RunAs Kabiliyeti ile Geçici Yetkilendirme

Run As Kabiliyeti

- Spring Security'nin **run as kabiliyeti** ile **Authentication** nesnesinin **GrantedAuthority** nesneleri arasına **geçici olarak ilave roller** eklenebilir
- **Geçici olarak bir takım rollerin atanması** bazı rollerin sürekli olarak kullanıcıda atanmış olmasının sakıncalı olabileceği durumlar için uygundur

Run As Kabiliyeti

- Kullanıcılar geçici olarak belirli rollere sahip olurlar ve ilgili metotları invoke edebilirler
- Metot çağrısı sonrasında da **atanan rol** kullanıcıdan **geri alınır**
- RunAsManager, **ConfigAttribute**'ler arasında **RUN_AS** ile başlayan bir değer tespit ederse **RunAsUserToken** oluşturur
- **RunAsUserToken** mevcut Authentication bilgileri ile oluşturulur

Run As Kabiliyeti

- **RUN_AS** ile başayan role **ROLE_ prefix**'i eklenerek mevcut rollerle birlikte RunAsUserToken'a eklenir
- Metot çağrısı sonunda RunAsUserToken **discard** edilir
- Güvenlik nedeni ile RunAsUserToken yaratıldıktan sonra **immutable**'dır

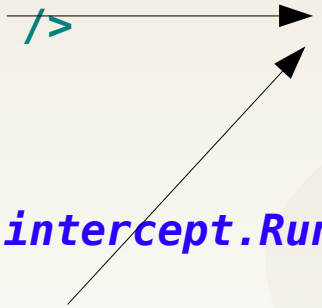
Run As Konfigürasyonu

```
<bean id="runAsManager"
class="org.springframework.security.access.intercept.RunAsManagerImpl">
    <property name="key" value="secret_key" />
</bean>

<bean id="runAsAuthenticationProvider"
class="org.springframework.security.access.intercept.RunAsImplAuthenticationProvider">
    <property name="key" value="secret_key" />
</bean>

<security:global-method-security secured-annotations="enabled"
run-as-manager-ref="runAsManager" />

<security:authentication-manager>
    <security:authentication-provider
ref="runAsAuthenticationProvider"/>
    <!-- diger authentication provider'lar -->
</security:authentication-manager>
```



Run As Konfigürasyonu

- **RunAsAuthenticationProvider** SecurityContext içerisinde gördüğü **RunAsUserToken**'ı validate eder
- Burada önemli nokta **RunAsUserToken** içindeki key değerinin **RunAsAuthenticationProvider**'ın key değeri ile eşleşmesidir

Run As Kullanımı

```
@Service
public class UserService {
 @Secured({"IS_AUTHENTICATED_FULLY","RUN_AS_AUDITOR"})
 public User findUser(String username) {
 //...
 }
}
```

- Yukarıdaki metot **kimliklendirilmiş herhangi bir kullanıcı** tarafından invoke edilebilecektir
- findUser() metot çağrısı boyunca da kullanıcı **ilaveten ROLE_RUN_AS_AUDITOR rolü**ne de sahip olacaktır

Run As Kullanımı

- Run As built-in olarak **sadece @Secured** anotasyonu ile kullanılabilir
- **@PreAuthorize** anotasyonunun kullanıldığı metot üzerinde aynı zamanda @Secured ile run as rolünü belirtmek mümkün değildir
- @PreAuthorize ile run as'i birlikte kullanabilmek için **custom bir RunAsManager implemantasyonu** yapmak gerekir

İletişim

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