ENTERPRISE DEVELOPMENT SERVICES DIRECT FROM THE CREATORS OF SPRING FRAMEWORK

INTERFACE21

Ease of Security

Java Forum Stuttgart 2007

Mike Wiesner, Interface21

Spring

INTERFACE21

Über mich

- Senior Consultant bei Interface21 Germany
- Acegi-/Spring-Consulting
- Trainings
- IT-Security Consulting
- mwiesner@interface21.com

Copyright 2004-2006, Interface21 GmbH. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

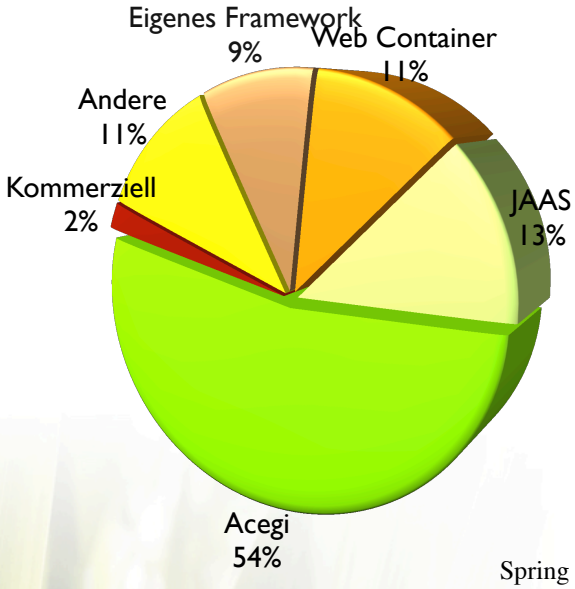
Acegi vs. Spring Security

- Ursprünglich:
  - Acegi Security System for Spring
  - Entwicklung getrennt von Spring durch Ben Alex
  - Acegi ist eine Firma von Ben Alex
- Ab Version 1.1:
  - Spring Security
  - Entwicklung zusammen mit Spring durch Interface21
  - Ben Alex arbeitet nun bei Interface21

Spring

INTERFACE21

Einsatz von Security-Frameworks



Spring User Group Umfrage

Spring

INTERFACE21

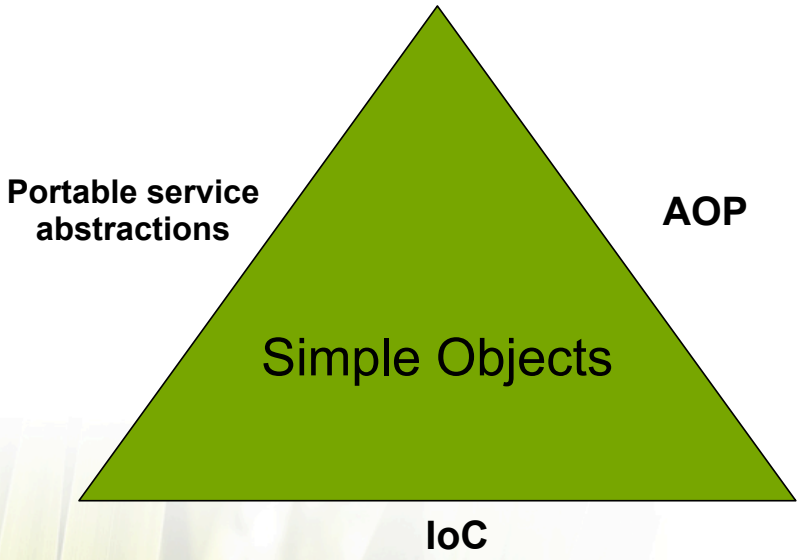
Was ist Spring Security?

- Ein Sicherheitsframework auf Basis von Spring
- Spring Security verwendet also Spring
- Muss die Anwendung selbst Spring verwenden?

Spring

INTERFACE21

Spring Anwendung?



Spring

INTERFACE21

Was davon ist nötig?

- Anwendung muss nichts von Spring verwenden
- Alle Web-Anwendung (JSF, Struts, JSP, Spring MVC)
- Nahezu alle Java-Anwendungen
- Integration und Abstraktion für JAAS möglich
- Nur Spring Security selbst benötigt Spring zur Konfiguration

Spring

INTERFACE21

Wieso brauchen wir das, es gibt doch JAAS?

