6.5. Approval

Table of contents

- Overview
  - Access authorization (request URL)
  - Access authorization (JSP)
  - Access authorization (Method)
- How to use
  - Access authorization (request URL)
    - `: <Sec intercept-url>` set of elements
    - Configuring URL that does not perform access authorization control
    - Exception handling in the URL pattern
  - Access authorization (JSP)
  - Access authorization (Method)
- How to extend
  - Role hierarchy function
    - Common setting
    - Access authorization (request URL), how they are used in access authorization (JSP)
    - How they are used in access authorization (Method)

6.5.1. Overview

In this section, I will describe the authorization features that are offered in Spring Security.

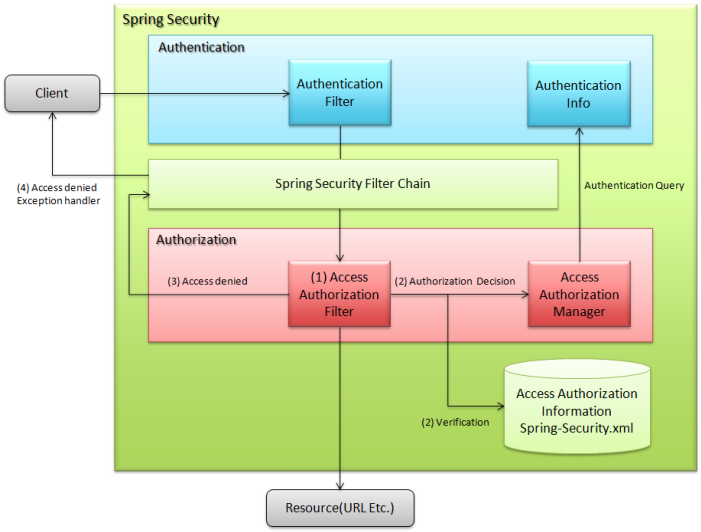
In order to realize by using the access authorization function of Spring Security, it is assumed to use the authentication function of the Spring Security.
For the authentication method using the Spring Security, *authenticated* see.

Subject of resource access authorization, the following three items.

1. Web (request URL)
   - It is possible to set the required permissions to access a specific URL
2. Screen item (JSP)
   - It is possible to set the required permissions to view the specific elements of the screen
3. Method
   - You can set the permissions required to perform a particular method

In Spring Security, describes the access authorization information in the configuration file or annotations, so as to realize the function.

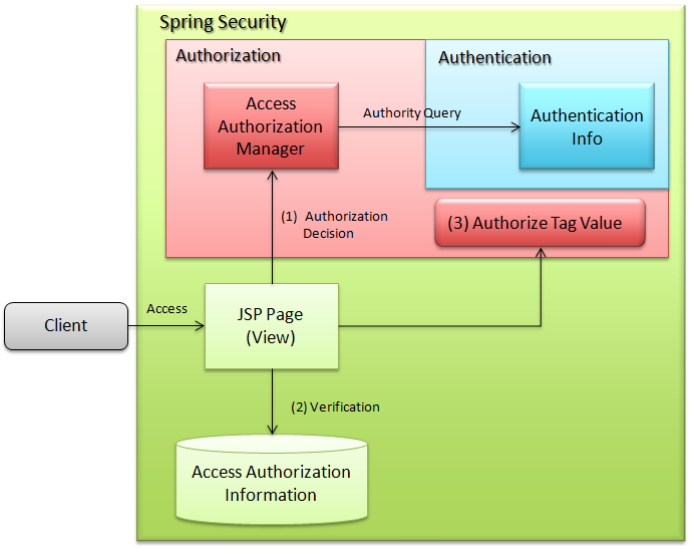
6.5.1.1. Access authorization (request URL)



Picture - Authorization (request URL)

1. To the user of the request, Spring Security filter chain is an interrupt processing.
2. It matches the URL and requests subject to admission control, query the determination of the access authorization the access authorization manager.
3. Access authorization manager, check the permissions and access authorization information of the user, if the required role is not assigned, throw an access denied exception.
4. If the required role has been assigned, I will continue the process.

