

Keep it simple

Spring Security in der Praxis

ex|Xcellent
solutions

Martin Renner
OOP 2009



- x| Überblick
- x| Ausflug zum Spring Framework
- x| Zentrale API Komponenten
- x| Authentifizierung
- x| Konfiguration
- x| Autorisierung
- x| Fazit

x| Überblick

Ausflug zum Spring Framework

Zentrale API Komponenten

Authentifizierung

Konfiguration

Autorisierung

Fazit

- x| Java SE / Java EE Framework für Authentifizierung und Autorisierung
- x| Authentifizierung: *Wer oder was interagiert mit dem System?*
- x| Autorisierung: *Was darf er oder es?*
- x| Hoch modular mit 83 Interfaces bei 423 Java-Dateien im Core-Paket.
- x| Bietet sich für Applikationen auf Basis des Spring Frameworks an, kann aber auch unabhängig davon benutzt werden.

- x| Entwicklung begann im November 2003.
- x| Im März 2004 auf der Mailingliste „springframework-user“ öffentlich als „Acegi Security System for Spring“ freigegeben ([Quelle](#)).
- x| Name stammt vom Firmennamen des Entwicklers Ben Alex: Acegi Computer Technologies ([Quelle](#)).
- x| Inzwischen in das Portfolio von SpringSource als „Spring Security“ aufgenommen.
- x| Aktuelle Version ist 2.0.x (2.0.4 am 02.10.08)

x| Frontend

- | HTTP Basic Authentication / HTTP Digest Authentication
- | X.509 Client Zertifikate
- | Form Based Login
- | RememberMe
- | JAAS
- | NTLM
- | Container Integration: JBoss, Tomcat, Jetty, Resin

x| Backend

- | JDBC
- | LDAP
- | In Memory
- | OpenID / CA Siteminder / JA-SIG Central Authentication Service

x| Channel Security

- | HTTP auf HTTPS (und vice versa) umleiten

- x|** HTTP Autorisierung über URL-Patterns (Frontend)

- x|** Autorisierung auf Methodenebene (Service Layer)
 - | Annotations (Framework-spezifische und JSR 250 („EJB 3“))
 - | AOP Alliance Interceptor (Proxy-basierter Ansatz)
 - | AspectJ Interceptor (Proxy-basierter Ansatz)

- x|** Domain Object Security (Business Objects)
 - | Access Control Lists (ACLs)
 - | Pro Domain Object eine ACL

Überblick

x| Ausflug zum Spring Framework

Zentrale API Komponenten

Authentifizierung

Konfiguration

Autorisierung

Fazit

x| Bietet unter anderem einen IoC (oder DI) Container

```
<jee:jndi-lookup id="myDataSource"  
                jndi-name="java:comp/env/jdbc/MyDataSource" />  
  
<bean id="myService" class="de.excellent.MyServiceImpl">  
  <property name="dataSource">  
    <ref local="myDataSource" />  
  </property>  
</bean>
```

x| Die definierten Beans werden in einem sog. „Context“ verwaltet

x| Erweiternde Elemente können über Namespaces hinzugefügt werden. Diese Elemente können beliebigen Code ausführen.

Überblick

Ausflug zum Spring Framework

x| Zentrale API Komponenten

Authentifizierung

Konfiguration

Autorisierung

Fazit

x| **GrantedAuthority**

- | Beschreibt ein Recht

x| **Authentication**

- | Beschreibt einen Benutzer oder einen Client, der mit dem System interagiert (nicht im Sinne eines Accounts)
- | Listet die „granted Authorities“ des Benutzers
- | Liefert per „getPrincipal()“ die Identität („Account“) zurück

x| **SecurityContextHolder** hält einen **SecurityContext**

- | Standard ist „ThreadLocal“, daneben auch „InheritableThreadLocal“, „Global“ oder eine benutzerdefinierte Klasse möglich
- | „SecurityContext“ hält das aktuelle „Authentication“ Objekt

x| UserDetails

| Beschreibt einen Benutzer als „Account“

x| UserDetailsService

| Verwaltet (erzeugt, lädt, speichert) „UserDetails“

```
Authentication auth = SecurityContextHolder.getContext()  
                    .getAuthentication()  
UserDetails ud = (UserDetails) auth.getPrincipal()
```

x| ConfigAttribute

| Definiert ein beliebiges Security-Attribut, das von einem „Provider“ ausgewertet werden kann.

Überblick

Ausflug zum Spring Framework

Zentrale API Komponenten

x| Authentifizierung

Konfiguration

Autorisierung

Fazit

- x| AuthenticationManager** (implementiert durch **ProviderManager**)
 - | Benutzt eine Liste von AuthenticationProvidern, um die Authentifizierung durchzuführen.

- x| DaoAuthenticationProvider**
 - | Ist ein solcher Provider, der einen UserDetailsService benutzt.
 - | Kann mit PasswordEncoder (MD5, SHA) und SaltSource umgehen.

- x| UserDetailsService**
 - | JdbcDaoImpl, LdapUserDetailsService und InMemoryDaoImpl sind ein paar dieser Services.

