

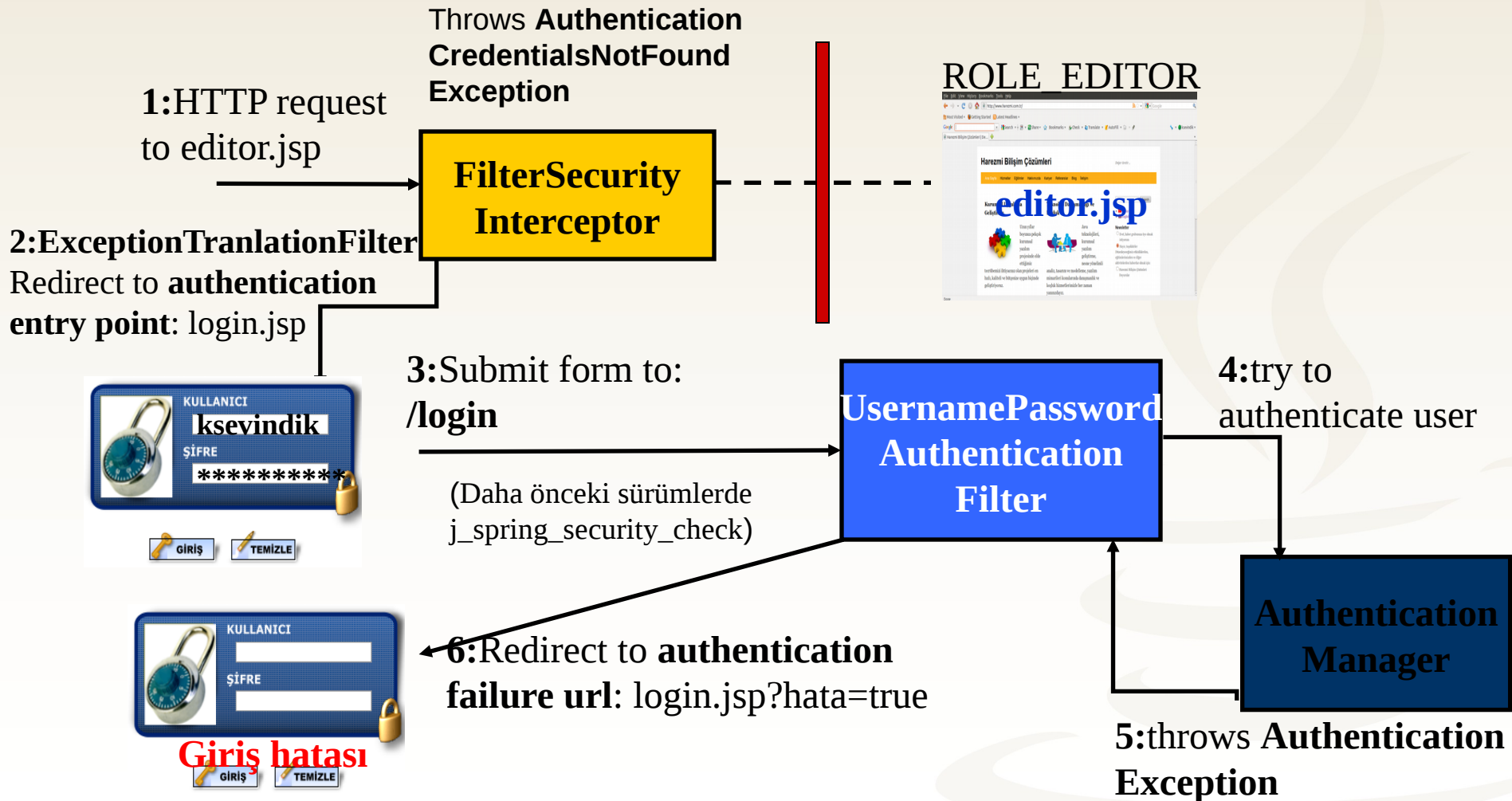
Kimliklendirme (Authentication)