- LoginModules sind Closed-Source
- Callback Handler stören mehr als sie nutzen
- Konfiguration immer für komplette JVM
- Konfiguration über Umgebungsvariablen
- Wenig Autorisierungsmöglichkeiten
- Abhängigkeiten zum Application Server
- Keine Trennung von Businesscode und Securitycode
- Schlechte Testbarkeit

Spring

INTERFACE21

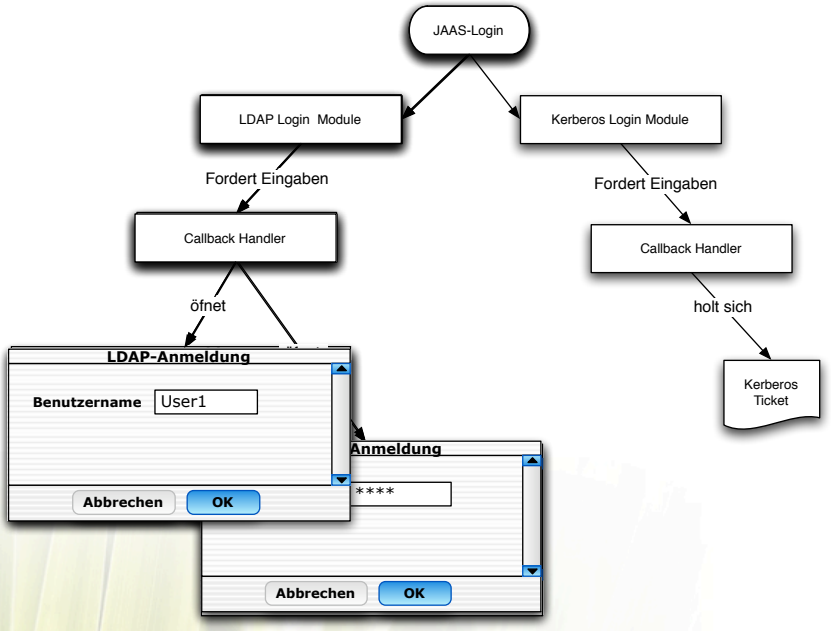
Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Copyright 2004-2006, Interface21 GmbH. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

JAAS



Authentifizierung mit JAAS

```
LoginContext lc = new LoginContext("loginConf", new
TextCallbackHandler());

lc.login();

String principal = ((Principal) lc.getSubject
().getPrincipals().iterator().next()).getName();
```

INTERFACE21

Authentifizierung mit JAAS

```
loginConf {
com.sun.security.auth.module.LdapLoginModule REQUIRED
userProvider="ldap:///cn=users,dc=example,dc=com"
authIdentity="{USERNAME}"
userFilter="(uid={USERNAME})"
     };


#> java ... -Djava.security.auth.login.config=conf/
simple.conf ...
```


Spring

INTERFACE21

JAAS Callback Handler

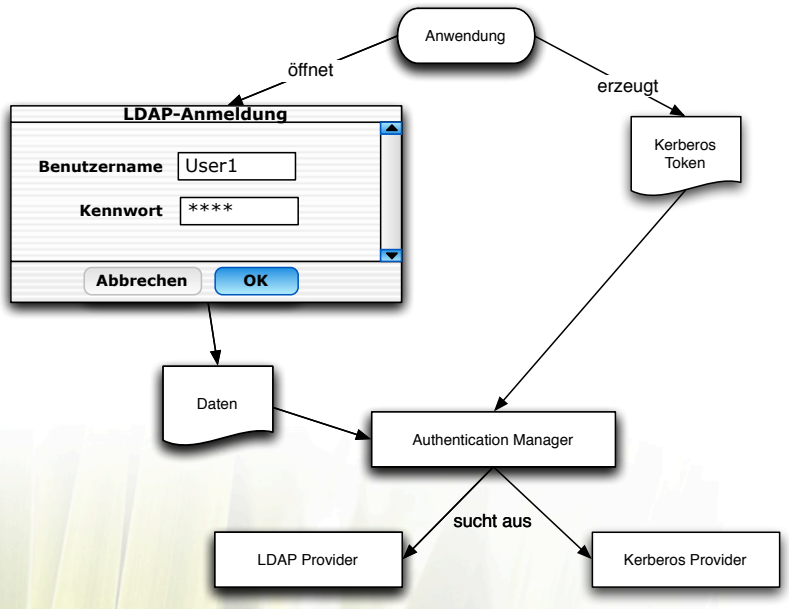
- LoginModule benötigt Eingaben und fragt beim Callback Handler nach
- Callback Handler fordert Eingaben vom Benutzer (Vorbild war hier PAM)
- Probleme bei Webanwendungen und Client-/ Serveranwendungen!
- Lösung: Callback Handler bekommt Datenobjekt

