Authorization

Summary

All users in the website has limited access to according to the site policies and this is called authorization.
That is, the authority means the process of determining whether specific user can access the contents(information or function) provided by web site.

Description

XML or DB supervises the authorization and supports by level.
Support hierarchical roles: Hierarchical roles: ROLE_A > ROLE_B > ROLE_C
Allows to indicate objectDefinitionSource of FilterSecurityInterceptor in properties file.

All Authentication implement should save the arrangement of GrantedAuthority objects.
The arrangement of such GrantedAuthority indicates the authority allowed in authentication entity.
GrantedAuthority object returns to Authentication object by AuthenticationManager and if performing decision making on authority granting, it is read by AccessDecisionManager.
AccessDecisionManager determines whether the user has proper authority to access the security resource.
Determine whether to access the security resource by over 1 approval object.

AccessDecisionManager is configured as following.

```
<b:bean id="accessDecisionManager"
    class="org.springframework.security.vote.AffirmativeBased">
    <b:property name="allowIfAllAbstainDecisions" value="false" />
    <b:property name="decisionVoters">
        <b:list>
            <b:bean class="org.springframework.security.vote.RoleVoter">
                <b:property name="rolePrefix" value="" />
            </b:bean>
            <b:bean class="org.springframework.security.vote.AuthenticatedVoter" />
        </b:list>
    </b:property>
</b:bean>
```

- org.springframework.security.vote.AffirmativeBased: approved if over 1 object is allowed
- org.springframework.security.vote.UnanimousBased: approved if all objects allow
- decisionVoters: set the list for the approved object

Resource management

url

Requested web url is inspected to compare it with the URL stored in DB to designate the authority.
\A/sale/.*\.do\Z
\A/cvpl/((?!EgovCvplLogin\.do).)*\Z

filterSecurityInterceptor

FilterSecurityInterceptor is responsible for security of HTTP resources.
Similar to other security interceptor, FilterSecurityInterceptor needs to refer to AuthenticationManager and AccessDecisionManager.
In addition, FilterSecurityInterceptor sets the setting property to be applied to other different HTTP URL request.

```
<b:bean id="filterSecurityInterceptor"
        class="org.springframework.security.intercept.web.FilterSecurityInterceptor">
        <custom-filter before="FILTER_SECURITY_INTERCEPTOR" />
```

```
        <b:property name="authenticationManager"
                ref="authenticationManager" />
        <b:property name="accessDecisionManager"
                ref="accessDecisionManager" />
        <b:property name="objectDefinitionSource"
                ref="databaseObjectDefinitionSource" />
</b:bean>
```

- authenticationManager:
- accessDecisionManager:
- databaseObjectDefinitionSource:

databaseObjectDefinitionSource

This is the extended class of DefaultFilterInvocationDefinitionSource of Spring Security to reflect the runtime.

```
<b:bean id="databaseObjectDefinitionSource"
        class="org.springframework.security.intercept.web.EgovReloadableDefaultFilterInvocationDefi
nitionSource">
        <b:constructor-arg ref="regexUrlPathMatcher" />
        <b:constructor-arg ref="requestMap" />
        <b:property name="securedObjectService" ref="securedObjectService"/>
</b:bean>
```

requestMap

DB based protection resource mapping information is attained to provide the initial data of beans.

Set resourceType with url and call getRolesAndUrl() of securedObjectService to get the role and mapping information of url from DB.

```
<b:bean id="requestMap"
        class="egovframework.rte.fdl.security.intercept.ResourcesMapFactoryBean"
        init-method="init">
        <b:property name="securedObjectService" ref="securedObjectService"/>
        <b:property name="resourceType" value="url"/>
</b:bean>
```

regexUrlPathMatcher

Supports custom URL

```
<b:bean id="regexUrlPathMatcher"
        class="org.springframework.security.util.RegexUrlPathMatcher" />
```

method

In relations to method that request the resources
egovframework.rte.fdl.security.web.CategoryController.selectCategoryList

Spring Security provides MethodDefinitionSourceAdvisor that can be used with DefaultAdvisorAutoProxyCreator of spring as an alternative and automatically connect the security interceptor to the front part of bean defining MethodSecurityInterceptor using this.