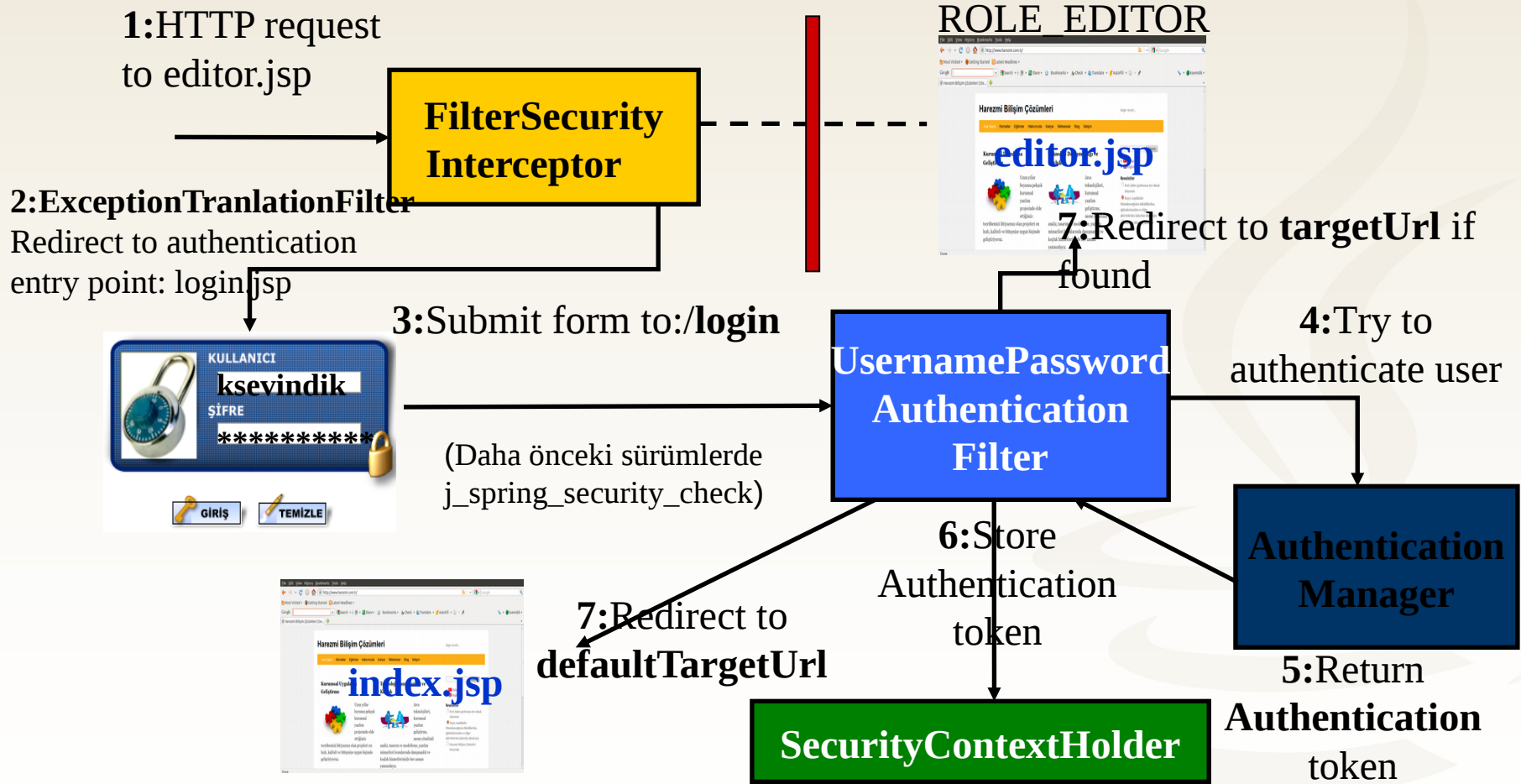
Kimliklendirme Nedir?

