SPRING FRAMEWORK

Spring Security 3.0

Application security

- Security is arguably one of the most critical architectural components of any application written in the 21st century

What is Spring Security

- a powerful and highly customizable authentication and access-control framework
- build on top of Spring Framework
- de-facto standard for securing Spring-based applications

Fundamentals (1)

- principal
  - user that performs the action
- authentication
  - confirming truth of credentials
- authorization
  - define access policy for principal

Fundamentals (2)

- Authentication
  - the principal in a Spring Security-specific manner
- GrantedAuthority
  - application-wide permissions granted to a principal
- SecurityContext
  - hold the Authentication and other security information
- SecurityContextHolder
  - provide access to SecurityContext

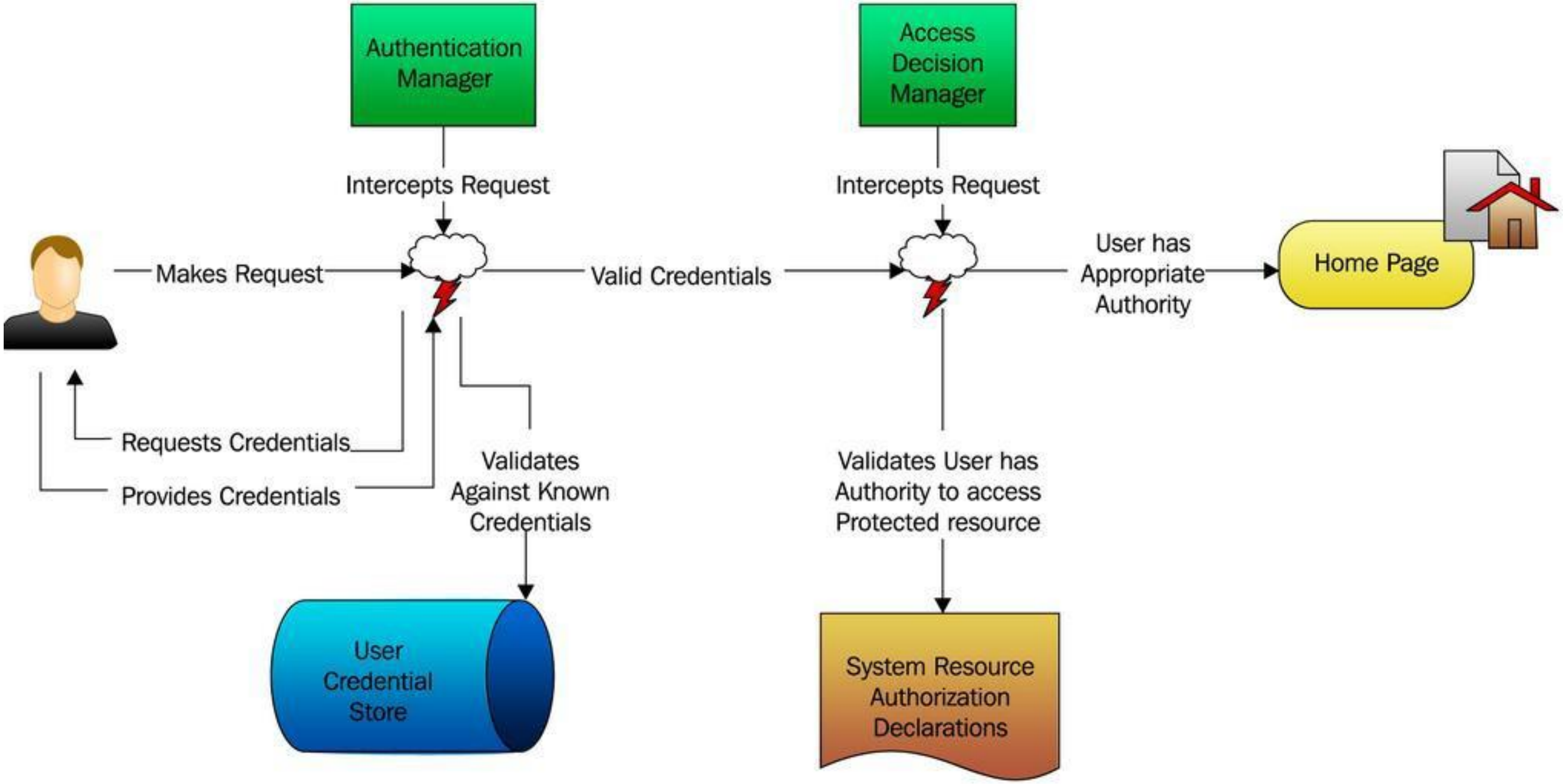
SecurityContextHolder

- provide access to SecurityContext
- strategies
  - ThreadLocal
  - InreritableThreadLocal
  - Global

Getting started

```java
SecurityContext context = SecurityContextHolder.getContext();
Object principal = context.getAuthentication().getPrincipal();

if (principal instanceof UserDetails) {
  String username = ((UserDetails)principal).getUsername();
} else {
  String username = principal.toString();
}
```

Use case


Authentication Manager
Access Decision Manager
Intercepts Request
Intercepts Request
Makes Request
Valid Credentials
User has Appropriate Authority
Home Page
Requests Credentials
Provides Credentials
Validates Against Known Credentials
Validates User has Authority to access Protected resource
User Credential Store
System Resource Authorization Declarations

Namespace

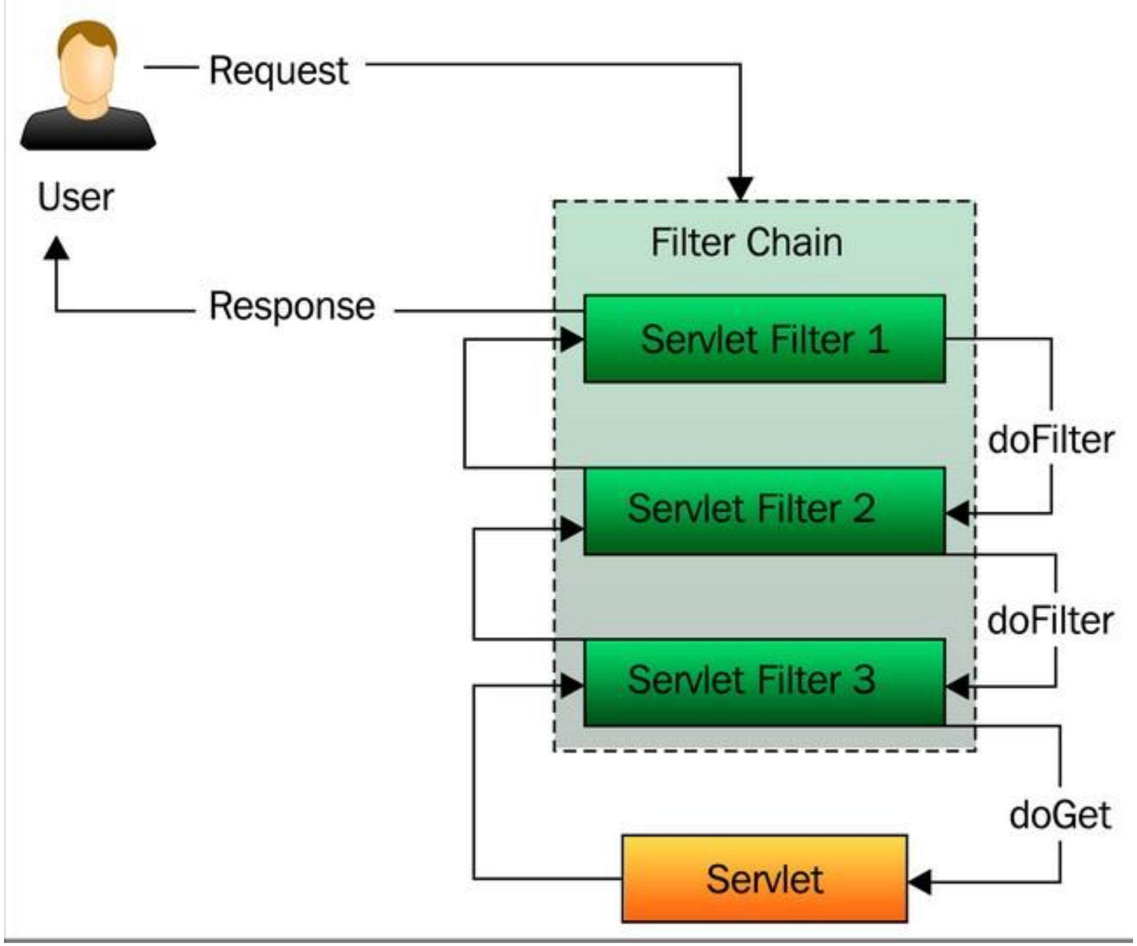
```
<beans  xmlns="http://www.springframework.org/schema/beans"
        xmlns:sec="http://www.springframework.org/schema/security"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security-3.0.xsd">
```

