http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte3:fdl:server_security:upgrade

Upgrade

Outline

Upgrading from eGovFramework 2.7(Spring Security 2.0.4) to eGovFramework 3.0(Spring Security 3.2.3) requires not only configuration upgrade but also source code amendments.

Notable Changes (Spring Security)

Changes in dependencies and packages

- spring-security-core (org.springframework.security.core, org.springframework.security.access, etc.)
- spring-security-web (org.springframework.security.web)
- spring-security-config (org.springframework.security.config)

Changes in API

- SpringSecurityException deleted
- ConfigAttributeDefinition ⇒ Collection<ConfigAttribute>
- SavedRequest : class ⇒ interface (replcing DefaultSavedRequest)

Other

- Multiple http elements now supported
- Stateless authentication now supported
- DebugFilter added (for debugging only)
- Supports the expression “hasPermission” (authorize JSP tag)
- , etc.

How to upgrade runtime environment

1. Amend dependency

```
<dependency>
    <groupId>egovframework.rte</groupId>
    <artifactId>egovframework.rte.fdl.security</artifactId>
    <version>3.0.0</version>
</dependency>
```

- Check for the latest versions (patch version, etc.) before applying upgrade

2. Amend web.xml

Change packages of HttpSessionEventPublisher listener on web.xml when restricting accesses.

- Conventional:

```
<listener>
        <listener-class>org.springframework.security.ui.session.HttpSessionEventPublisher</listener-class>
</listener>
```

- Updated:

```
<listener>
	<listener-class>org.springframework.security.web.session.HttpSessionEventPublisher</listener-class>
</listener>
```