- Sistemdeki herhangi bir kaynağa erişmeye çalışan **kullanıcı veya harici sistemin kimliğinin denetlenmesi** işlemidir
- Öncelikle kullanıcı veya harici sistem belirttiği kimlik bilgisini kanıtlayan **gizli veya benzersiz bir bilgiyi** sunucuya iletir
- Sunucu bu bilgiyi kullanarak kullanıcının **kimlik bilgisini denetler**
- İşlem başarılı ise kullanıcı **sisteme giriş** yapabilir

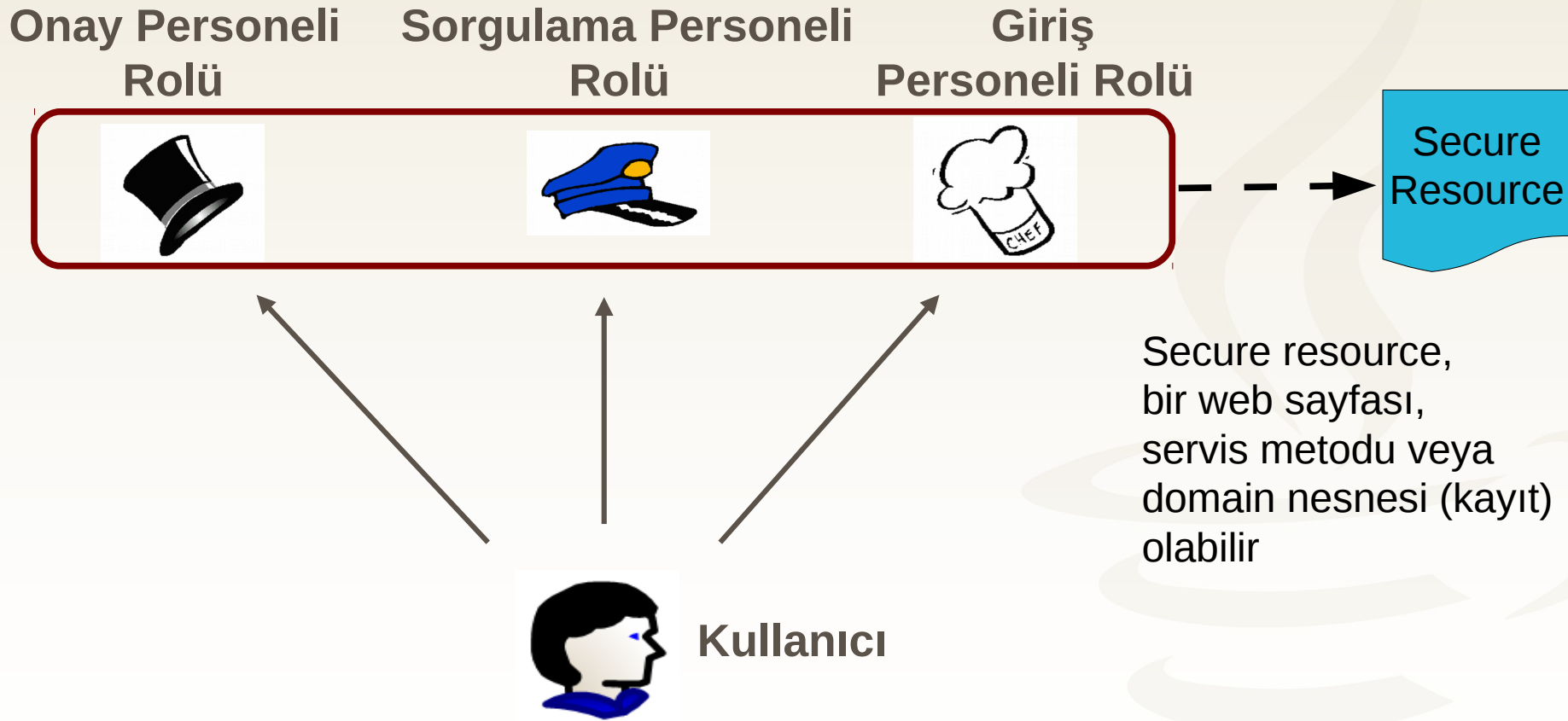