Filters

Security filter

```
<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>
```

Filter chain



Filter chain (2)

```
<bean id="springSecurityFilterChain"
      class="org.springframework.security.web.FilterChainProxy">
  <sec:filter-chain-map path-type="ant">
    <sec:filter-chain pattern="/login.do*" filters="none"/>
    <sec:filter-chain pattern="/**.do*"
                      filters="
                        securityContextPersistenceFilter,
                        logoutFilter,
                        usernamePasswordAuthenticationFilter,
                        rememberMeAuthenticationFilter,
                        exceptionTranslationFilter,
                        filterSecurityInterceptor" />
  </sec:filter-chain-map>
</bean>
```

Basic filters

Filter	Description
ChannelProcessingFilter	ensures that a request is being sent over HTTP or HTTPS
SecurityContextPersistentFilter	Populates the security context using information obtained from the repository (http session)
LogoutFilter	Used to log a user out of the application
UsernamePasswordAuthenticationFilter	Accepts the user's principal and credentials and attempts to authenticate the user
BasicAuthenticationFilter	Attempts to authenticate a user by processing an HTTP Basic authentication
ExceptionTranslationFilter	Handles any AccessDeniedException or AuthenticationException
FilterSecurityInterceptor	Decides whether or not to allow access to a secured resource

http://static.springsource.org/spring-security/site/docs/3.0.x/reference/ns-config.html#ns-custom-filters

Authentication

Authentication variants

- credential-based
- two-factor
- hardware
- other…

Authentication mechanisms

- basic
- form
- x.509
- JAAS
- etc.

Authentication storage

- RDMBS
- LDAP
- custom storage
- etc.

Fundamentals

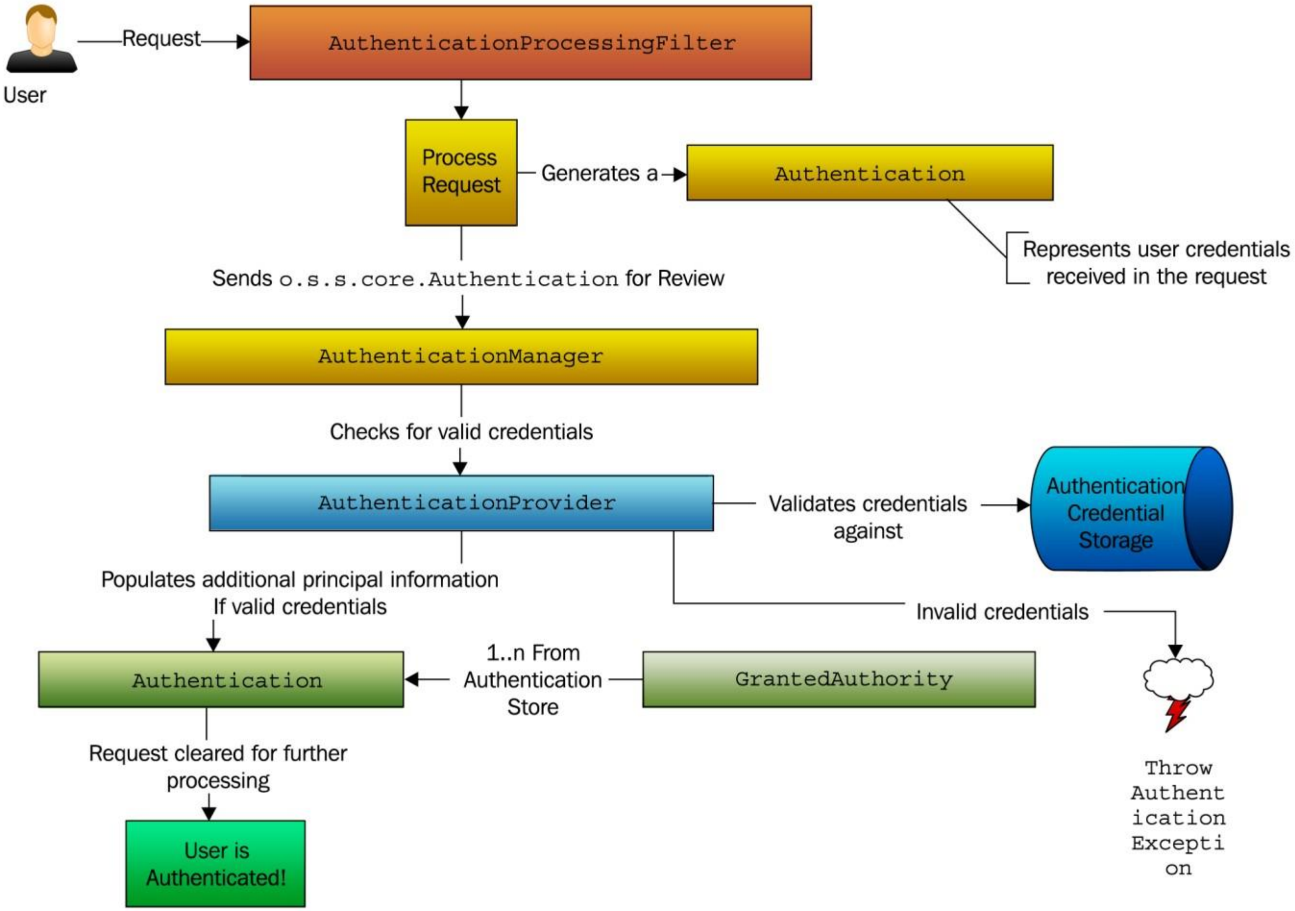
Filter

Manager

Provider

Authentication

UserDetails


User
Request
AuthenticationProcessingFilter
Process Request
Generates a
Authentication
Represents user credentials received in the request
Sends o.s.s.core.Authentication for Review
AuthenticationManager
Checks for valid credentials
AuthenticationProvider
Validates credentials against
Authentication Credential Storage
Populates additional principal information If valid credentials
Invalid credentials
Authentication
1..n From Authentication Store
GrantedAuthority
Throw Authent ication Excepti on
Request cleared for further processing
User is Authenticated!

HTML form

```
<form action="j_spring_security_check" method="post">
  <span>Login:</span><input name="login" type="text"/>
  <span>Password:</span><input name="password" type="password"/>
  <input type="submit" value="Login">
</form>
```

Username-password filter

```
<bean id="..." class="...security.web.authentication.UsernamePasswordAuthenticationFilter">
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="filterProcessesUrl"
  <property value="/j_spring_security_check"/> name="usernameParameter"
  <property value="login"/> name="passwordParameter" value="password"/>
  <property name="authenticationSuccessHandler">
    <bean class="...security.web.authentication.SavedRequestAwareAuthenticationSuccessHandler">
      <property name="defaultTargetUrl" value="/index.do"/>
   </bean>
  </property>
  <property name="authenticationFailureHandler">
    <bean class="...security.web.authentication.SimpleUrlAuthenticationFailureHandler">
      <property name="defaultFailureUrl" value="/login.do"/>
    </bean>
  </property>
  <property name="rememberMeServices" ref="rememberMeService"/>
</bean>
```

Core authentication services

- AuthenticationManager
  - handles authentication requests
- AuthenticationProvider
  - performs authentication
- UserDetailsService
  - responsible for returning an UserDetails object
- UerDetails
  - provides the core user information

AuthenticationManager

```
public interface AuthenticationManager {
  /* Attempts to authenticate the passed Authentication object,
   * returning a fully populated Authentication object (including
   * granted authorities) if successful.
   * @param authentication the authentication request object
   * @return a fully authenticated object including credentials
   * @throws AuthenticationException if authentication fails */
  Authentication authenticate(Authentication authentication)
        throws AuthenticationException;
}
```

AuthenticationProvider

```
public interface AuthenticationProvider {
  /* Performs authentication.
   * @param authentication the authentication request object.
   * @return a fully authenticated object including credentials.
   * @throws AuthenticationException if authentication fails.*/
  Authentication authenticate(Authentication authentication)
       throws AuthenticationException;

  /*Returns true if this provider supports the indicated
   *Authentication object.*/
  boolean supports(Class<? extends Object> authentication);
}
```