Spring

Authentifizierung mit JAAS

```
public class MyCallbackHandler implements CallbackHandler {
    private String password;
    private String username;

    public MyCallbackHandler(String password, String username) {
        this.password = password;
        this.username = username;
    }

    public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
        Callback cb;
        for (int i = 0; i < callbacks.length; i++) {
            cb=callbacks[i];
            if (cb instanceof NameCallback) {
                NameCallback nameCallback = (NameCallback) cb;
                nameCallback.setName(username);
            }
            else if (cb instanceof PasswordCallback) {
                PasswordCallback passwordCallback = (PasswordCallback) cb;
                passwordCallback.setPassword(password.toCharArray());
            }
            else
            {
                throw new UnsupportedCallbackException(cb, "Only NameCallback and
PasswordCallback supported");
            }
        }
    }
}
```


Spring Security



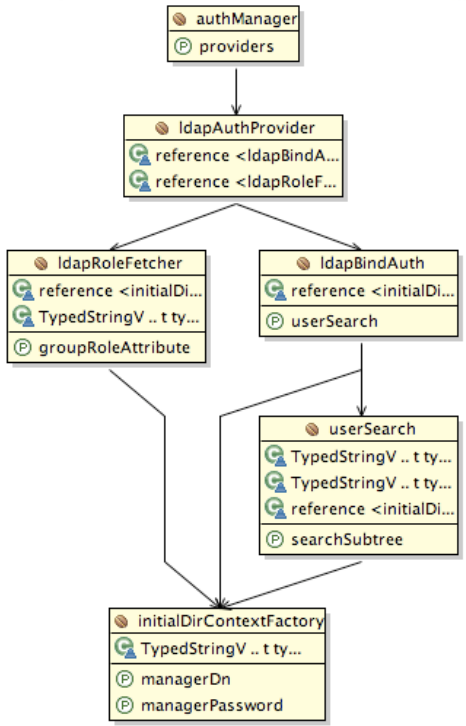
INTERFACE21

Authentifizierung mit Spring Security

```
ApplicationContext ctx = new
ClassPathXmlApplicationContext("springsecurity.xml");

AuthenticationManager manager =
(AuthenticationManager) ctx.getBean("authManager");

Authentication auth = manager.authenticate(new
UsernamePasswordAuthenticationToken(benutzername,
passwort));

User principal = (User) auth.getPrincipal();
```


Spring

INTERFACE21



Copyright 2004-2006, Interface21 GmbH. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

Authentifizierung mit Spring Security

```
<bean id="userSearch"
class="...FilterBasedLdapUserSearch">
    <constructor-arg value=""/>
    <constructor-arg value="(uid={0})"/>
    <constructor-arg ref="initialDirContextFactory" />
    <property name="searchSubtree" value="true" />
</bean>

<bean id="ldapRoleFetcher"
class="...DefaultLdapAuthoritiesPopulator">
    <constructor-arg ref="initialDirContextFactory" />
    <constructor-arg value="ou=groups" />
    <property name="groupRoleAttribute" value="cn" />
</bean>
```


Spring

INTERFACE21

Authentifizierung mit Spring Security

- Auswahl des passenden Providers anhand von Eingaben und nicht umgekehrt
- Unabhängig von der Technologie/Umgebung
- Mehrere Konfigurationen in einer JVM möglich
- Keine Umgebungsvariablen nötig
- Feingranulare Authentication Provider für DAO (InMemory, JDBC), Siteminder, CAS, X509, LDAP, Testing, JAAS

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

Web-Sicherheit mit JAAS/Tomcat

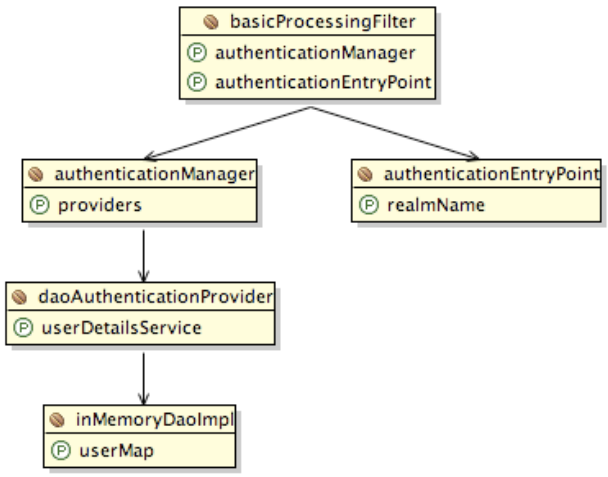
```
<tomcat-users>
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <role rolename="manager"/>
  <user username="tomcat" password="tomcat"
roles="tomcat,manager"/>
  <user username="both" password="tomcat"
roles="tomcat,role1"/>
  <user username="role1" password="tomcat"
roles="role1"/>
</tomcat-users>
```