Başarısız Bir Login Akış Örneği



Başarılı Bir Login Akış Örneği



Spring Security Kullanıcı – Rol İlişkisi



Spring Security

Rol Grubu – Rol İlişkisi

Onay Personeli
Rolü



Sorgulama Personeli
Rolü



Giriş
Personeli Rolü



Sistem Admin
Rolü



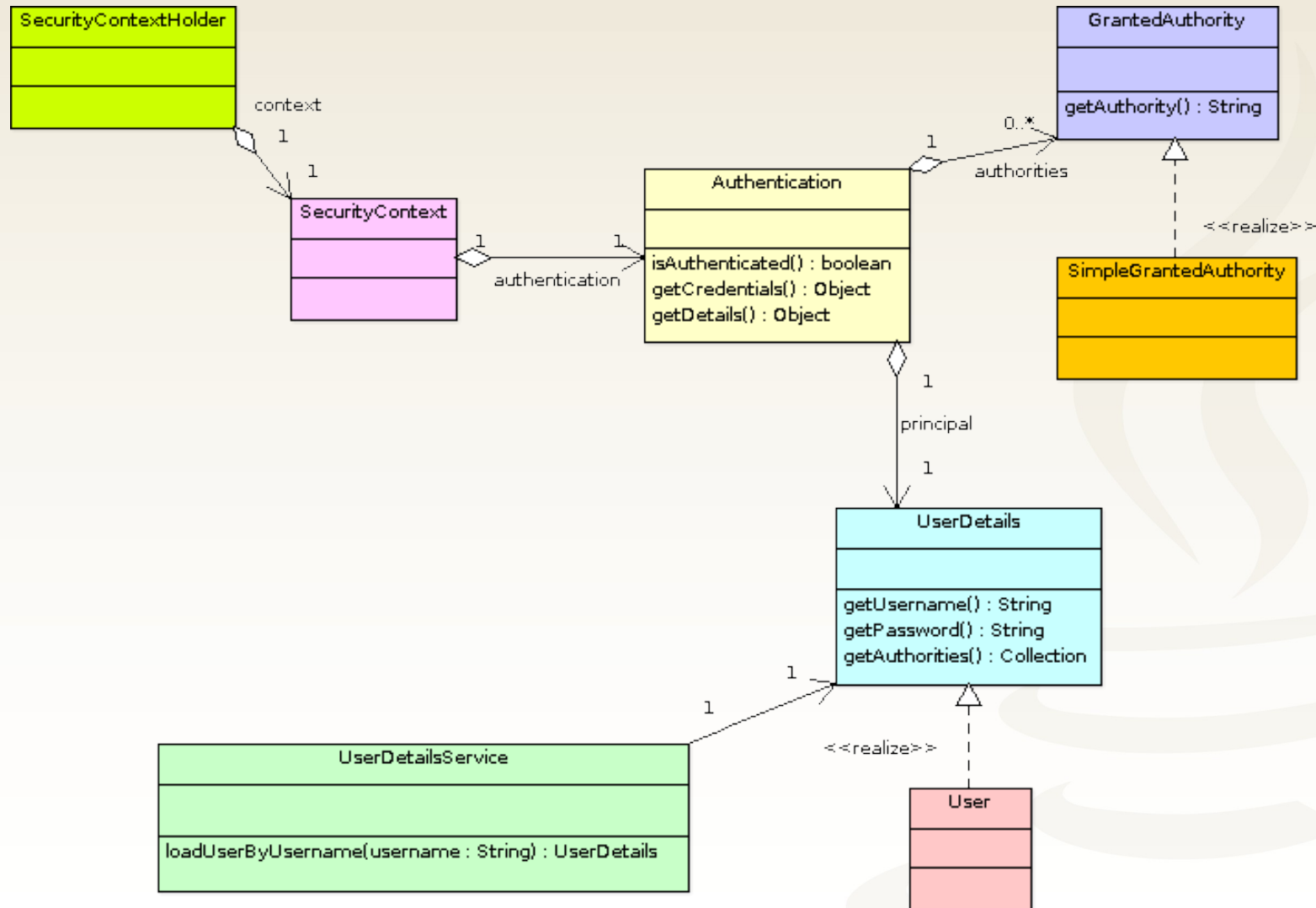
Kullanıcı

Rol grubu, birden fazla rolün
bir araya getirilip bir kullanıcıya
topluca atanmasını sağlar