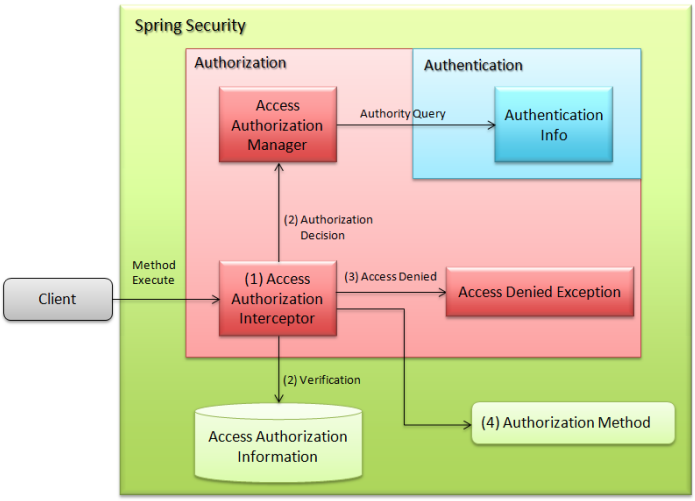
6.5.1.2. Access authorization (JSP)



Picture - Authorization (JSP)

1. Servlet that is generated from the JSP, query the access authorization manager.
2. Access authorization manager checks the privileges and access authorization information of the user, if the required role is not assigned, not to evaluate the internal tag.
3. If the required role has been assigned, I evaluate the internal tags.

6.5.1.3. Access authorization (Method)



Picture - Authorization (Method)

1. Spring container based on the access authorization information, and generates an interceptor for the object of interest, and to interrupt.
2. Interceptor query the access authorization manager based on the role that has been set.
3. Access authorization manager, check the permissions and access authorization information with the user, to throw an access denied exception if the required role has not been assigned.
4. If the required roles have been assigned, continuing the process (by the setting, it is possible to check the authority after performing the process).

6.5.2. How to use

Access authorization (request URL), access authorization (JSP), I will describe how to use the access authorization (Method).

6.5.2.1. Access authorization (request URL)

To use the access authorization (request URL) function is shown below describing contents in the configuration file of Spring Security.
For basic settings, *Spring Security Overview* see.

. 6.5.2.1.1 `<sec: Intercept-url>` element set of

`: <Sec http>` is a child element of the element `<sec: Intercept-url>` URL to be controlled to the elements, be to describe the role to authorize,
It is possible to perform the admission control path units URL.

Below, I describe the setting example.

- spring-security.xml

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <sec: Intercept-url  pattern = "/ admin / *"  access = "hasRole
('ROLE_ADMIN')" />
    ! <- OMITTED ->
</ sec: http>
```

Attribute name	Description
pattern	Describe the subject of URL patterns to perform the access authorization. Wildcard "*", "**" is I can use.

	The "*", while only the same level is the target, the "**", the following full URL specified hierarchy, subject to authorization settings.
`access`	And access control expression in Spring EL expressions, I specify the accessible role.
`method`	I specify the HTTP method (GET or POST, etc.). with respect to only the specified method, to perform a URL pattern and matching. If not specified, it is applied to any HTTP method. Mainly I can take advantage of at the time of use of Web services using REST.
`requires-channel`	"Http", or I to specify the "https". I enforce the access of a specified protocol. If you do not specify, either can be accessed.

The attributes other than the above, <Intercept-url> see.

The case that the login user has role called "ROLE_USER" "ROLE_ADMIN" as an example, I show an example of setting.

- spring-security.xml

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <sec: Intercept-url  pattern = "/ Reserve / *"  access = "HasAnyRole
('ROLE_USER', 'ROLE_ADMIN')"  />  < ! - (1) ->
    <sec: Intercept-url  pattern = "/ admin / *"  access = "hasRole
('ROLE_ADMIN')"  />  ! <- (2) ->
    <sec: Intercept- url  pattern = "/ **"  access = "DenyAll"  />  <- (3) ->!
    <- OMITTED ->!
</ sec: http>
```

No.	Description
(1)	To access the "/ reserve / *" is "ROLE_USER" or "ROLE_ADMIN" role required. `hasAnyRole` For, I will be described later.
(2)	To access the "/ admin / *", it is necessary to "ROLE_ADMIN" role. `hasRole` For, I will be described later.
(3)	`denyAll` set to all of the pattern, For URL that authority setting has not been written is set to any user can not access settings. `denyAll` For, I will be described later.