3. Amend Spring Security Configuration

Refer to the following configuration for amendment (Spring Security packages, etc.):

```
<beans:beans	xmlns="http://www.springframework.org/schema/security"
	xmlns:beans="http://www.springframework.org/schema/beans"
	xmlns:aop="http://www.springframework.org/schema/aop"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
								http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-3.2.xsd
								http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.2.xsd">

	<beans:bean id="securedObjectService"
class="egovframework.rte.fdl.security.securedobject.impl.SecuredObjectServiceImpl">
		<beans:property name="securedObjectDAO" ref="securedObjectDAO"/>
		<beans:property name="requestMatcherType" value="regex"/>		<!--	default : ant -->
	</beans:bean>

	<beans:bean id="securedObjectDAO"
class="egovframework.rte.fdl.security.securedobject.impl.SecuredObjectDAO" >
		<beans:property name="dataSource" ref="dataSource"/>
		<!--
		<beans:property name="sqlHierarchicalRoles">
			<beans: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)
			</beans:value>
		</beans:property>
		<beans:property name="sqlRolesAndUrl">
			<beans: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
			</beans:value>
		</beans:property>
		<beans:property name="sqlRolesAndMethod">
			<beans: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
			</beans:value>
		</beans:property>
		<beans:property name="sqlRolesAndPointcut">
			<beans: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
			</beans:value>
		</beans:property>
```

```
-->
</beans:bean>

<!-- Paragraph deletedas not required -->
<!--
<beans:bean id="userDetailsServiceWrapper"
class="org.springframework.security.userdetails.hierarchicalroles.UserDetailsServiceWrapper">
<beans:property name="roleHierarchy" ref="roleHierarchy"/>
<beans:property name="userDetailsService" ref="jdbcUserService"/>
</beans:bean>
-->

<beans:bean id="roleHierarchy"
class="org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl" >
<!-- Using XML
<beans:property name="hierarchy">
<beans:value>
ROLE_ADMIN > ROLE_USER
ROLE_USER > ROLE_RESTRICTED
ROLE_RESTRICTED > IS_AUTHENTICATED_FULLY
IS_AUTHENTICATED_REMEMBERED >
IS_AUTHENTICATED_ANONYMOUSLY
</beans:value>
</beans:property>
-->
<!-- Using DB -->
<beans:property name="hierarchy" ref="hierarchyStrings"/>
</beans:bean>

<beans:bean id="hierarchyStrings"
class="egovframework.rte.fdl.security.userdetails.hierarchicalroles.HierarchyStringsFactoryBean" init-method="init">
<beans:property name="securedObjectService" ref="securedObjectService"/>
</beans:bean>

<!--
Access Decision Manager is auto-generated. No declaration is necessary.
bean id : org.springframework.security.access.vote.AffirmativeBased#0
※  Figures such as #0 are sequentially generated.
-->
<!--
<beans:bean id="accessDecisionManager"
class="org.springframework.security.access.vote.AffirmativeBased">
<beans:property name="allowIfAllAbstainDecisions" value="false" />
<beans:property name="decisionVoters">
<beans:list>
<beans:bean class="org.springframework.security.access.vote.RoleVoter">
<beans:property name="rolePrefix" value="ROLE_" />
</beans:bean>
<beans:bean
class="org.springframework.security.access.vote.AuthenticatedVoter" />
</beans:list>
</beans:property>
</beans:bean>
-->

<beans:bean id="filterSecurityInterceptor"
class="org.springframework.security.web.access.intercept.FilterSecurityInterceptor">
<beans:property name="authenticationManager"
ref="org.springframework.security.authenticationManager" />
```

```
                <beans:property name="accessDecisionManager"
ref="org.springframework.security.access.vote.AffirmativeBased#0" />
                <beans:property name="securityMetadataSource" ref="databaseSecurityMetadataSource" />
        </beans:bean>

        <beans:bean id="databaseSecurityMetadataSource"
class="egovframework.rte.fdl.security.intercept.EgovReloadableFilterInvocationSecurityMetadataSource">
                <beans:constructor-arg ref="requestMap" />
                <beans:property name="securedObjectService" ref="securedObjectService"/>
        </beans:bean>

        <!--    url    -->
        <beans:bean id="requestMap"
        class="egovframework.rte.fdl.security.intercept.UrlResourcesMapFactoryBean" init-method="init">
                <beans:property name="securedObjectService" ref="securedObjectService"/>
        </beans:bean>

        <!-- Assignment unnecessary : See request-matcher -->
        <!--
        <beans:bean id="regexUrlPathMatcher"
class="org.springframework.security.web.util.matcher.RegexRequestMatcher" />
        -->


        <http pattern="/css/ *  * " security="none"/>

    <http pattern="/images/ *  * " security="none"/>

        <http pattern="/js/ *  * " security="none"/>
        <http pattern="\A/WEB-INF/jsp/.*\Z" request-matcher="regex" security="none"/>

        <http access-denied-page="/system/accessDenied.do" request-matcher="regex">
                <form-login login-processing-url="/j_spring_security_check"
                                        authentication-failure-
url="/cvpl/EgovCvplLogin.do?login_error=1"
                                        default-target-url="/index.jsp?flag=L"
                                        login-page="/cvpl/EgovCvplLogin.do" />
                <anonymous/>
                <logout logout-success-url="/cvpl/EgovCvplLogin.do"/>

                <!-- for authorization -->
                <custom-filter before="FILTER_SECURITY_INTERCEPTOR" ref="filterSecurityInterceptor"/>
        </http>

        <!--
        Default bean id for authentication-manager :    org.springframework.security.authenticationManager
                (You can replace the bean id into an alias)
        -->
        <authentication-manager>
                <authentication-provider user-service-ref="jdbcUserService">
                        <password-encoder    hash="sha-256" base64="true"/>
                </authentication-provider>
        </authentication-manager>

        <!-- userDetailsServiceWrapper -->
        <!-- customizing user table, authorities table -->

        <!--<jdbc-user-service id="jdbcUserService" data-source-ref="dataSource"
                users-by-username-query="SELECT USER_ID,PASSWORD,ENABLED,BIRTH_DAY FROM
USERS WHERE USER_ID = ?"
                authorities-by-username-query="SELECT USER_ID,AUTHORITY FROM AUTHORITIES
WHERE USER_ID = ?"/>-->
```

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

		<!-- method -->
		<beans:bean id="methodSecurityMetadataSourceAdvisor"
class="org.springframework.security.access.intercept.aopalliance.MethodSecurityMetadataSourceAdvisor">
			<beans:constructor-arg value="methodSecurityInterceptor" />
			<beans:constructor-arg ref="delegatingMethodSecurityMetadataSource" />
			<beans:constructor-arg value="delegatingMethodSecurityMetadataSource" />
		</beans:bean>

		<beans:bean id="methodSecurityInterceptor"
class="org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor">
			<beans:property name="validateConfigAttributes" value="false" />
			<beans:property name="authenticationManager"
ref="org.springframework.security.authenticationManager"/>
			<beans:property name="accessDecisionManager"
ref="org.springframework.security.access.vote.AffirmativeBased#0"/>
			<beans:property name="securityMetadataSource"
ref="delegatingMethodSecurityMetadataSource" />
		</beans:bean>

