Securing Flows

Summary

Securing is an important concept for any application. Spring Security is a proven security platform that can integrate with your application at multiple levels. This section will focus on securing flow execution.

Description

How do I secure a flow?

Securing flow execution is a three step process:

1. Configure Spring Security with authentication and authorization rules
2. Annotate the flow definition with the secured element to define the security rules
3. Add the SecurityFlowExecutionListener to process the security rules.

The secured element

The secured element designates that its containing element should apply the authorization check before fully entering. This may not occur more then once per stage of the flow execution that is secured.

Three phases of flow execution can be secured: flows, states and transitions. In each case the syntax for the secured element is identical. The secured element is located inside the element it is securing. For example, to secure a state the secured element occurs directly inside that state:

```
<view-state id="secured-view">
 <secured attributes="ROLE_USER" />
 ...
</view-state>
```

Security attributes

The attributes attribute is a comma separated list of Spring Security authorization attributes. Often, these are specific security roles. The attributes are compared against the user's granted attributes by a Spring Security access decision manager.

```
 <secured attributes="ROLE_USER" />
```

By default, a role based access decision manager is used to determine if the user is allowed access. This will need to be overridden if your application is not using authorization roles.

Matching type

There are two types of matching available: **any** and **all**.

```
 <secured attributes="ROLE_USER, ROLE_ANONYMOUS" match="any" />
```

This attribute is optional. If not defined, the default value is any.

SecurityFlowExecutionListener

Add to Web Flow setting.
Can be applied to flow executor when imSecurityFlowExecutionListener is defined in Web Flow setting.

```
<webflow:flow-executor id="flowExecutor"
	flow-registry="flowRegistry">
	<webflow:flow-execution-listeners>
		<webflow:listener ref="securityFlowExecutionListener" />
	</webflow:flow-execution-listeners>
</webflow:flow-executor>
<bean id="securityFlowExecutionListener"
	class="org.springframework.webflow.security.SecurityFlowExecutionListener" />
```

If access is denied by security setting, AccessDeniedException occurs.
While role-based decision-making is not made by default, custom decision-making manager can be designated.

```
<bean id="securityFlowExecutionListener"
	class="org.springframework.webflow.security.SecurityFlowExecutionListener">
	<property name="accessDecisionManager" ref="myCustomAccessDecisionManager" />
</bean>
```

Configuring Spring Security

Spring configuration

The Spring configuration defines http specifics (such as protected URLs and login/logout mechanics) and the authentication-provider.

```
<security:http auto-config="true">
	<security:form-login login-page="/spring/login"
		login-processing-url="/spring/loginProcess" default-target-url="/spring/main"
		authentication-failure-url="/spring/login?login_error=1" />
	<security:logout logout-url="/spring/logout" logout-success-url="/spring/logout-success" />
</security:http>

<security:authentication-provider>
	<security:password-encoder hash="md5" />
	<security:user-service>
		<security:user name="keith" password="417c7382b16c395bc25b5da1398cf076"
	authorities="ROLE_USER,ROLE_SUPERVISOR" />
		<security:user name="erwin" password="12430911a8af075c6f41c6976af22b09"
	authorities="ROLE_USER,ROLE_SUPERVISOR" />
		<security:user name="jeremy" password="57c6cbff0d421449be820763f03139eb"
authorities="ROLE_USER" />
		<security:user name="scott" password="942f2339bf50796de535a384f0d1af3e"
	authorities="ROLE_USER" />
	</security:user-service>
</security:authentication-provider>
```

web.xml Configuration

Filter configuration.(SS basics)

```
<filter>
	<filter-name>springSecurityFilterChain</filter-name>
	<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
	<filter-name>springSecurityFilterChain</filter-name>
	<url-pattern>/*</url-pattern>
</filter-mapping>
```

Reference

- Spring Web Flow reference 2.0.x
- Spring Web-Flow Framework Reference beta with Korean (by Park Chan Wook)
- Whiteship's Note