Note

For a description order of URL pattern

The request from the client, the pattern is written in intercept-url, and matched in order from the top to perform the access authorization for the matched pattern. Therefore, description of the pattern is always, be described from a more limited pattern.

`<Sec: http>` to attribute `use-expressions = "true"` so that was the setting of, Spring EL expression is valid. `access` Spring EL expressions that describe the attributes are evaluated in the boolean value, if the expression is true, access is granted.
Below, I demonstrate the use cases.

- spring-security.xml

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <sec: Intercept-url  pattern = "/ admin / *"  access = "hasRole
```

```
('ROLE_ADMIN')" />    ! <- ( 1) ->
    <- OMITTED ->!
</ sec: http>
```

No.	Description
(1)	`hasRole ('role name')` and by specifying, and returns true if it holds the role login user specified.

Available Expression list example

Attribute name	Description
`hasRole ('role name')`	and if holding a roll designated by the user, and returns true.
`hasAnyRole ('roll 1', 'roll 2')`	and if holds one of the rolls designated by the user, and returns true.
`permitAll`	Always I return true. Also If unauthorized, it is noted that it can access.
`denyAll`	Always I return false.
`isAnonymous ()`	If an anonymous user, and returns true.
`isAuthenticated ()`	If authenticated user and returns true.
`isFullyAuthenticated ()`	If authentication of an anonymous user, or RememberMe function, it returns false.
`hasIpAddress ('IP address')`	The only access authorization request URL, and the JSP tags, it is effective. If requests from the specified IP address, it returns true.

Others, available Spring EL expressions, Common built-in expressions see.

Determination using the operator can also be performed.
In the following example, the roll, when matching the both requested IP address, it becomes accessible.

- spring-security.xml

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <sec: Intercept-url  pattern = "/ admin / *"  access = "hasRole ('ROLE_ADMIN')
and HasIpAddress ('192.168.10.1' ) " />
    <- OMITTED ->!
</ sec: http>
```

Available operators List

Operator	Description
`[Equation 1] and [Equation 2]`	Equation 1, Equation 2 is, if both of the true, returns true.
`[Equation 1] or [Equation 2]`	Either expression is, in the case of a true, returns true.
`! [Expression]`	False if the expression is true, I return true if false.

6.5.2.1.2. Configuring URL that does not perform access authorization control

Top page and login screen, such as the path to the css file, for authentication do not need a URL,
I use the pattern attribute, and security attributes of the http element.

- spring-security.xml

```
<Sec: http  pattern = "/ css / *"  Security = "none" />   <-! in the
attributes of the specified order (1) ~ (2) ->
<sec: http  pattern = "/ Login"  Security = "none" />
<sec: http  auto-config = "true"  use-expressions = "true" >
    ! <- OMITTED ->
</ sec: http>
```

No.	Description
(1)	`pattern` to describe the subject of the URL pattern to set the attribute. `pattern` If you do not want to describe the attributes, to match all of the pattern.
(2)	`security` to attribute `none` By specifying a, `pattern` path that has been described in the attributes, can be avoided the Spring Security filter chain.

6.5.2.1.3. Exception handling in the URL pattern

If you have access to the authorization has not been URL,
`Org.Springframework.Security.Access.AccessDeniedException` is thrown.
In the default setting, `Org.Springframework.Security.Web.Access.IxceptionTranslationFilter` it is set to
`org.springframework.security.web.access.AccessDeniedHandlerImpl` is, to return the error code 403.
the http element by setting the error page when access denied, it is possible to make a transition to the specified error page when access denied.

- spring-security.xml

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <-! OMITTED ->
    <sec: access-denied-handler  error-page = "/ AccessDeneidPage"  />   ! <- ( 1)
->
</ sec: http>
```

No.	Description
(1)	`:` `<Sec access-denied-handler>` element `error-page` to attribute, I specify the transition destination path.

6.5.2.2. Access authorization (JSP)

You can control the screen display items, custom JSP tags that Spring Security is providing `<sec: authorize>` use.