_methodDefinitionSourceAdvisor

```
<b:bean id="_methodDefinitionSourceAdvisor"
class="org.springframework.security.intercept.method.aopalliance.MethodDefinitionSourceAdvisor">
        <b:constructor-arg value="_methodSecurityInterceptor" />
```

```
        <b:constructor-arg ref="_delegatingMethodDefinitionSource" />
</b:bean>
```

_methodSecurityInterceptor

To protect the resources on method request, MethodSecurityInterceptor should be added to the application context and the bean is chained to the interceptor that requires security.
Such connection is created using ProxyFactoryBean of spring or BeanNameAutoProxyCreator. Spring is similar to the type where several parts of spring is generally used.
MethodSecurityInterceptor is set as follows.

```
<b:bean id="_methodSecurityInterceptor"
class="org.springframework.security.intercept.method.aopalliance.MethodSecurityInterceptor">
        <b:property name="validateConfigAttributes" value="false" />
        <b:property name="authenticationManager" ref="authenticationManager"/>
        <b:property name="accessDecisionManager" ref="accessDecisionManager"/>
        <b:property name="objectDefinitionSource" ref="_delegatingMethodDefinitionSource" />
</b:bean>
<b:bean id="_delegatingMethodDefinitionSource"
class="org.springframework.security.intercept.method.DelegatingMethodDefinitionSource">
        <b:property name="methodDefinitionSources">
                <b:list>
                        <b:ref bean="methodDefinitionSources"/>
                        <b:bean
class="org.springframework.security.annotation.SecuredMethodDefinitionSource" />
                        <b:bean
class="org.springframework.security.annotation.Jsr250MethodDefinitionSource" />
                </b:list>
        </b:property>
</b:bean>
```

methodDefinitionSources

```
<b:bean id="methodDefinitionSources"
        class="org.springframework.security.intercept.method.MapBasedMethodDefinitionSource">
        <b:constructor-arg ref="methodMap" />
</b:bean>
```

methodMap

The mapping information of the DB-based protection resource is attained to provide the initial data of beans.
Set resourceType with method and call getRolesAndMethod() of securedObjectService to get mapping information of method and role in DB.

```
<b:bean id="methodMap"
        class="egovframework.rte.fdl.security.intercept.ResourcesMapFactoryBean"
        init-method="init">
        <b:property name="securedObjectService" ref="securedObjectService"/>
        <b:property name="resourceType" value="method"/>
</b:bean>
```

pointcut

It is provided as the initial data of beans by attaining the mapping information of DB based protected information.
Set the resourceType as pointcut and call getRolesAndPointcut() of securedObjectService to get the role and mapping information of Pointcut from DB. execution(* egovframework.rte.security..service.*Service.insert*(..))

```
<b:bean id="pointcutMap"
```

```
        class="egovframework.rte.fdl.security.intercept.ResourcesMapFactoryBean"
        init-method="init">
        <b:property name="securedObjectService" ref="securedObjectService"/>
        <b:property name="resourceType" value="pointcut"/>
</b:bean>
```

Role management

The role is managed by upper or lower level, and it is stored in the application context or DB.

When the hierarchy role is registered in XMl Context

```
<b:bean id="roleHierarchy"
                class="org.springframework.security.userdetails.hierarchicalroles.RoleHierarchyImpl"
>
        <b:property name="hierarchy">
                <b:value>
                        ROLE_ADMIN > ROLE_USER
                        ROLE_USER > ROLE_RESTRICTED
                        ROLE_RESTRICTED > IS_AUTHENTICATED_FULLY
                        IS_AUTHENTICATED_REMEMBERED > IS_AUTHENTICATED_ANONYMOUSLY
                </b:value>
        </b:property>
</b:bean>
```

When the DB manages level role

```
<b:bean id="roleHierarchy"
                class="org.springframework.security.userdetails.hierarchicalroles.RoleHierarchyImpl"
>
        <b:property name="hierarchy" ref="hierarchyStrings"/>
</b:bean>

<b:bean id="hierarchyStrings"
        class="egovframework.rte.fdl.security.userdetails.hierarchicalroles.HierarchyStringsFactoryBe
an"
        init-method="init">
        <b:property name="securedObjectService" ref="securedObjectService"/>
</b:bean>
<b:bean id="jdbcUserService"
                class="egovframework.rte.fdl.security.userdetails.jdbc.EgovJdbcUserDetailsManager"
>
        <b:property name="usersByUsernameQuery" value="SELECT
USER_ID,PASSWORD,ENABLED,USER_NAME,BIRTH_DAY,SSN FROM USERS WHERE USER_ID = ?"/>
        <b:property name="authoritiesByUsernameQuery" value="SELECT USER_ID,AUTHORITY
FROM AUTHORITIES WHERE USER_ID = ?"/>
        <b:property name="roleHierarchy" ref="roleHierarchy"/>
        <b:property name="dataSource" ref="dataSource"/>
        <b:property name="mapClass"
value="egovframework.rte.fdl.security.userdetails.EgovUserDetailsMapping"/>
</b:bean>
```

- class: egovframework.rte.fdl.security.userdetails.jdbc.EgovJdbcUserDetailsManager
- usersByUsernameQuery: inquires user information from user table for user authority.
- authoritiesByUsernameQuery: inquires user authority information from the user authority table for user authentication.
- roleHierarchy: sets the hierarchy role for hierarchical management of role.
- mapClass: sets the mapping class between session VO and session query for session use.
- dataSource: sets the dataSource for JDBC service.