	<beans:bean id="delegatingMethodSecurityMetadataSource"
class="org.springframework.security.access.method.DelegatingMethodSecurityMetadataSource">
		<beans:constructor-arg>
			<beans:list>
				<beans:ref bean="methodSecurityMetadataSources" />
				<beans:bean
class="org.springframework.security.access.annotation.SecuredAnnotationSecurityMetadataSource" />
				<beans:bean
class="org.springframework.security.access.annotation.Jsr250MethodSecurityMetadataSource" />
			</beans:list>
		</beans:constructor-arg>
	</beans:bean>

		<beans:bean id="methodSecurityMetadataSources"
class="org.springframework.security.access.method.MapBasedMethodSecurityMetadataSource">
			<beans:constructor-arg ref="methodMap" />
		</beans:bean>

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

		<!-- pointcut -->
		<!-- if no map, there is a error that "this map must not be empty; it must contain at least one entry" -->
		<!-- // so there is dummy entry
```

```
        <beans:bean id="protectPointcutPostProcessor"
class="org.springframework.security.config.method.ProtectPointcutPostProcessor">
                <beans:constructor-arg ref="methodSecurityMetadataSources" />
                <beans:property name="pointcutMap" ref="pointcutMap"/>
        </beans:bean>

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

4. Unnecessary classes cleared

Unnecessary server security source codes are cleared as follows:
Example:

- org.springframework.security.intercept.web.EgovReloadableDefaultFilterInvocationDefinitionSource : Unnecessary with the package amended (egovframework.rte.fdl.security.intercept.EgovReloadableFilterInvocationSecurityMetadataSource)
- …security.intercept.*, …security.securedobject.securedobject.*, …security.securedobject.userdetails.*, etc. : Unnecessary as the upgraded runtime environment pacakge is referred (egovframework.rte.fdl.security.*).

5. Mapping Class mapRow Method changed

The mapClass designated by jdbcUserService(egovframework.rte.fdl.security.userdetails.jdbc.EgovJdbcUserDetailsManager) is intended to extend the class EgovUsersByUsernameMapping, where the return type of the method mapRow() of the concerned EgovUsersByUsernameMapping is replaced from Object into EgovUserDtails.

- Conventional:

```
public class EgovSessionMapping extends EgovUsersByUsernameMapping {
    ...

    @Override
    protected Object mapRow(ResultSet rs, int rownum) throws SQLException {
        ...
    }
}
```

- Updated:

```
public class EgovSessionMapping extends EgovUsersByUsernameMapping {
    ...

    @Override
    protected EgovUserDetails mapRow(ResultSet rs, int rownum) throws SQLException {
        ...
    }
}
```

6. Reference packages and classes changed

Packages of the reference classes are changed by changing packages related to spring security.

- org.springframework.security.Authentication → org.springframework.security.core.Authentication

- org.springframework.security.GrantedAuthority → org.springframework.security.core.GrantedAuthority
- org.springframework.security.context.SecurityContext → org.springframework.security.core.context.SecurityContext
- org.springframework.security.context.SecurityContextHolder → org.springframework.security.core.context.SecurityContextHolder
- , etc.

7. GrantedAuthority changed (type change from array to collection)

GrantedAuthority[] → Collection<GrantedAuthority>

- Previous:

```
GrantedAuthority[] authorities = authentication.getAuthorities();

for (int i = 0; i < authorities.length; i++) {
	listAuth.add(authorities[i].getAuthority());
}
```

- Changed for:

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

for (GrantedAuthority authority : authorities) {
	listAuth.add(authority.getAuthority());
}
```

8. getAuthentication() in SecurityContext changed

Contrary to the conventional null return from getAuthentication() of SecurityContext:

```
SecurityContext context = SecurityContextHolder.getContext();
Authentication authentication = context.getAuthentication();
if (EgovObjectUtil.isNull(authentication)) {
	return null;
}
```

getAuthentication() is now deemed valid as follows:

```
SecurityContext context = SecurityContextHolder.getContext();
Authentication authentication = context.getAuthentication();

if (EgovObjectUtil.isNull(authentication)) {
	log.debug("## authentication object is null!!");
	return null;
}

if (authentication.getPrincipal() instanceof EgovUserDetails) {
	EgovUserDetails details = (EgovUserDetails) authentication.getPrincipal();


	log.debug("## EgovUserDetailsHelper.getAuthenticatedUser : AuthenticatedUser is {}",
details.getUsername());

	return details.getEgovUserVO();
} else {
	return authentication.getPrincipal();
}
```

The user information is thus based upon the runtime environment (egovframework.rte.fdl.security.userdetails.util.EgovUserDetailsHelper).

9. Authentication of GET Method no longer valid

Authentication of GET method to call j_spring_security_check is no longer valid (this applies when j_spring_security_check URL is called for internal redirection)
Error message thrown: Authentication request failed: org.springframework.security.authentication.AuthenticationServiceException: Authentication method not supported: GET

Where the error message is thrown, you need to call the filter chain as follows:

- Conventional:

```
return "redirect:/j_spring_security_check?j_username=" + resultVO.getUserSe() + resultVO.getId() + "&j_password=" + resultVO.getUniqId();
```

- Changed for (for general controller only):

```
@RequestMapping(value="/uat/uia/actionSecurityLogin.do")
public String actionSecurityLogin(@ModelAttribute("loginVO") LoginVO loginVO,
	HttpServletRequest request, HttpServletResponse response,
	ModelMap model)
		throws Exception {
	...

	UsernamePasswordAuthenticationFilter springSecurity = null;

	ApplicationContext act = WebApplicationContextUtils.getRequiredWebApplicationContext(request.getSession().getServletContext());
	@SuppressWarnings("rawtypes")
	Map beans = act.getBeansOfType(UsernamePasswordAuthenticationFilter.class);
	if (beans.size() > 0) {
		springSecurity = (UsernamePasswordAuthenticationFilter)beans.values().toArray()[0];
	} else {
		throw new IllegalStateException("No AuthenticationProcessingFilter");
	}

	springSecurity.setContinueChainBeforeSuccessfulAuthentication(false); // “false”does not activate the filter chain call (you need to configure “false” is you do not want to activate filter chain call)

	springSecurity.doFilter(
		new RequestWrapperForSecurity(request, resultVO.getUserSe() + resultVO.getId() , resultVO.getUniqId()),
		response, null);

	return “forward:/cmm/main/mainPage.do”;		// Page only forwarded when successful (you cannot have the page redirected)

	...
}

...
class RequestWrapperForSecurity extends HttpServletRequestWrapper {
	private String username = null;
	private String password = null;

	public RequestWrapperForSecurity(HttpServletRequest request, String username, String password) {
		super(request);
```

```
                this.username = username;
                this.password = password;
        }

        @Override
        public String getRequestURI() {
                return ((HttpServletRequest)super.getRequest()).getContextPath() + "/j_spring_security_check";
        }

        @Override
        public String getParameter(String name) {
        if (name.equals("j_username")) {
        return username;
        }

        if (name.equals("j_password")) {
        return password;
        }

        return super.getParameter(name);
    }
}
```

- Changed for (for filter):

```
public class EgovSpringSecurityLoginFilter implements Filter{
        private FilterConfig config;

        public void init(FilterConfig filterConfig) throws ServletException {
                this.config = filterConfig;
        }


        public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException {
                ...

                HttpServletRequest httpRequest = (HttpServletRequest)request;
                HttpServletResponse httpResponse = (HttpServletResponse)response;

                ...

                UsernamePasswordAuthenticationFilter springSecurity = null;

                ApplicationContext act =
WebApplicationContextUtils.getRequiredWebApplicationContext(config.getServletContext());
                @SuppressWarnings("rawtypes")
                Map beans = act.getBeansOfType(UsernamePasswordAuthenticationFilter.class);
                if (beans.size() > 0) {
                        springSecurity = (UsernamePasswordAuthenticationFilter)beans.values().toArray()[0];
                } else {
                        throw new IllegalStateException("No AuthenticationProcessingFilter");
                }

                springSecurity.setContinueChainBeforeSuccessfulAuthentication(false); // “false”does not
activate the filter chain call (you need to configure “false” is you do not want to activate filter chain call)

                springSecurity.doFilter(
```

```
                                new RequestWrapperForSecurity(request, resultVO.getUserSe() + resultVO.getId() ,
resultVO.getUniqId()),
                                response, chain);

                        ...
                }

                ...
}

class RequestWrapperForSecurity extends HttpServletRequestWrapper {
                private String username = null;
                private String password = null;

                public RequestWrapperForSecurity(HttpServletRequest request, String username, String password) {
                        super(request);

                        this.username = username;
                        this.password = password;
                }

                @Override
                public String getRequestURI() {
                        return ((HttpServletRequest)super.getRequest()).getContextPath() + "/j_spring_security_check";
                }

                @Override
                public String getParameter(String name) {
                if (name.equals("j_username")) {
                return username;
                }

                if (name.equals("j_password")) {
                return password;
                }

                return super.getParameter(name);
        }
}
```