UserDetailsService

```
/*Core interface which loads user-specific data.*/
public interface UserDetailsService {
  /* Locates the user based on the username.
   * @param username the username identifying the user
   * @return a fully populated user record (never null)
   * @throws UsernameNotFoundException if the user could not be
     found or the user has no GrantedAuthority
   * @throws DataAccessException if user could not be found for a
     repository-specific reason*/
  UserDetails loadUserByUsername(String username)
     throws UsernameNotFoundException, DataAccessException;
}
```

UserDetails

```
/* Provides core user information.*/
public interface UserDetails extends Serializable {

  Collection<GrantedAuthority> getAuthorities();
  String getPassword();
  String getUsername();

  boolean isAccountNonExpired();
  boolean isAccountNonLocked();
  boolean isCredentialsNonExpired();
  boolean isEnabled();
}
```

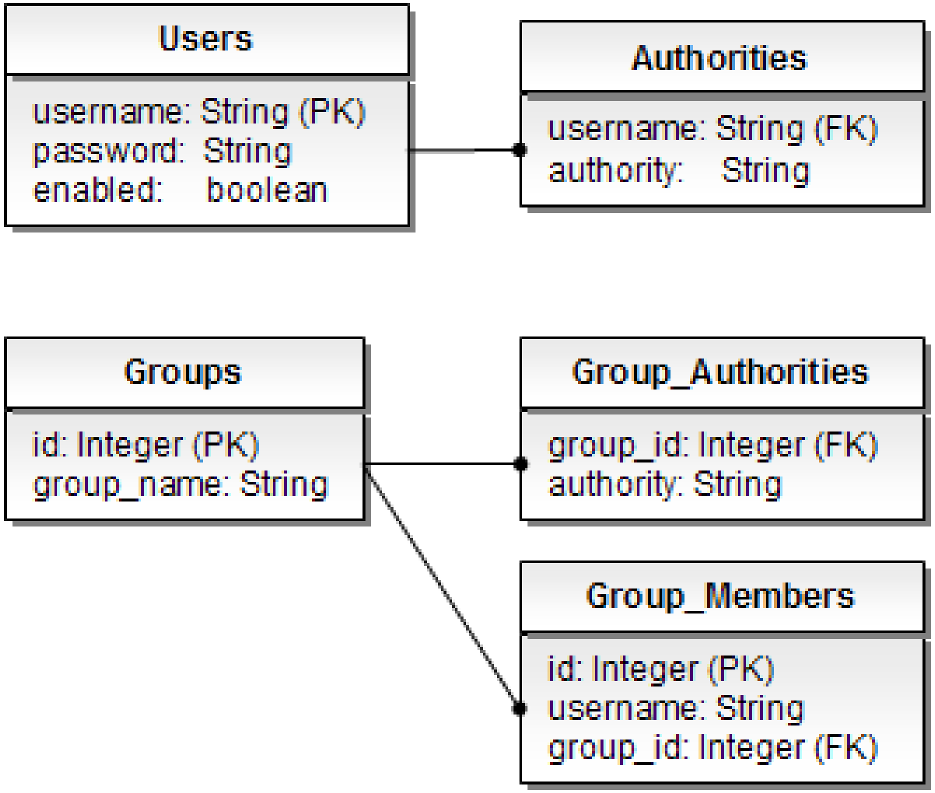
Authentication manager

```
<bean id="..." class="...security.authentication.ProviderManager">
  <property name="providers">
    <list>
      <ref local="casAuthenticationProvider"/>
      <ref local="daoAuthenticationProvider"/>
      <ref local="ldapAuthenticationProvider"/>
    </list>
  </property>
</bean>
```

Authentication provider

```
<bean id="daoAuthenticationProvider"
      class="org.springframework.security.authentication.dao.DaoAuthenticationProvider">
  <property name="userDetailsService" ref="userDetailsService"/>
  <property name="saltSource" ref bean="saltSource"/>
  <property name="passwordEncoder" ref="passwordEncoder"/>
</bean>

<bean id="userDetailsService"
      class="org.springframework.security.core.userdetails.jdbc.JdbcDaoImpl">
  <property name="dataSource" ref="dataSource"/>
</bean>
```

Authentication DB schema



Users

username: String (PK)
password: String
enabled: boolean

Authorities

username: String (FK)
authority: String

Groups

id: Integer (PK)
group_name: String

Group_Authorities

group_id: Integer (FK)
authority: String

Group_Members

id: Integer (PK)
username: String
group_id: Integer (FK)

Password encoding

- PasswordEncoder
  - MD5
  - SHA
- SaltSource
  - SystemWide
  - reflection

Session management

```
<bean id="sessionManagementFilter"
      class="org.springframework.security.web.session.SessionManagementFilter">
  <property name="invalidSessionUrl" value="/timeout.do"/>
  <property name="sessionAuthenticationStrategy" ref="strategy"/>
</bean>

<bean id="strategy"
      class="…SessionFixationProtectionStrategy">
  <property name="alwaysCreateSession" value="true"/>
  <property name="migrateSessionAttributes" value="true"/>
</bean>
```

Logout

```
<bean id="logoutFilter"
      class="org.springframework.security.web.authentication.logout.LogoutFilter">
  <constructor-arg>
    <bean class="…SimpleUrlLogoutSuccessHandler">
      <property name="defaultTargetUrl" value="/login"/>
    </bean>
  </constructor-arg>
  <constructor-arg>
    <bean class="…SecurityContextLogoutHandler"/>
  </constructor-arg>
  <property name="filterProcessesUrl" value="/logout"/>
</bean>
```