Configuration

```
<b:bean id="securedObjectService"
        class="egovframework.rte.fdl.security.securedobject.impl.SecuredObjectServiceImpl">
        <b:property name="securedObjectDAO" ref="securedObjectDAO"/>
</b:bean>

<b:bean id="securedObjectDAO"
class="egovframework.rte.fdl.security.securedobject.impl.SecuredObjectDAO" >
        <b:property name="dataSource" ref="dataSource"/>
</b:bean>
```

Following property query is embedded in SecuredObjectDAO bean and can directly reflect the query changed due to DB schema difference or SQL statement in DBMS vendor.

Embedded queries are as follows.

- sqlHierarchicalRoles

```
SELECT a.child_role child, a.parent_role parent
FROM ROLES_HIERARCHY a LEFT JOIN ROLES_HIERARCHY b on (a.child_role = b.parent_role)
```

- sqlRolesAndUrl

```
SELECT a.resource_pattern url, b.authority authority
FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
WHERE a.resource_id = b.resource_id
AND a.resource_type = 'url' ORDER BY a.sort_order
```

- sqlRolesAndMethod

```
SELECT a.resource_pattern method, b.authority authority
FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
WHERE a.resource_id = b.resource_id
AND a.resource_type = 'method' ORDER BY a.sort_order
```

- sqlRolesAndPointcut

```
SELECT a.resource_pattern pointcut, b.authority authority
FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
WHERE a.resource_id = b.resource_id
AND a.resource_type = 'pointcut' ORDER BY a.sort_order
```

If not using SQL embedded in SecuredObjectDAO bean, designate SQL as follows.

```
<b:bean id="securedObjectDAO"
class="egovframework.rte.fdl.security.securedobject.impl.SecuredObjectDAO" >
        <b:property name="dataSource" ref="dataSource"/>
        <b:property name="sqlHierarchicalRoles">
                <b:value>
                        SELECT a.child_role child, a.parent_role parent
                        FROM ROLES_HIERARCHY a LEFT JOIN ROLES_HIERARCHY b on
(a.child_role = b.parent_role)
                </b:value>
        </b:property>
        <b:property name="sqlRolesAndUrl">
                <b:value>
                        SELECT a.resource_pattern url, b.authority authority
                        FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
                        WHERE a.resource_id = b.resource_id
                        AND a.resource_type = 'url' ORDER BY a.sort_order
                </b:value>
        </b:property>
```

```
        <b:property name="sqlRolesAndMethod">
                <b:value>
            SELECT a.resource_pattern method, b.authority authority
                        FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
                        WHERE a.resource_id = b.resource_id
                        AND a.resource_type = 'method' ORDER BY a.sort_order
                </b:value>
        </b:property>
        <b:property name="sqlRolesAndPointcut">
                <b:value>
                        SELECT a.resource_pattern pointcut, b.authority authority
                        FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
                        WHERE a.resource_id = b.resource_id
                        AND a.resource_type = 'pointcut' ORDER BY a.sort_order
                </b:value>
        </b:property>
</b:bean>
```

Using session

Importing roles

```
List<String> authorities = EgovUserDetailsHelper.getAuthorities();

// 1. Check for authority in authorites TRUE/FALSE
assertTrue(authorities.contains("ROLE_USER"));
assertTrue(authorities.contains("ROLE_RESTRICTED"));
assertTrue(authorities.contains("IS_AUTHENTICATED_ANONYMOUSLY"));
assertTrue(authorities.contains("IS_AUTHENTICATED_FULLY"));
assertTrue(authorities.contains("IS_AUTHENTICATED_REMEMBERED"));

// 2. Multiple configurations of roles in authorites
for (Iterator<String> it = authorities.iterator(); it.hasNext();) {
        String auth = it.next();
}

// 3. When there is only one role in authorites
String auth = (String) authorities.toArray()[0];
```

N. Reference