- x| „Ein Authentication Objekt muss in den SecurityContext“

- x| Form Based Login
 - | Trivial: „AuthenticationProcessingFilter“ holt Name und Passwort aus dem Request, führt die Anmeldung beim „AuthenticationManager“ durch und schreibt ein „Authentication“ Objekt in den „SecurityContext“.

- x| HTTP Requests
 - | Ähnlich trivial: Login-Daten aus dem HTTP-Header holen, beim „AuthenticationManager“ anmelden und den „SecurityContext“ befüllen. Im Fehlerfall die „SecurityException“ in einen HTTP Code (401, 403) umwandeln.

Überblick

Ausflug zum Spring Framework

Zentrale API Komponenten

Authentifizierung

x| Konfiguration

Autorisierung

Fazit

3 Möglichkeiten

- x| Per Namespace Konfiguration
- x| Einzelne Komponenten manuell per `<bean>` definieren und per Dependency Injection zusammen fügen
- x| Einzelne Komponenten programmatisch instanziiieren und programmatisch zusammen fügen

```
<beans:beans
  xmlns:beans="http://www.springframework.org/schema/beans"
  xmlns="http://www.springframework.org/schema/security"
  ...>

<http auto-config="true">
  <intercept-url pattern="/app/**" access="ROLE_USER" />
  <intercept-url pattern="/admin/**" access="ROLE_ADMIN" />
</http>
```

- x| Erzeugt u.a. einen Filter für „Form Based Login“, „HTTP Basic Authentication“, „Anonymous Authentication“ und „RememberMe“. Ebenso einen „AffirmativeBased“ „AccessDecisionManager“.
- x| Es fehlt nur noch ein „AuthenticationProvider“ (mit einem „UserDetailsService“)

```
<authentication-provider> <!-- DaoAuthenticationProvider -->
  <user-service> <!-- InMemoryDaoImpl -->
    <user name="user" password="geheim1"
      authorities="ROLE_USER" />
    <user name="admin" password="geheim2"
      authorities="ROLE_USER, ROLE_ADMIN" />
  </user-service>
</authentication-provider>
```

x| Mit `<password-encoder hash="sha" />` kann bspw. ein SHA-Hash anstelle der Klartextpasswörter benutzt werden.

- x| Namespace Konfiguration ist übersichtlich und kompakt.
- x| Dokumentation ist inzwischen (Stand 2.0.4) brauchbar.
- x| Nachteile dieser Namespace Konfiguration
 - | Die Beans werden mit konstanten Namen im Spring-Context abgelegt. Dadurch ist es nicht möglich, zwei unterschiedliche Authentifizierungs-Mechanismen mit unterschiedlichen Parametern zu benutzen.
 - | Es können nicht alle Parameter aller beteiligten Komponenten beeinflusst werden.

- x| Konfiguration über einzelne `<bean>` Elemente
 - | Zum umfangreich und komplex für diesen Vortrag.
 - | Erkläre ich aber gerne, wenn Sie mit einer Flasche Rotwein bei mir vorbeikommen.

- x| Programmatische Konfiguration
 - | Siehe vorheriger Absatz

Überblick

Ausflug zum Spring Framework

Zentrale API Komponenten

Authentifizierung

Konfiguration

x| Autorisierung

Fazit

- x|** `AbstractSecurityInterceptor` mit den Methoden
 - | `beforeInvocation()` --> `AccessDecisionManager`
 - | `afterInvocation()` --> `AfterInvocationManager`

- x|** Konkrete Implementierungen des Interceptors rufen vor und nach dem Aufruf der zu schützenden Methode/URL obige Methoden auf.

- x|** „AffirmativeBased“, „ConsensusBased“ und „UnanimousBased“ sind Implementierungen eines „AccessDecisionManager“. Sie befragen eine konfigurierbare Liste von „AccessDecisionVotern“.

- x|** „RoleVoter“, „AuthenticatedVoter“ und „Jsr250Voter“ sind solche Voter.

- x| „AfterInvocationProviderManager“ ist eine Implementation von „AfterInvocationManager“ und ruft eine Liste von „AfterInvocationProvidern“ auf.
- x| „AclEntryAfterInvocationProvider“ und „AclEntryAfterInvocationCollectionFilteringProvider“ sind solche „AfterInvocationProvider“.

x| Bereits benutzt im Abschnitt „Konfiguration“ benutzt (Element `<intercept-url />`)

x| Umfangreicheres Beispiel

```
<intercept-url pattern="/app/**"  
              method="POST"  
              requires-channel="https"  
              access="ROLE_WRITER,MY_POLICY_DEPT_A" />
```

x| Attribut „access“ enthält eine Liste von „ConfigAttributes“ (siehe Abschnitt „Zentrale API Komponenten“), welche von den „Providern“ ausgewertet werden.

x| Auf Methoden-Ebene für bspw. Service-Layer

x| Wird aktiviert über das Element `<global-method-security>`

```
<global-method-security secured-annotations="enabled"  
                        jsr250-annotations="disabled" />
```

x| Spring baut um die von Spring verwalteten Beans einen Proxy herum, der einen „MethodSecurityInterceptor“ einfügt.