```
<% @ taglib prefix = "sec" uri = "Http://Www.Springframework.Org/security/tags" %>
```

It is a prerequisite that has been the use declaration setting of the tag library.

- `<Sec: authorize>` tag attributes List

Attribute name	Description
`access`	Describes the access control type. If true, the tag is evaluated.
`url`	If permission is given to the set URL, the tag is evaluated. I use to display the control of

	the link.
`method`	I specify the HTTP method (GET or POST, etc.). is utilized in conjunction with the url attribute, for only the given method, to perform the specified URL pattern and matching. If you do not specify, GET is applied.
`ifAllGranted`	If the set roles are given all the tag is evaluated. Role hierarchy function does not work.
`ifAnyGranted`	For the set roll, if any is given, the tag is evaluated. Role hierarchy function does not work.
`ifNotGranted`	If the configured roles are not given, the contents of the tag is evaluated. Role hierarchy function does not work.
`var`	Declare the page scope of variable to store the results of evaluation of the tag. Is utilized when performing the same authorization checks on the page.

Below, **:** `<sec authorize>` I show an example of the use of tags.

- spring-security.xml

```
<Div>
  <sec: authorize  access = "ROLE_USER" >   <-! (1) ->
      <P> This screen IS for ROLE_USER </ P>
  </ sec: authorize>
  <sec: authorize  url = "/ admin / menu " >   <- (2) ->!
      <P>
        <a  href = "/ admin / menu" > Go to admin screen </a>
      </ P>
  </ sec: authorize>
</ div>
```

No.	Description
(1)	Only if it has a "ROLE_USER" tag is displayed.
(2)	If access is authorized for "/ admin / menu", the tag is displayed.

Warning

`<Sec: authorize>` approval process by the tag, **it is not only the control of the screen display** for, even if it is not display a link in a particular authority, if the URL is presumed, would be able to access the direct link of the URL. Therefore, always, the aforementioned "access authorization (request URL)", or in combination the "access authorization (Method)" below, be carried out essentially authorization control.

6.5.2.3. Access authorization (Method)

For the method, I can authorize control.
Bean that are managed by Spring of DI container, subject to authorization.

Although two ways of approval described above was approved control at the application layer,
Admission control method-level I do for the domain layer (Service Class).
for a controlled want method `org.springframework.security.access.prepost.PreAuthorize` can be set annotations.

- spring-security.xml

```
<Sec: global-method-Security  pre-POST-Annotations = "enabled" />   <-! (1) ->
```

No.	Description
(1)	: <Sec global-method-security> element pre-post-annotations attributes enabled me to specify. The default is disabled is.

- Java code

```
Service
Transactional
public  class  UserServiceImpl  implements  UserSerice
    // OMITTED

    AttoPreAuthorize ( "hasRole ('ROLE_ADMIN')" )  // (1)
    Override
    public  User  create ( User  user )  {
       // OMITTED
    }


    AttoPreAuthorize ( "isAuthenticated ()" )
    Override
    public  User  Update ( User  user )  {
       // OMITTED
    }
}
```

No.	Description
(1)	Describes the access control type. Expression is evaluated before performing the method, if true, the method is executed. If false, Org.Springframework.Security.Access.AccessDeniedException is thrown. Possible values are *: setting of <sec intercept-url> element* Expression and mentioned in, and Spring Expression Language (SpEL) is an expression that is written in.

Tip

In the above configuration org.springframework.security.access.prepost.PreAuthorize besides, the following annotations I can use.

- org.springframework.security.access.prepost.PostAuthorize
- org.springframework.security.access.prepost.PreFilter
- org.springframework.security.access.prepost.PostFilter

These details Spring Security manual see.

Note

Of JSR-250 is a Spring Security in Java standard javax.annotation.security.RolesAllowed approval control by annotation is also possible, RolesAllowed can not be described by SpEL in. ThePreAuthorize if using SpEL, admission control in the same notation as the setting of the spring-security.xml

Note

Admission control for the request path, rather than annotate the method of Controller, to recommend that you do a set to spring-security.xml.

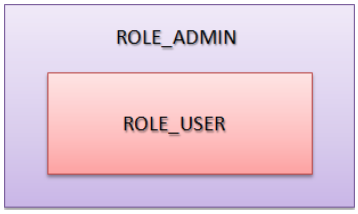
Service is not performed only through the Web, all patterns of the request path may not be performed authorization control of the Service as long as have been approved controlled. Service is not know is run from anywhere, it is preferable to use the annotation in case authorization control necessary.

6.5.3. How to extend

6.5.3.1. Roll hierarchy function

It is possible to set a hierarchical relationship to the role.
Roll is set to upper, it is possible to all the access granted to the lower roll.
If the relationship of the roll is complex, consider the hierarchy function.

Top the ROLE_ADMIN roll, I described in the example of setting a hierarchical relationship ROLE_USER as a subordinate role.



Picture - RoleHierarchy

In this case, if you set the access authorization as described below,
Users with the role of "ROLE_ADMIN" also can access the URL of "/ user / *".

Spring Security configuration file