Remember Me authentication

□ RememberMeAuthenticationFilter

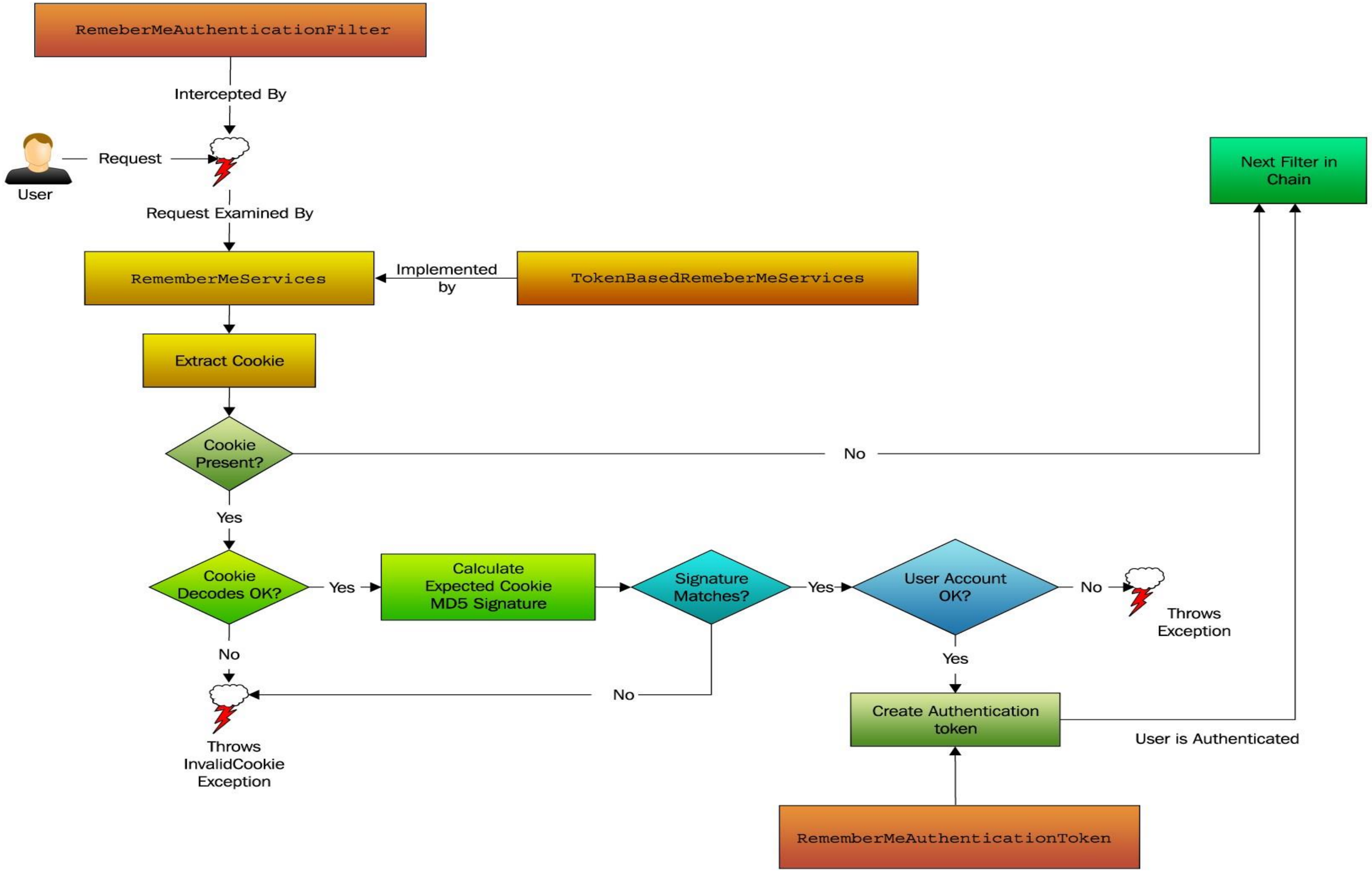
□ RememberMeServices

□ RememberMeAuthenticationProvider

RememberMe service

```
public interface RememberMeServices {

  Authentication autoLogin(HttpServletRequest request,
                           HttpServletResponse response);

  void loginFail(HttpServletRequest request,
                 HttpServletResponse response);

  void loginSuccess(HttpServletRequest request,
                    HttpServletResponse response,
                    Authentication successfulAuthentication);
}
```

Remember Me shema



Anonymous authentication

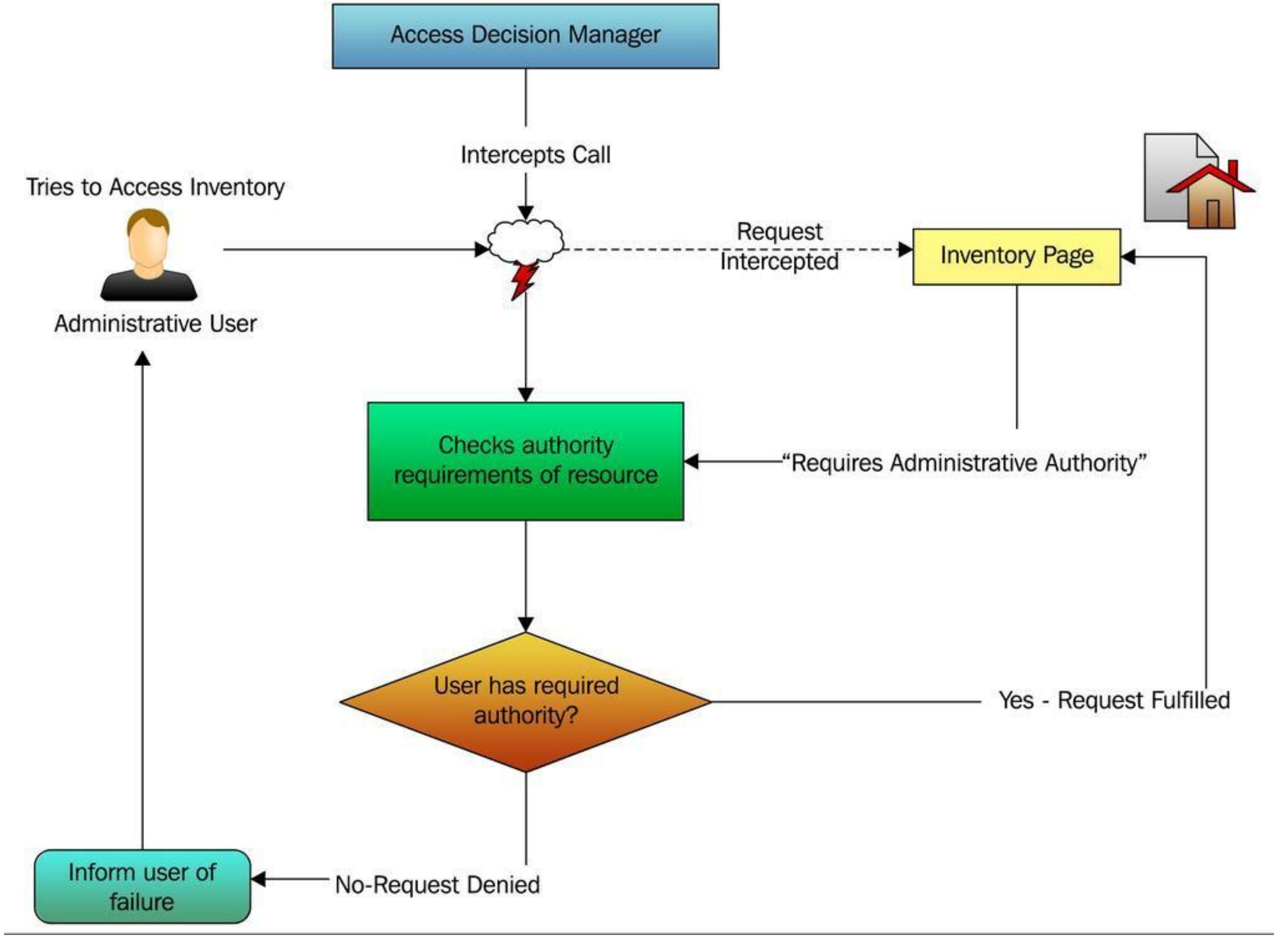
```
<bean id="anonymousAuthenticationFilter"
      class="...web.authentication.AnonymousAuthenticationFilter">
  <property name="key" value="foobar"/>
  <property name="userAttribute" value="anonymous,ROLE_ANONYMOUS"/>
</bean>

<bean id="anonymousAuthenticationProvider"
      class="...authentication.AnonymousAuthenticationProvider">
  <property name="key" value="foobar"/>
</bean>
```

Authentication with magic tags

```
<sec:http auto-config="true">
  <sec:form-login login-page="" login-processing-url=""/>
  <sec:anonymous enabled="true"/>
  <sec:logout invalidate-session="true" logout-url=""/>
  <sec:remember-me services-ref=""/>
</sec:http>
```

Authorization

Use case


Access Decision Manager
Intercepts Call
Tries to Access Inventory
Administrative User
Request
Intercepted
Inventory Page
Checks authority requirements of resource
"Requires Administrative Authority"
User has required authority?
Yes - Request Fulfilled
No-Request Denied
Inform user of failure

