir değer** olabilir
- Ya da her bir **kullanıcı** için onun **spesifik bir attribute**'u olabilir

Kriptolu Şifrelerin Kullanılması

- **authentication-provider** elemanı altında tanımlı **password-encoder**'ın hash attribute'u ile kripto algoritması belirtilir

```
<security:authentication-manager>

    <security:authentication-provider
user-service-ref="userService">

        <security:password-encoder hash="sha">
            <security:salt-source system-wide="secret" />
        </security:password-encoder>

    </security:authentication-provider>

</security:authentication-manager>
```

bcrypt, plaintext, sha, sha-256, md4, md5 gibi hashing algoritmaları desteklenmektedir. Spring 4 ile **bcrypt** kullanılması önerilir

Kriptolu Şifrelerin Kullanılması

- Kriptolu şifreleri oluşturmak için seçilen algoritmaya karşılık gelen sınıf aşağıdaki örneklerdeki gibi kullanılabilir

```
ShaPasswordEncoder passwordEncoder = new ShaPasswordEncoder();

String encodedPasswd = passwordEncoder.encodePassword("plain text
passwd","salt value");


BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();

String encodedPasswd = passwordEncoder.encode("plain text passwd");
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com