x| Schutz über Annotations auf Methoden oder Klassen/Interfaces

```
public interface BillingService {  
    @Secured( { "ROLE_USER" })  
    public void createBill();  
}
```

- x| Auf Methoden-Ebene für bspw. Service-Layer
- x| Wird aktiviert über das Element `<intercept-methods>` als Dekoration in einem `<bean>` Element

```
<beans:bean id="myService" class="de.excellent.MyServiceImpl">  
  <intercept-methods>  
    <protect method="set*" access="ROLE_ADMIN" />  
  </intercept-methods>  
</beans:bean>
```

- x| Vom Verhalten und Handling her wie Annotations

- x| Auf Methoden-Ebene für bspw. Service-Layer
- x| Wird wie die Annotations über `<global-method-security>` aktiviert.

```
<global-method-security>  
  <protect-pointcut  
    expression="execution(* de.excellent.*Service.*(..))"  
    access="ROLE_USER,MY_ACL_SYSTEM" />  
</global-method-security>
```

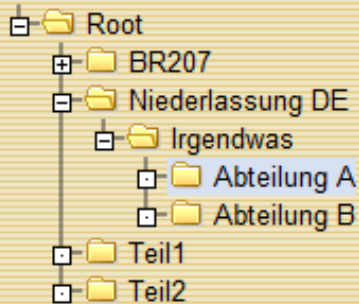
- x| Mächtige AspectJ Joinpoints auf Methoden-Ebene möglich.
- x| Bei manueller Konfiguration können auch echte .aj-Advices benutzt werden.

- x| Annotations können evtl. vergessen oder mit falschen Attributen bestückt werden, was für die Sicherheit fatal wäre.
Durch die Verteilung über den Code gibt es keinen Überblick.
- x| Gleiches gilt für die AOP Alliance Dekoration der Beans
- x| Mein Favorit: AspectJ
 - | Restriktive Standard-Attribute, welche mit Wildcards bspw. alle Service-Methoden abdecken.
 - | Explizite Lockerung bei einzelnen Methoden/Services.

- x| Koppelung von ACL an Objekt über ObjectIdentity
- x| ACL hat einen Parent -> Bäume können aufgebaut werden
- x| ACL hat einen Eigentümer
- x| ACL hält eine Liste von ACEs
- x| Koppelung von ACE an Benutzer/Rolle über SID
- x| ACE besteht aus SID, Permission und boolean granting
- x| AclService verwaltet ACLs und kann einen AclCache benutzen (EhCache)

Berechtigungen

Teile Benutzer und Gruppen



Berechtigungen für Teil Abteilung A:

Neu Löschen Speichern

Besitzer:

admin

	Benutzer oder Gruppe	Zugriff	Prüfplan			Daten		Admin
			Str	L	S	L	S	
<input type="radio"/>	Abt A	Zulassen	✓	✓	✓	✓	✓	
<input type="radio"/>	DE Chef	Verweigern		✗	✗			
<input type="radio"/>	Abt A	Zulassen	✓	✓				
<input type="radio"/>	Abt B	Zulassen	✓	✓				
<input type="radio"/>	DE Chef	Zulassen	✓	✓				
<input type="radio"/>	ROLE_ADMINISTRATOR	Zulassen	✓	✓	✓	✓	✓	✓

1 1 .. 6 von 6 8

- x| ACLs nur auf Objekte möglich, nicht auf Attribute oder Klassen.
- x| „ObjectIdentity“ benötigt Id (PK) und Typ (Class), um die Relation zum Domain Object herzustellen.
`delete(long contractId)` somit nicht direkt prüfbar.
- x| Aus dem selben Grund können zwei Domain Objekte nicht mit einer ACL abgedeckt werden.
- x| Collection-Filter ist ineffektiv (eine Abfrage beim AclService pro Collection-/Array-Element!)
- x| AclImpl benutzt zum Vergleich von „ACE.permission“ und „requiredPermission“ ein „==“ anstatt „&“.
- x| Issue Tracker: [SEC-479](#), [SEC-593](#), [SEC-642](#), [SEC-925](#)

Überblick

Ausflug zum Spring Framework

Zentrale API Komponenten

Authentifizierung

Konfiguration

Autorisierung

x| Fazit

- x|** „Keep it simple“? Definitiv ja!
Trotzdem steht man vor einem mächtigen und modularen Framework, das sich zu Beginn als komplex darstellen kann.
- x|** Spring Security ist nur ein Baustein unter vielen, um eine Applikation abzusichern. Dieser Baustein ist aber in sich schlüssig und bietet „out of the box“ Feeling.
- x|** Schwächen im ACL-Bereich, aber auf der Roadmap zu 2.5 stehen viele dieser Punkte.
- x|** Dokumentation enthält eine Lücke zwischen Namespace-Konfiguration und Bean-Konfiguration. Der Bogen hin zur neuen Namespace-Konfiguration ist noch nicht ganz gespannt.

x| Bei mir:

- | Hier und Jetzt
- | Im Ausstellungsbereich
- | Email: m.renner@excellent.de