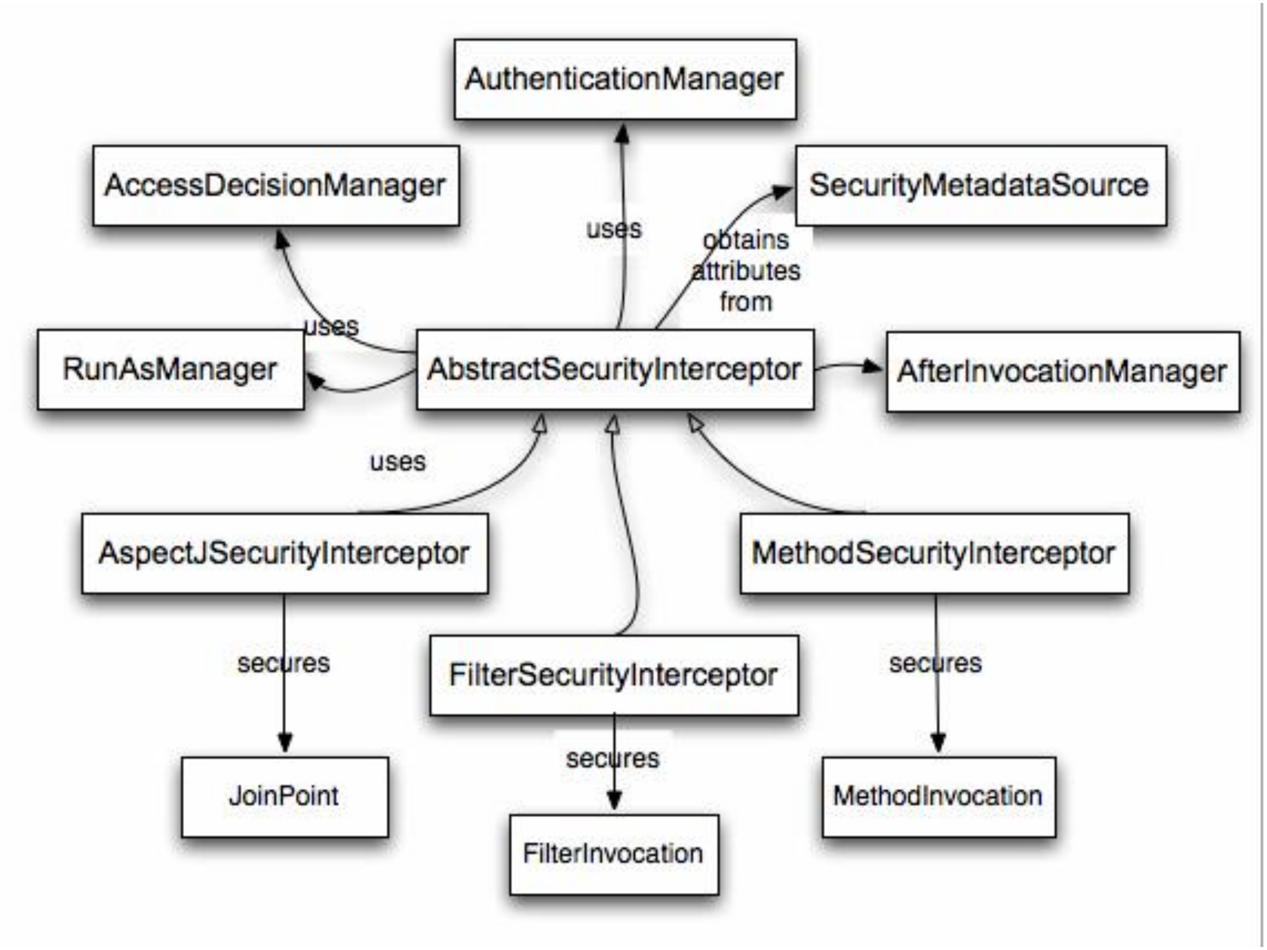
Authorization

- □ handling
  - ➢ pre-invocation
  - ➢ after invocation
- □ implementations
  - ➢ voting based
  - ➢ expression based

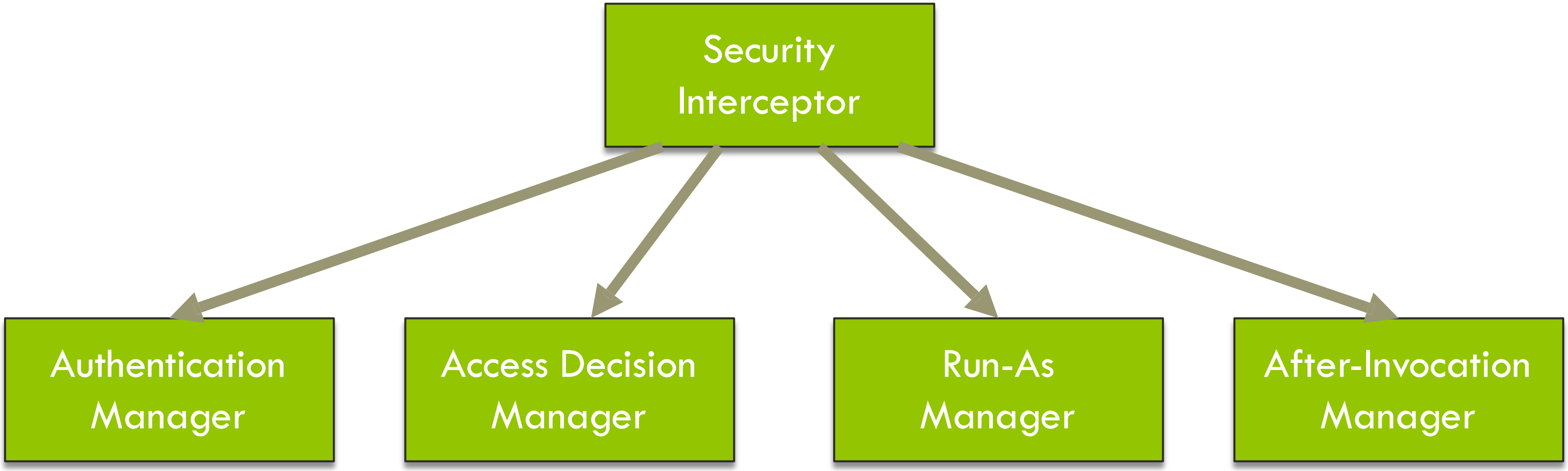
Security layers

- WEB (URLs)
  - Servlet Filter
- methods
  - Spring AOP
  - AspectJ
- content
  - JSP tag

Security interceptor (1)



Security interceptor (2)



Security Interceptor

Authentication Manager

Access Decision Manager

Run-As Manager

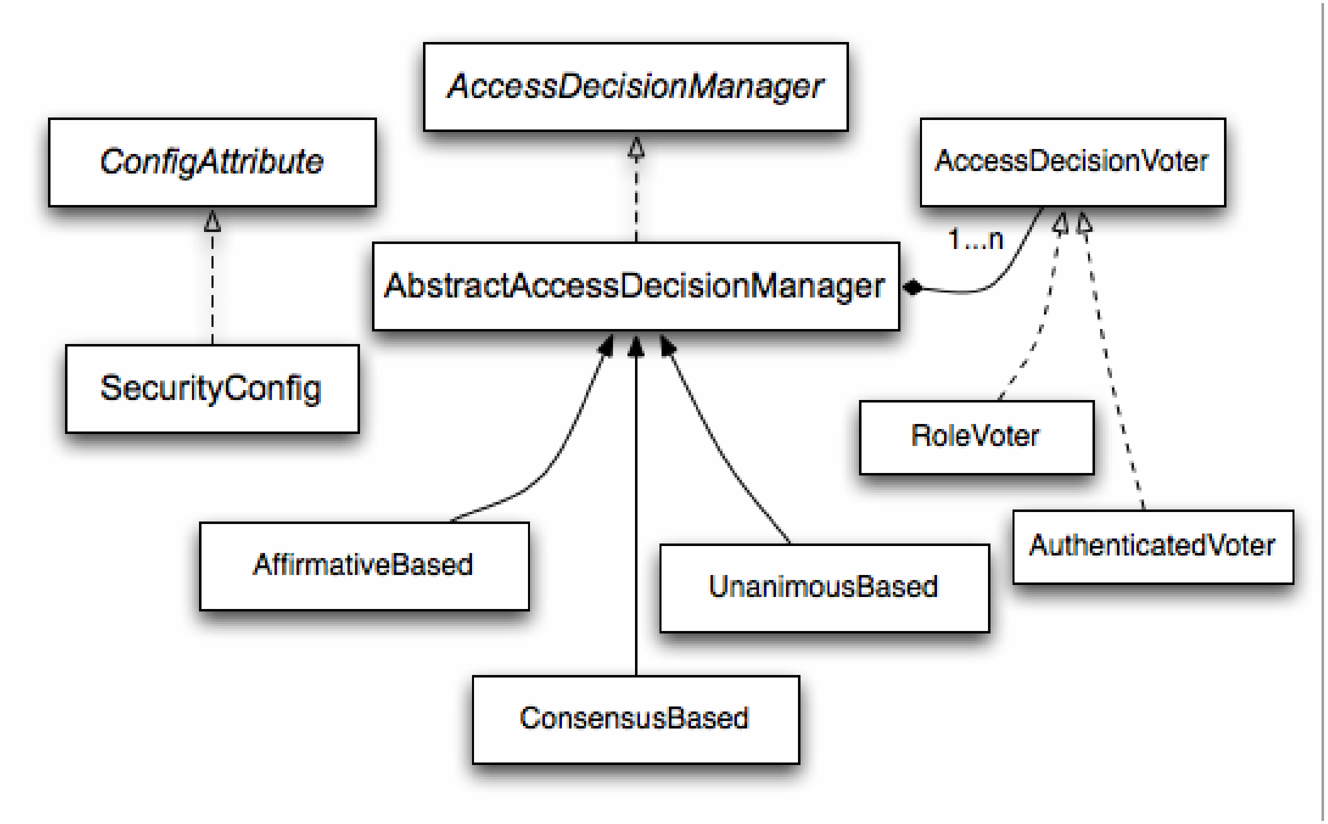
After-Invocation Manager

Voting based

DecisionManager

DecisionVoter

ConfigAttribute



Decision managers

Decision manager	Description
AffirmativeBased	Allows access if at least one voter votes to grant access
ConsensusBased	Allows access if a consensus of voters vote to grant access
UnanimousBased	Allows access if all voters vote to grant access

