

Spring Security ve Request URL



ContextPath, ServletPath ve PathInfo Bilgileri

- Servlet spesifikasyonu **HttpServletRequest** üzerinden erişilebilecek **bir takım property'ler** tanımlamıştır
 - **ContextPath**: Request URI'ında uygulamanın sunulduğu dizine karşılık gelen kısımdır
 - **ServletPath**: Servlet'e erişim sağlayan kısımdır
 - **PathInfo**: Servlet'e erişim sağlayan kısımdan sonra queryString'e kadar olan bölümdür
 - **QueryString**: ? işareti ile başlayan bölümdür

ContextPath, ServletPath ve PathInfo Bilgileri

- Yalnız ServletPath ve PathInfo'nun **tam olarak ne döneceği** spesifikasyonda detaylandırılmamıştır
- Eğer servlet “/” şeklinde map edilmiş ise
 - **URI:**http://localhost/petclinic/x/y/z?a=1&b=2
 - **ContextPath:**/petclinic
 - **PathInfo:**NULL
 - **ServletPath:**/x/y/z
 - **QueryString:**a=1&b=2

ContextPath, ServletPath ve PathInfo Bilgileri

- Eğer servlet “/*” şeklinde map edilmiş ise
 - **URI:** `http://localhost/petclinic/x/y/z?a=1&b=2`
 - **ContextPath:** `/petclinic`
 - **PathInfo:** `/x/y/z`
 - **ServletPath:** `""` (boş String)
 - **QueryString:** `a=1&b=2`

ContextPath, ServletPath ve PathInfo Bilgileri

- Eğer servlet “/xyz” şeklinde map edilmiş ise
 - **URI:**http://localhost/petclinic/xyz?a=1&b=2
 - **ContextPath:**/petclinic
 - **PathInfo:**NULL
 - **ServletPath:**/xyz
 - **QueryString:**a=1&b=2

ContextPath, ServletPath ve PathInfo Bilgileri

- Eğer servlet “/xyz/*” şeklinde map edilmiş ise
 - **URI:**http://localhost/petclinic/xyz/qqq?a=1&b=2
 - **ContextPath:**/petclinic
 - **PathInfo:**/qqq
 - **ServletPath:**/xyz
 - **QueryString:**a=1&b=2

URL Path Parametreleri

- Ayrıca URL'deki her bir **path segment**'i bir takım parametreler de tanımlayabilir
- Örneğin
`http://localhost/petclinic/x;param1=1/y;param2=2/z?a=1&b=2`
- Param1 ve param2 **path segment parametreleridir**
- Servlet Container'ların bu path parametrelerini nasıl ele alacağı da **net değildir**

URL Path Parametreleri

- Bazı container'lar bunları **normalize ederler**, bazıları ise hiç **dokunmazlar**
- Ayrıca request URI'ında ./,/../,// gibi path traversal ifadeleri de olabilir
- Bu tür path segment parametreleri ve path traversal ifadeleri request pattern eşleşmelerinin **başarısız olmasına** neden olabilir

Spring Security ve Path Bilgileri ile Eşleştirme

- Spring Security **sadece uygulama içerisindeki path bilgileri** üzerinden yetkilendirme yapar
- Dolayısı ile **contextPath** ile ilgilenmez
- Default durumda **servletPath** ve **pathInfo** bilgilerini birleştirerek ant stili path'ler üzerinden eşleştirme yapar
- **Query string** de göz ardı edilir
- Eşleştirme **case insensitive** gerçekleşir

AntPathRequestMatcher

- Ant stili path eşleştirmede kullanılan **RequestMatcher** implementasyonudur

Pattern	Eşleşen Örnek URI
<code>/** veya **</code>	Universal matcher, bütün request'leri yakalar
<code>/a/**</code>	<code>/a,/a</code> veya <code>/a/b/c</code>
<code>/a/t?st.jsp</code>	<code>/a/test.jsp</code> veya <code>/a/tast.jsp</code>
<code>/a/*.jsp</code>	<code>/a</code> dizini altındaki <code>.jsp</code> uzantılı bütün dosyaları yakalar
<code>/a/**/test.jsp</code>	<code>/a/test.jsp</code> , <code>/a/b/test.jsp</code> veya <code>/a/b/c/test.jsp</code>

RegexRequestMatcher

- Eğer daha komplike eşleme ihtiyacı varsa **regular expression** üzerinden çalışan implemantasyon kullanılabilir
- Burada **query string de dikkate alınır**
- Ayrıca eşleştirme **case sensitive veya insensitive** gerçekleştirilebilir

```
<security:http auto-config="true" request-matcher="regex">
    <security:intercept-url pattern="/.*" access="isFullyAuthenticated()" />
</security:http>
```

Alabileceği değerler:

ant,
regex
ciRegex

Spring Security

HttpFirewall

- Spring Security **FilterChainProxy** **HttpFirewall** arayüzü vasıtası ile current request'i kontrol eder ve sarmalar
- Böylece **normalize olmamış istekleri** (./,/../,/. ifadelerini içeren) red eder, **path parametreleri, forward slash'ları** ayıklayabilir
- Sonuç olarak pattern eşleşmesini olumsuz etkileyecek kısımlardan **request'i arındırır**

Spring Security

HttpFirewall

- Çoğu uygulama için **DefaultHttpFirewall** konfigürasyonu yeterlidir
- İstenirse FilterChainProxy içerisinde **uygulamaya özel başka bir HttpFirewall implemantasyonunun** kullanılması da sağlanabilir

```
<beans...>  
    <security:http-firewall ref="customHttpFirewall"/>  
</beans>
```

İletişim

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