```
<Sec: http  auto-config = "true"  use-expressions = "true" >
    <sec: Intercept-url  pattern = "/ user / *"  access = "HasAnyRole ('ROLE_USER')"  />
    ! <- OMITTED ->
</ sec: http>
```

Access authorization (request URL), access authorization (JSP), is different for each in the method of setting the access authorization (Method),
How to use, I described in the following.

6.5.3.1.1. Common Settings

I describe the common required settings.
To manage the hierarchical relationship
`org.springframework.security.access.hierarchicalroles.RoleHierarchy` I do Bean definition of class.

- spring-security.xml

```
<Bean  id = "RoleHierarchy"
    class =
"Org.Springframework.Security.Access.Hierarchicalroles.RoleHierarchyImpl" >  <-!
(1) ->
    <property  name = "hierarchy" >
        <value>  <-! (2 ) ->
            ROLE_ADMIN> ROLE_STAFF
            ROLE_STAFF> ROLE_USER
        </ Value>
    </ property>
</ bean>
```

No.	Description
(1)	`RoleHierarchy` default `org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl` I specify a class.
(2)	`hierarchy` I define a hierarchical relationship to the property. Format: [Top roll> lower roll]

In the example, STAFF for all resources that have been approved by the USER, can be accessed.
ADMIN is USER, to all of the resources that have been approved STAFF, can be accessed.

6.5.3.1.2. Access authorization (request URL), how they are used in access authorization (JSP)

Request URL, I describe the set of role hierarchy for JSP.

- spring-security.xml

```
<Bean  id = "WebExpressionHandler"
    class =
"Org.Springframework.Security.Web.Access.Expression.DiiefueiyuerutiWebSecurityExpr
essionHandler" >   <-! (1) ->
    <property  name = "RoleHierarchy"  ref = "RoleHierarchy" />   < ! - (2) ->
</ bean>

<Sec: http  auto-config = "true"  use-expression = "true" >
    <-! OMITTED ->
    <sec: expression-handler  ref = "WebExpressionHandler"  />   <-! (3) ->
</ sec: http>
```

No.	Description
(1)	In class `org.springframework.security.web.access.expression.DefaultWebSecurityExpressionHandler` I specify a.
(2)	`roleHierarchy` to property `RoleHierarchy` I set the Bean ID of the property.
(3)	`expression-handler` to the element, `Org.Springframework.Security.Access.Expression.SecurityExpressionHandler` I specify the Bean ID of the handler class that implements.

6.5.3.1.3. How they are used in access authorization (Method)

Role hierarchy setting will be described a case of performing the admission control is annotated to the method of Service.

- spring-security.xml

```
<Bean  id = "MethodExpressionHandler"
    class =
"Org.Springframework.Security.Access.Expression.Method.Diiefueiyuerutiemuitietchio
diSecurityExpressionHandler" >  <-! (1) ->
    <property  name = "RoleHierarchy"  ref = "RoleHierarchy" />  < ! - (2) ->
</ bean>

<Sec: global-method-Security  pre-POST-Annotations = "enabled" >
    <sec: expression-handler  ref = "MethodExpressionHandler"  />  <- (3) -!>
</ sec: global-method-Security >
```

No.	Description
(1)	In class `org.springframework.security.access.expression.method.DefaultMethodSecurityExpressionHandler` I specify a.
(2)	`roleHierarchy` to property `RoleHierarchy` I set the Bean ID of the property.
(3)	`expression-handler` to the element, `Org.Springframework.Security.Access.Expression.SecurityExpressionHandler` I specify the Bean ID of

the handler class that implements.