Decision voter

```
public interface AccessDecisionVoter {
  int ACCESS_GRANTED = 1;
  int ACCESS_ABSTAIN = 0;
  int ACCESS_DENIED = -1;

  boolean supports(ConfigAttribute attribute);

  boolean supports(Class<?> clazz);

  int vote(Authentication authentication,
           Object object,
           Collection<ConfigAttribute> attributes);
}
```

Basic expressions

Expression	Description
hasRole(‘ROLE_USER’)	Returns true if the current principal has the specified role
hasAnyRole(‘ROLE_USER’, ‘ROLE_ADMIN’)	Returns true if the current principal has any of the roles
principal	Allows direct access to the principal object representing the current user
authentication	Allows direct access to the current Authentication object obtained from the SecurityContext
permitAll	Always evaluates to true
denyAll	Always evaluates to false
isAnonymous()	Returns true if the current principal is an anonymous user
isRememberMe()	Returns true if the current principal is a remember-me user

WEB authorization

Web authorization

```
<bean id="..." class="web.access.intercept.FilterSecurityInterceptor">
  <property name="authenticationManager" ref="authManager"/>
  <property name="accessDecisionManager" ref="decisionManager"/>
  <property name="securityMetadataSource">
    <sec:filter-security-metadata-source>
      <sec:intercept-url pattern="/index.do*"
                         access="IS_AUTHENTICATED_FULLY"/>
      <sec:intercept-url pattern="/**"
                         access="ROLE_USER"
                         filters="none"
                         method="GET"
                         requires-channel="https"/>
    </sec:filter-security-metadata-source>
  </property>
</bean>
```

WEB authorization with magic tags

```
<sec:http use-expressions="true">
  <sec:intercept-url pattern="/index*"
                     access="isAuthenticated()"/>

  <sec:intercept-url pattern="/**"
                     access="hasRole('ROLE_USER')"
                     filters="none"
                     method="GET"
                     requires-channel="https"/>
</sec:http>
```

WEB authorization

```
<bean id="webExpressionHandler"
      class="…DefaultWebSecurityExpressionHandler"/>

<bean id="webExpressionVoter" class="…WebExpressionVoter">
  <property name="expressionHandler" ref="webExpressionHandler"/>
</bean>

<bean class="org.springframework.security.access.vote.AffirmativeBased">
  <property name="decisionVoters">
     <list>
      <ref bean="webExpressionVoter"/>
     </list>
  </property>
</bean>
```

Custom expression root

```
public class CustomWebSecurityExpressionRoot
                              extends WebSecurityExpressionRoot {

  public CustomWebSecurityExpressionRoot(Authentication a,
                                         FilterInvocation fi) {
    super(a, fi);
  }

  public boolean hasAllRoles(String... roles) {
    return false;
  }
}
```

Custom expression handler

```
public class CustomWebSecurityExpressionHandler
                        extends DefaultWebSecurityExpressionHandler {

  @Override
  public EvaluationContext createEvaluationContext(Authentication a,
                                                FilterInvocation fi) {
    StandardEvaluationContext ctx =
     (StandardEvaluationContext)super.createEvaluationContext(a, fi);
    SecurityExpressionRoot root =
         new CustomWebSecurityExpressionRoot(a, fi);
    ctx.setRootObject(root);
    return ctx;
  }
}
```

Method authorization

Method authorization

- annotation driven
  - voting based - `@Secured`
  - expression based - `@Pre/@Post`
  - JSR-250 - `@RolesAllowed`
- xml driven

Configuration

```xml
<sec:global-method-security>
  access-decision-manager-ref="accessDecisionManager" jsr250-
  annotations="disabled"
  pre-post-annotations="disabled"
  secured-annotations="enabled"
</sec:global-method-security>
```

Annotation driven (voting)

- voting

```
@Secured({"ROLE_USER"})
void create(Customer customer);
```

- jsr-250

```
@RolesAllowed({"ROLE_USER"})
void create(Customer customer);
```

Annotation driven (expression)

- 1

```
@PreAuthorize("hasRole('ROLE_USER')")
void create(Customer customer);
```

- 2

```
@PreAuthorize("hasRole('ROLE_USER') and hasRole('ROLE_ADMIN')")
void create(Customer customer);
```

- 3

```
@PreAuthorize("hasAnyRole('ROLE_USER', 'ROLE_ADMIN')")
void create(Customer customer);
```

XML driven authorization (1)

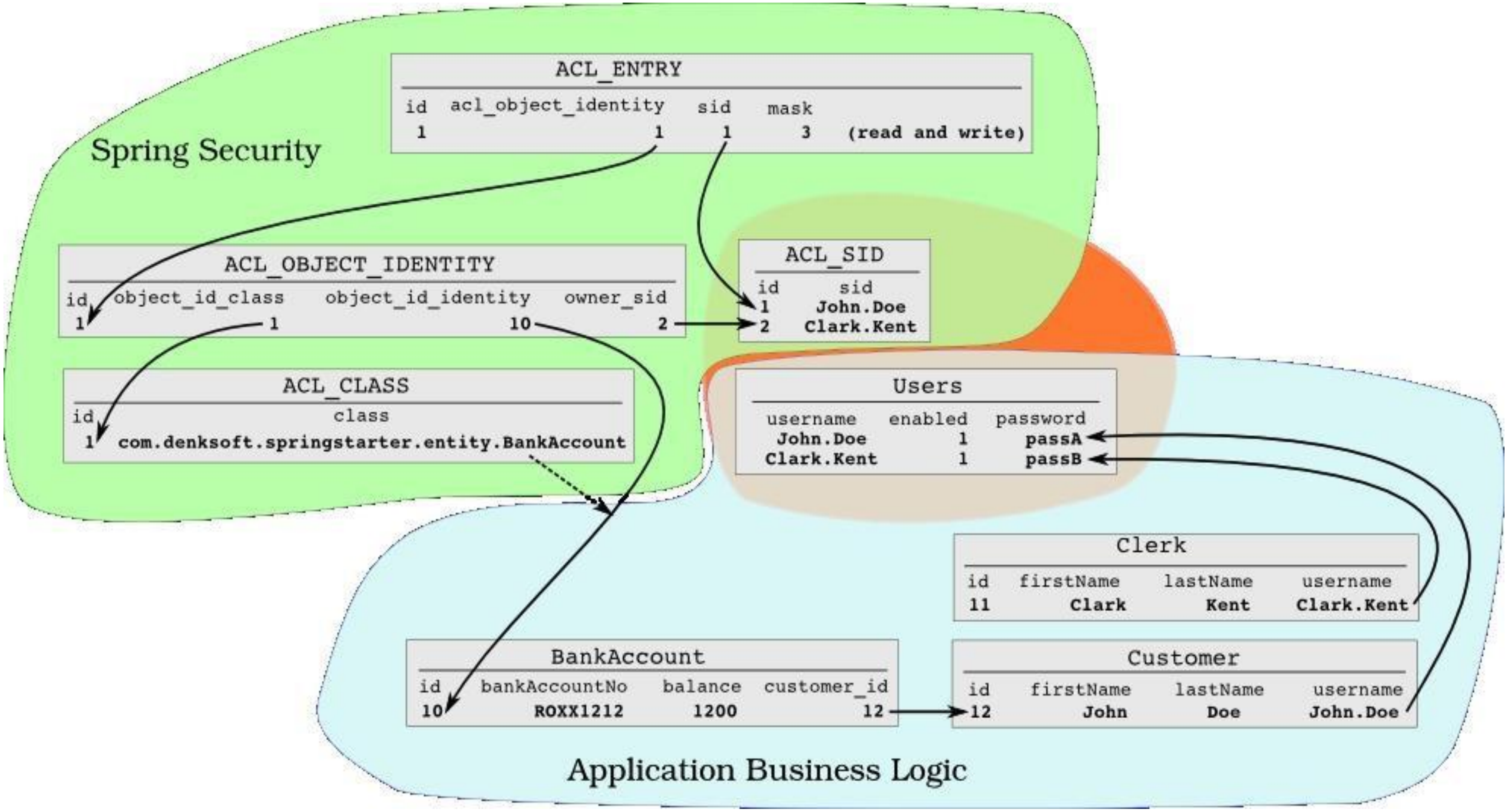
```
<bean id="methodInterceptor" class="…MethodSecurityInterceptor">
  <property name="authenticationManager" ref="authManager"/>
  <property name="accessDecisionManager" ref="decisionManager"/>
  <property name="securityMetadataSource">
    <value>
      org.training.AccountService.createAccount=ROLE_USER
      org.training.AccountService.delete*=ROLE_ADMIN
    </value>
  </property>
</bean>
```

