

Acegi Security System for Spring Framework



Acegi
**SECURITY
SYSTEM**
FOR SPRING

Part 1

Securing URL Access

Outline

- Intro
- Case Study
- Servlet Filter and Context Loader
- Authentication Processing Filter
- Storing username, password, and role
- Http Session Context Integration Filter
- Security Enforcement Filter
- Filter Invocation Interceptor
- Conclusion

Introduction to Acegi



Acegi ++

- ❑ Acegi Security System for Spring
 - <http://acegisecurity.sourceforge.net>
- ❑ Security control in various granularity
 - Securing URL
 - Securing Method Invocation
 - Securing Object Instance
- ❑ Non-invasive architecture
 - Servlet Filter based
- ❑ Modular Architecture
 - Pluggable ACL backend
 - Pluggable authentication mechanism
 - Integration with Application Server

Alternatives

- ❑ Seraph: <http://opensource.atlassian.com/seraph/>
- ❑ JSai - Servlet Security : <http://oss.ipov.org/jsai/>
- ❑ Gabriel : <http://gabriel.codehaus.org/>
- ❑ JOSSO : <http://www.josso.org/>
- ❑ Kasai: <http://www.manentiasoftware.com/kasai/goToHome.action>
- ❑ jPAM : <http://jpam.sourceforge.net/>
- ❑ OpenSAML : <http://www.opensaml.org/>
- ❑ JAAS
- ❑ Custom Implementation

Acegi and Spring

- ❑ Acegi use Spring for its lifecycle management and configuration
- ❑ Acegi does not mandate our application to use Spring

Security Granularity

- URL authorization
 - <http://apps:8080/index.htm> -> for public
 - <http://apps:8080/user.htm> -> for authorized user
- Method Invocation
 - public void getData() -> all user
 - public void modifyData() -> supervisor only
- Object instance
 - order.getValue() < \$100 -> all user
 - order.getValue() > \$100 -> supervisor only
 - customer.getName() == endy -> account mgr only

Non-invasive architecture

- Use servlet filter to intercept HttpRequest
- Use integration filter to transparently store and access security data in session

Convolutated Servlet Controller

□ Avoid this :

```
public void doGet(HttpServletRequest req, ... )
{
    User u = (User)req.getSession().getAttribute("user");
    if (!u.getDepartment().equals("marketing")){
        response.sendRedirect("access_denied.jsp");
    }
    String id = req.getParameter("order_id");
    Order o = dao.getOrder(id);
    o.approve();
}
```

Clean Servlet Controller

- Should be like this :

```
public void doGet(HttpServletRequest req, ... )
{
    String id = req.getParameter("order_id");
    Order o = dao.getOrder(id);
    o.approve();
}
```

Case Study

- ❑ Simple Purchase Order Application
- ❑ Static pages only :
 - list.htm : display list of Purchase Orders
 - ❑ Authenticated user and supervisor can view
 - create.htm : create new Purchase Order
 - ❑ Only user can view
 - approve.htm : display approval screen for selected Purchase Order
 - ❑ Only supervisor can view
 - login.htm : login form for unauthenticated user
 - ❑ All visitor, user, and supervisor can open
- ❑ No Java code (static html only)
- ❑ No control flow code (provided by Acegi)
- ❑ No authentication logic code (provided by Acegi)

Servlet Filter and Context Loader

- ❑ Configured in web.xml
- ❑ Servlet Filter intercept all request (/*)
- ❑ Context Loader is a Spring beans definition for loading Acegi security interceptor and collaborators

Servlet Filter Configuration

```
<filter>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <filter-class>
    net.sf.acegisecurity.util.FilterToBeanProxy
  </filter-class>
  <init-param>
    <param-name>targetClass</param-name>
    <param-value>
      net.sf.acegisecurity.util.FilterChainProxy
    </param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>Acegi Filter Chain Proxy</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Servlet Listener

```
<listener>
```

```
  <listener-class>
```

```
    org.springframework.web.context.ContextLoaderListener
```

```
  </listener-class>
```

```
</listener>
```

```
<listener>
```

```
  <listener-class>
```

```
    net.sf.acegisecurity.ui.session.HttpSessionEventPublisher
```

```
  </listener-class>
```

```
</listener>
```

Context Loader

```
<context-param>
  <param-name>
    contextConfigLocation
  </param-name>
  <param-value>
    /WEB-INF/acegi-auth-ctx.xml
  </param-value>
</context-param>
```

Beans in Context

- Filter Chain
 - Define filter execution orders
- Http Session Context Integration Filter
 - Automates storing/retrieval of security object to/from http session
- Authentication Processing Filter
 - Automates login process control flow
- Security Enforcement Filter
 - Define authentication entry point : html page used for entering username and password
 - Define filterSecurityInterceptor : we will elaborate later

Filter Chain configuration

```
<bean id="filterChainProxy"  
  class="net.sf.acegisecurity.util.FilterChainProxy">  
  
  <property name="filterInvocationDefinitionSource">  
    <value>  
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON  
      PATTERN_TYPE_APACHE_ANT  
      /**=httpSessionContextIntegrationFilter,  
        authenticationProcessingFilter,  
        securityEnforcementFilter  
    </value>  
  </property>  
  
</bean>
```

httpSessionContextIntegrationFilter configuration

```
<bean id="httpSessionContextIntegrationFilter"  
      class="net.sf.acegisecurity.context.  
            HttpSessionContextIntegrationFilter">  
  <property name="context">  
    <value>  
      net.sf.acegisecurity.context.  
      security.SecureContextImpl  
    </value>  
  </property>  
</bean>
```

authenticationProcessingFilter configuration