Spring Security Domain Sınıfları

- Authentication
- UserDetails
- User
- GrantedAuthority
- SimpleGrantedAuthority
- UserDetailsService
- SecurityContext
- SecurityContextHolder

Spring Security Domain Sınıfları



Kimlik Bilgisinin İki Web Requesti Arasında Saklanması

- Web uygulamalarında principal bilgisi genellikle request'ler arasında **HttpSession** içerisinde saklanır
- **SecurityContextPersistenceFilter** bu işi yapar
- Request başlangıcında **SecurityContext** HttpSession'dan alınır ve **SecurityContextHolder**'a set edilir
- Request sonlandığında da **SecurityContextHolder** **temizlenir**

Kimlik Bilgisinin İki Web Requesti Arasında Saklanması

- HttpSession kullanmayan uygulama servislerinde (REST gibi) **her request'de kimliklendirme** yapılması gerekir
- Her **request sonunda SecurityContextHolder'in temizlenmesini** sağladığı için RESTful senaryolarda bile bu Filter önemlidir

Kimliklendirme Bilgilerine Erişim

- **SecurityContextHolder** üzerinden isteğimiz herhangi bir yerden **Authentication** bilgilerine erişebiliriz

```
SecurityContext securityContext = SecurityContextHolder.getContext();
```

```
UsernamePasswordAuthenticationToken authenticationToken =  
(UsernamePasswordAuthenticationToken) securityContext.getAuthentication();
```

```
String username = authenticationToken.getName();
```

```
User principal = (User) authenticationToken.getPrincipal();
```

```
Collection<GrantedAuthority> authorities =  
    authenticationToken.getAuthorities();
```

```
boolean authenticated = authenticationToken.isAuthenticated();
```

Login Sayfasının Özelleştirilmesi

Spring 4 öncesi login processing url
j_spring_security_check idi

Beni hatırla kabiliyeti
aktif ise gereklidir

```
<form action="login" method="post">  
  Username:<input name="username" type="text"/><br/>  
  Password:<input name="password" type="password"/><br/>  
  Remember:<input name="remember-me" type="checkbox">  
  
  <input type="hidden" name="${_csrf.parameterName}"  
value="${_csrf.token}">  
  
  <input type="submit" value="Login">  
</form>
```

Spring Security 4 ile birlikte csrf koruması default olarak aktif gelmeye başladı. Dolayısı ile CSRF aktif ise login sayfası içerisinde de csrf token'ın yönetilmesi için bir gizli input alanın eklenmesi gerekir

Spring 4 öncesi username parametresi j_username, password parametresi j_password, remember-me ise _spring_security_remember_me idi

Login Sayfasının Özelleştirilmesi

```
<security:http>
```

```
<security:form-login login-page="/login.jsp"
  authentication-failure-url="/login.jsp?error=true"/>
```

```
<security:intercept-url pattern="/login.jsp"
access="permitAll"/>
```

```
<security:intercept-url pattern="/**"
access="hasRole('ROLE_USER')"/>
```

```
</security:http>
```


Anonim kimliklendirme yerine **permitAll** ifadesi de kullanılabilir

/login.jsp request URL'inin intercept edilmemesi gerekir. Aksi takdirde login.jsp sayfasında yetkilendirmeye tabi tutulacağı için sayfa düzgün biçimde render edilemeyecek ve **"The page isn't redirecting properly"** şeklinde bir hata alınacaktır.

Login Sayfasının Özelleştirilmesi

```
<beans...>
```

```
<security:http pattern="/login.jsp" security="none"/>
```



csrf koruma
aktif ise security none
kullanımı
uygun değildir!

Bu kullanım şekli daha
çok Spring Security 4
öncesi dönemde karşımıza
çıkmaktadır

```
<security:http>
```

```
<security:form-login login-page="/login.jsp"  
authentication-failure-url="/login.jsp?error=true"/>
```

```
<security:intercept-url pattern="/**"  
access="hasRole('ROLE_USER')" />
```

```
</security:http>  
</beans>
```

İletişim

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