XML driven authorization (2)

```
<bean id="accountService"
      class="org.training.AccountServiceImpl">
  <sec:intercept-methods>
    <sec:protect access="ROLE_USER" method="createAccount"/>
    <sec:protect access="ROLE_ADMIN" method="delete*"/>
  </sec:intercept-methods>
</bean>
```

Domain Object Security

ACL DB scheme



ACL_ENTRY

id	acl_object_identity	sid	mask	
1	1	1	3	(read and write)

ACL_OBJECT_IDENTITY

id	object_id_class	object_id_identity	owner_sid
1	1	10	2

ACL_SID

id	sid
1	John.Doe
2	Clark.Kent

ACL_CLASS

id	class
1	com.denksoft.springstarter.entity.BankAccount

Users

username	enabled	password
John.Doe	1	passA
Clark.Kent	1	passB

Clerk

id	firstName	lastName	username
11	Clark	Kent	Clark.Kent

BankAccount

id	bankAccountNo	balance	customer_id
10	ROXX1212	1200	12

Customer

id	firstName	lastName	username
12	John	Doe	John.Doe

- ACL_CLASS
- ACL_SID
- ACL_OBJECT_IDENTITY
- ACL_ENTRY

Basic classes

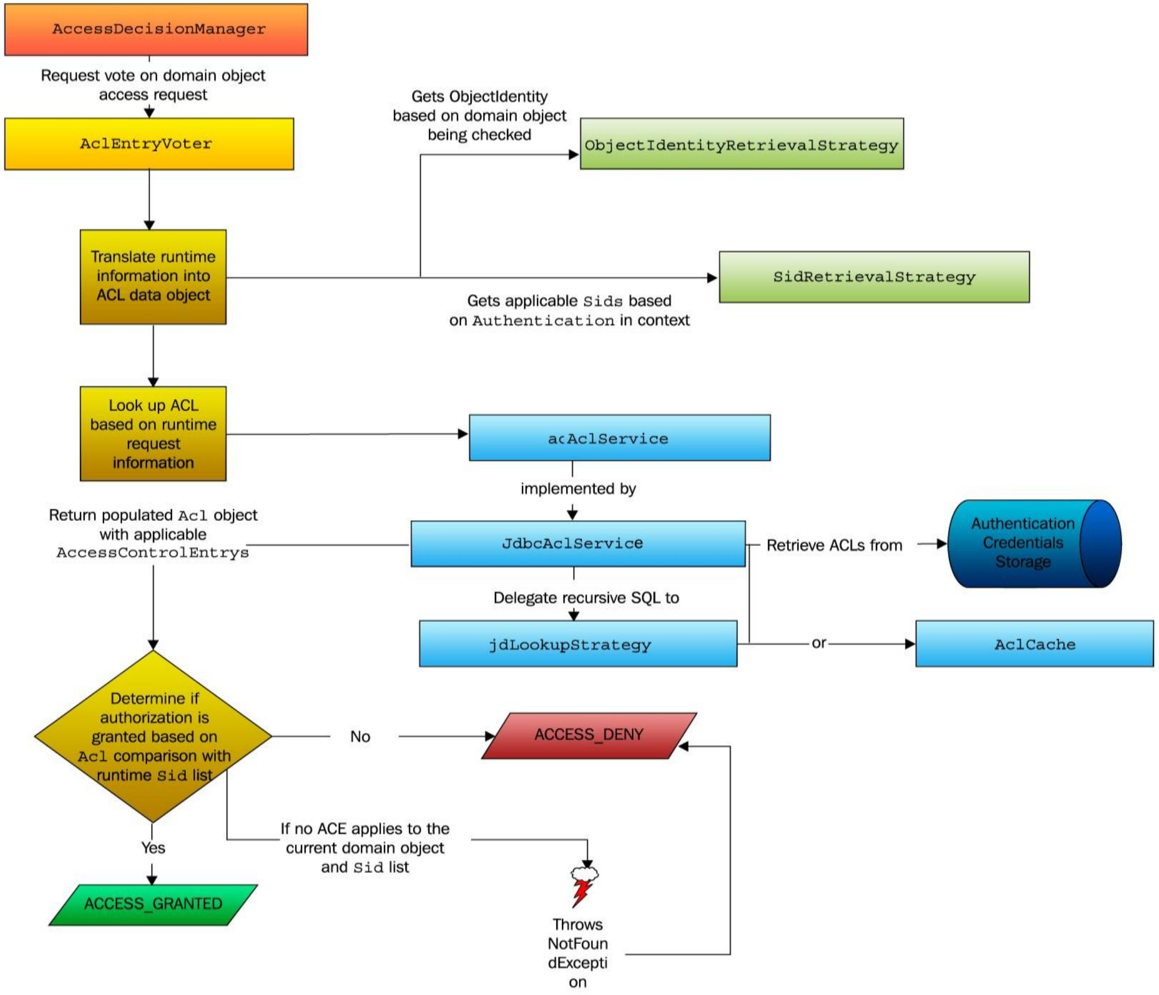
- Acl
- AccessControlEntry
- Permission
- Sid
- ObjectIdentity
  - represents the identity of an individual domain object

Basic ACL services

- AclService
- MutableAclService
- LookupStrategy
- ObjectIdentityRetrievalStrategy
- SidRetrievalStrategy

Permissions

- base permissions
  - read (1)
  - write (2)
  - create (4)
  - delete (8)
  - administration (16)
- custom permissions


AccessDecisionManager
Request vote on domain object access request
AclEntryVoter
Gets ObjectIdentity based on domain object being checked
ObjectIdentityRetrievalStrategy
Translate runtime information into ACL data object
SidRetrievalStrategy
Gets applicable Sids based on Authentication in context
Look up ACL based on runtime request information
acAclService
implemented by
Return populated Acl object with applicable AccessControlEntrys
JdbcAclService
Retrieve ACLs from
Authentication Credentials Storage
Delegate recursive SQL to
jdLookupStrategy
or
AclCache
Determine if authorization is granted based on Acl comparison with runtime Sid list
No
ACCESS_DENY
If no ACE applies to the current domain object and Sid list
Yes
ACCESS_GRANTED
Throws NotFoundException

Configuration (voting)

```
<sec:global-method-security
  access-decision-manager-ref="accessDecisionManager"
  secured-annotations="enabled">
</sec:global-method-security>

<bean id="accessDecisionManager" class="…AffirmativeBased">
  <property name="decisionVoters">
    <list>
      <ref bean="voter1"/>
      <ref bean="voter2"/>
    </list>
  </property>
</bean>
```

@Secured

□ annotation

```
@Secured("ACL_CUSTOMER_READ")
public Customer getProjectsByCustomer(Customer customer) {}
```

□ voter

```
<bean id="customerReadVoter" class="…AclEntryVoter">
  <constructor-arg ref="aclService"/>
  <constructor-arg value="ACL_CUSTOMER_READ"/>
  <constructor-arg>
    <array>
      <util:constant static-field="…BasePermission.READ"/>
    </array>
  </constructor-arg>
  <property name="processDomainObjectClass" value="…Customer"/>
</bean>
```

Configuration (expressions)

```
<sec:global-method-security pre-post-annotations="enabled">
  <sec:expression-handler ref="expressionHandler"/>
</sec:global-method-security>

<bean id="expressionHandler"
      class="…DefaultMethodSecurityExpressionHandler">
  <property name="permissionEvaluator" ref="permissionEvaluator"/>
</bean>

<bean id="permissionEvaluator" class="…AclPermissionEvaluator">
  <constructor-arg ref="aclService"/>
</bean>
```

Permission evaluator

```java
public interface PermissionEvaluator {

  boolean hasPermission(Authentication authentication,
                        Object targetDomainObject,
                        Object permission);

  boolean hasPermission(Authentication authentication,
                        Serializable targetId,
                        String targetType,
                        Object permission);
}
```

@PreAuthorize

□ by domain object

```
@PreAuthorize("hasPermission(#customer, 'delete')")
public void delete(Customer customer);
```

□ by identifier

```
@PreAuthorize(
  "hasPermission(#id, 'org.training.Customer', 'read') or " +
  "hasPermission(#id, 'org.training.Customer', 'admin')")
public Customer getById(Long id);
```

~~□ hardcode~~

```
@PreAuthorize("#customer.owner.id == principal.id")
public void create(Customer customer);
```

@PreFilter

- single parameter

```
@PreFilter("hasPermission(filterObject, 'read')")
public List<Customer> filterCustomers(List<Customer> customers) {
  return customers;
}
```