```
<bean id="authenticationProcessingFilter"  
      class="net.sf.acegisecurity.ui.webapp.  
          AuthenticationProcessingFilter">  
  <property name="authenticationManager">  
    <ref bean="authenticationManager"/>  
  </property>  
  <property name="authenticationFailureUrl">  
    <value>/login.htm</value>  
  </property>  
  <property name="defaultTargetUrl">  
    <value>/list.htm</value>  
  </property>  
  <property name="filterProcessesUrl">  
    <value>/cek_login</value>  
  </property>  
</bean>
```

authenticationProcessingFilter

explanation

- authenticationManager
 - Define how to commence authentication
- loginFailedUrl
 - Define where to redirect user upon login failed event
- defaultTargetUrl
 - Define where to redirect user after login process successfully performed (only if no target found in session)
- filterProcessUrl
 - Define what to put in `<form action="">` field

authenticationManager configuration

```
<bean id="authenticationManager"
      class="net.sf.acegisecurity.providers.ProviderManager">
  <property name="providers">
    <list>
      <ref bean="daoAuthenticationProvider"/>
    </list>
  </property>
</bean>
<bean id="daoAuthenticationProvider"
      class="net.sf.acegisecurity.providers.dao.DaoAuthenticationProvider">
  <property name="authenticationDao">
    <ref local="memoryAuthenticationDao"/>
  </property>
</bean>
<bean id="memoryAuthenticationDao"
      class="net.sf.acegisecurity.providers.dao.memory.InMemoryDaoImpl">
  <property name="userMap">
    <value>
      supervisor=supervisor,ROLE_USER,ROLE_SUPERVISOR
      user=user,ROLE_USER
    </value>
  </property>
</bean>
```

authenticationManager

explanation

- We can supply one or more username/password storage using daoAuthProvider
- For the sake of simplicity, we use InMemoryDaoImpl
- JdbcDaoImpl is available :
 - Using default schema (HypersonicSQL)
 - Using custom schema (configurable SQL Query)

Security Enforcement Filter configuration

```
<bean id="securityEnforcementFilter"
      class="net.sf.acegisecurity.intercept.web.
            SecurityEnforcementFilter">
  <property name="filterSecurityInterceptor">
    <ref bean="filterInvocationInterceptor"/>
  </property>
  <property name="authenticationEntryPoint">
    <ref bean="authenticationEntryPoint"/>
  </property>
</bean>

<bean id="authenticationEntryPoint"
      class="net.sf.acegisecurity.ui.webapp.
            AuthenticationProcessingFilterEntryPoint">
  <property name="loginFormUrl">
    <value>/login.htm</value>
  </property>
</bean>
```

Security Enforcement Filter

explanation

- ❑ Perform security check as defined in `filterSecurityInterceptor`
- ❑ If `filterSecurityInterceptor` found unauthenticated user try to access protected resources, redirect to `authenticationEntryPoint`
- ❑ `authenticationEntryPoint` define URL used to display login form

Filter Security Interceptor configuration

```
<bean id="filterInvocationInterceptor"  
      class="net.sf.acegisecurity.intercept.web.  
          FilterSecurityInterceptor">  
  <property name="authenticationManager">  
    <ref bean="authenticationManager"/></property>  
  <property name="accessDecisionManager">  
    <ref bean="accessDecisionManager"/></property>  
  <property name="objectDefinitionSource">  
    <value>  
      CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON  
      PATTERN_TYPE_APACHE_ANT  
      /approve.htm=ROLE_SUPERVISOR  
      /create.htm=ROLE_USER  
      /list.htm=ROLE_USER  
    </value>  
  </property>  
</bean>
```

Filter Security Interceptor

explanation

- ❑ Object Definition Source
 - Define protected URLs
 - Choose ant-like syntax or regular expression syntax
- ❑ Authentication Manager
 - Define authentication mechanism to be used, as configured in `authenticationProcessingFilter`
- ❑ Access Decision Manager
 - Compare accessed URL with accessing user and decide whether the user has sufficient authorization

Access Decision Manager configuration

```
<bean id="accessDecisionManager"  
      class="net.sf.acegisecurity.vote.  
          UnanimousBased">  
  <property name="allowIfAllAbstainDecisions">  
    <value>false</value>  
  </property>  
  <property name="decisionVoters">  
    <list>  
      <ref local="roleVoter"/>  
    </list>  
  </property>  
</bean>
```

Access Decision Manager

explanation

- ❑ Use voting mechanism to decide access control
- ❑ Currently has 3 implementation:
 - Unanimous based : all decisionVoters has to vote ACCESS_GRANTED
 - Affirmative based : one voter votes ACCESS_GRANTED is enough
 - Consensus based : majority wins
- ❑ Sample apps use only one voter : roleVoter

Role Voter configuration

```
<bean id="roleVoter"  
      class="net.sf.acegisecurity.vote.  
          RoleVoter"/>
```

RoleVoter

explanation

- Examine ROLE_* in Object Definition Source and Authentication object
- Grant access if match found, for example:
 - User: Supervisor (supervisor=supervisor, ROLE_USER,ROLE_SUPERVISOR)
 - Request for: approve.htm (/approve.htm=ROLE_SUPERVISOR)
- Currently has 2 voter implementation:
 - RoleVoter
 - BasicAclEntryVoter

Conclusion

- ❑ Acegi security enables declarative security
- ❑ No more coding security control flow
- ❑ No more coding interceptor

Next ...

- ❑ Implement method invocation interceptor
- ❑ Implement business object access control
- ❑ Implement CAS single sign on
- ❑ Implement https connection
- ❑ Integrate with JAAS

Thank you



Endy Muhardin

<http://endy.artivisi.com>

last updated : 20050521