- multiple parameters

```
@PreFilter(filterTarget = "customers",
           value = "hasPermission(filterObject, 'update')")
public void updateCustomers(List<Customer> customers, State st) {
}
```

Additional features

RunAsManager

```
/*Creates a new temporary Authentication object.*/
public interface RunAsManager {

  / *Returns a replacement Authentication object for the current
    *secure object, or null if replacement not required*/
  Authentication buildRunAs(Authentication authentication,
                            Object object,
                            Collection<ConfigAttribute> attr);

  boolean supports(ConfigAttribute attribute);

  boolean supports(Class<?> clazz);
}
```

RunAs configuration (1)

```
<bean id="runAsManager" class="…RunAsManagerImpl">
  <property name="rolePrefix" value="ROLE_"/>
  <property name="key" value="someKey"/>
</bean>

<bean class="…RunAsImplAuthenticationProvider">
  <property name="key" value="someKey"/>
</bean>
```

RunAs configuration (2)

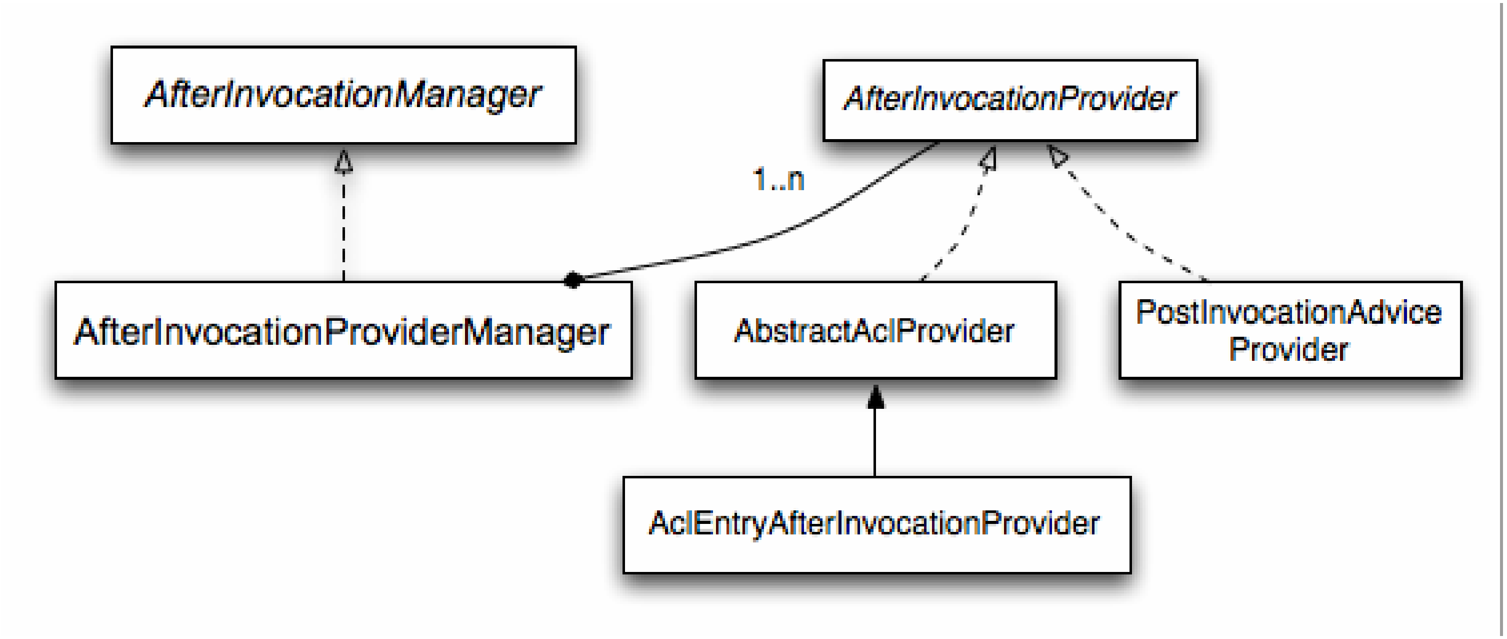
- magic tag

```
<sec:global-method-security run-as-manager-ref="runAsManager">
</sec:global-method-security>
```

- interceptor bean

```
<bean class="…MethodSecurityInterceptor">
  <property name="runAsManager" ref="runAsManager"/>
</bean>
```

After invocation


AfterInvocationManager
AfterInvocationProvider
1..n
AfterInvocationProviderManager
AbstractAclProvider
PostInvocationAdvice Provider
AclEntryAfterInvocationProvider

Basic services

```
public interface AfterInvocationManager {
  Object decide(Authentication authentication, Object object,
                Collection<ConfigAttribute> attributes,
                Object returnedObject) throws AccessDeniedException;

  boolean supports(ConfigAttribute attribute);
  boolean supports(Class<?> clazz);
}
```

```
public interface AfterInvocationProvider {
  Object decide(Authentication authentication, Object object,
                Collection<ConfigAttribute> attributes,
                Object returnedObject) throws AccessDeniedException;

  boolean supports(ConfigAttribute attribute);
  boolean supports(Class<?> clazz);
}
```

Configuration

- custom provider

```
<sec:global-method-security>
  <sec:after-invocation-provider ref="myProvider"/>
</sec:global-method-security>
```

- custom manager

```
<bean class="…MethodSecurityInterceptor">
  <property name="afterInvocationManager" ref="myManager"/>
</bean>
```

@Post

- @PostAuthorize

```
@PreAuthorize("hasRole('ROLE_USER')")
@PostAuthorize("hasPermission(returnObject, 'read')")
public Employee getEmployeeByName(String name) {
}
```

- @PostFilter

```
@PreAuthorize("hasRole('ROLE_USER')")
@PostFilter("hasPermission(filterObject, 'read')")
public List<Employee> getEmployees() {
}
```

JSP tag library

Authentication

```
<%@ taglib prefix="sec"
           uri="http://www.springframework.org/security/tags" %>

<sec:authentication property="principal" var="user"/>
<div class="links"><div>Logged in: ${user.name}</div></div>
```

```
<div class="links">
  <div><sec:authentication property="principal.name"/></div>
</div>
```

Authorize (1)

```
<%@ taglib prefix="sec"
                uri="http://www.springframework.org/security/tags" %>

<sec:authorize ifAllGranted="ROLE_ADMIN, ROLE_SUPERVISOR">
</sec:authorize>

<security:authorize ifAnyGranted="ROLE_ADMIN, ROLE_SUPERVISOR">
</security:authorize>

<security:authorize ifNotGranted="ROLE_ADMIN, ROLE_SUPERVISOR">
</security:authorize>
```

Authorize (2)

```
<%@ taglib prefix="sec"
           uri="http://www.springframework.org/security/tags" %>

<sec:authorize access="hasRole('supervisor')">
  This content will only be visible to users who have
  the "supervisor" authority in their list of
  <tt>GrantedAuthority</tt>s.
</sec:authorize>
```

Authorize (3)

- JSP

```
<sec:authorize url="/admin"  >
  This content will only be visible to users who are authorized
  to send requests to the "/admin" URL.
</sec:authorize>
```

- security interceptor

```
<bean id="..." class="web.access.intercept.FilterSecurityInterceptor">
  <property name="securityMetadataSource">
    <sec:filter-security-metadata-source>
      <sec:intercept-url pattern="/admin*" access="ROLE_ADMIN"/>
    </sec:filter-security-metadata-source>
  </property>
</bean>
```

ACL

```
<%@ taglib prefix="sec"
           uri="http://www.springframework.org/security/tags" %>

<sec:accesscontrollist hasPermission="1,2" domainObject="object">
  This will be shown if the user has either of the permissions
  represented by the values "1" or "2" on the given object.
</sec:accesscontrollist>
```

Summary

Separation of concerns

- business logic is decoupled from security concern
- authentication and authorization are decoupled

Flexibility

- authentication mechanisms
  - basic, form, cookies, SSO
- user data storage
  - RDBMS, LDAP, etc.
- based on Spring

Portability

- portable across containers
- can be deployed as-is
- runs in standalone environment

Links

- □ main features

  http://static.springsource.org/spring-security/site/features.html

- □ articles

  http://static.springsource.org/spring-security/site/articles.html

- □ reference

  http://static.springsource.org/spring-security/site/docs/3.0.x/reference/springsecurity.html

- □ blog

  http://blog.springsource.com/category/security/

- □ refcardz

  http://refcardz.dzone.com/refcardz/expression-based-authorization

Questions

The end