Spring

Web-Sicherheit mit JAAS/Tomcat

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Test</web-resource-name>
    <url-pattern>/html/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
     <role-name>manager</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>manager</role-name>
</security-role>
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Test</realm-name>
</login-config>
```


Web-Sicherheit mit Spring Security



INTERFACE21

Web-Sicherheit mit Spring Security

```
<bean id="filterSecurityInterceptor"
class="...FilterSecurityInterceptor">
    <property name="authenticationManager"
ref="authenticationManager" />
    <property name="accessDecisionManager"
ref="accessDecisionManager" />
    <property name="objectDefinitionSource"><value>
          CONVERT_URL_TO_LOWERCASE_BEFORE_COMPARISON
          PATTERN_TYPE_APACHE_ANT
          /admin/**=ROLE_ADMIN
    </value></property>
</bean>
```


Spring

INTERFACE21

Vorteile Spring Security

- Webanwendung ist unabhängig von Spring
- Umfangreichere URL-Patterns (ANT, RegExp)
- Unabhängig vom Container
- Konfiguration an einer Stelle
  - Kann auch aus Datenbank kommen
- Unterstützung von BASIC, Form, Digest, X509, Remember-Me, Captchas, CAS, (SPNEGO)

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

Methoden-Sicherheit mit JAAS

```
SecurityManager sm = System.getSecurityManager();
if (sm != null) {
   try {
      sm.checkPermission(new PropertyPermission
("my.perm", "read"));
   } catch (SecurityException e) {
      // No permission
   }
}
```

```
if (request.isUserInRole("ROLE_USER")) {
    // do something
}
```


Spring

INTERFACE21

Methoden-Sicherheit mit JAAS

- Nur programmatisch
- Vermischung des Businesscodes vom Securitycode
- Keine zentrale Verwaltung/Übersicht
- Kann leicht vergessen werden
- Ausname: EJB
  - Security Constraints
  - Annotations
  - Aber dann Abhängig von EJB!

Spring

INTERFACE21

Methoden-Sicherheit mit Spring Security

```
<bean id="securityInterceptor"
    class="...AspectJSecurityInterceptor">
    <property name="authenticationManager" .../>
    <property name="accessDecisionManager" ... />
    <property name="objectDefinitionSource"><value>
          BeanToSecure.adminOp=ROLE_ADMIN
          BeanToSecure.userOp=ROLE_USER,ROLE_ADMIN
    </value></property>
</bean>
```


Spring

INTERFACE21

Methoden-Sicherheit mit Spring Security

```
<bean id="mySecurityAspect" class="...MySecurityAspect"
    factory-method="aspectOf">
    <property name="securityInterceptor"
       ref="securityInterceptor"/>
</bean>
```


Spring

INTERFACE21

Methoden-Sicherheit mit Spring Security

```
public aspect MySecurityAspect
{
    private AspectJSecurityInterceptor
       securityInterceptor;

    pointcut domainObjectInstanceExecution():
       execution(public * BeanToSecure.*(..))

    ...

}
```

Copyright 2004-2006, Interface21 GmbH. Copying, publishing, or distributing without expressed written permission is prohibited.

Spring

INTERFACE21

Methoden-Sicherheit mit Spring Security

- deklarativ
- zentrale Verwaltung/Übersicht
- getrennt vom Businesscode
- Businesscode ist nicht Abhängig von Spring
- Kann leicht angepasst werden
- Funktioniert auch in der SE und ohne SecurityManager
- Aspekte werden einkompiliert
- Auch mit Annotations möglich!

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

Instanz-Sicherheit

- Zugriffsregeln hängen oft von Instanzen ab, z.B.
  - Benutzer dürfen eigene Objekte lesen und schreiben
  - Benutzer dürfen fremde Objekte nur lesen
  - Gruppenleiter dürfen Objekte von Gruppenmitglieder lesen und schreiben

Spring

INTERFACE21

Instanz-Sicherheit mit JAAS

- Nur programmatisch im Businesscode
- Abhängigkeit des Businesscodes vom Securitycode
- Keine zentrale Verwaltung/Übersicht
- Kann leicht vergessen werden
- EJBs bieten auch nichts an

Spring

INTERFACE21

Instanz-Sicherheit mit Spring Security

- ObjectDefinitionSource erweitern um z.B.
  - BeanToSecure.save=ROLE_USER,ACL_BTS_SAVE
- Zusätzlicher ACL-Voter zum Role-Voter
- Methoden-Eingaben/-Ausgaben werden vom ACL-Provider geprüft
- Zugriff wird gewährt bzw. Exception geworfen
- Rückgabewerte werden transparent gefiltert
- ACL sind sozusagen dynamische Rollen

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

Single Sign On mit JAAS

- Keine Out-Of-The-Box Lösung
- SSO mit Kerberos nur in Verbindung mit GSS-API
  - Betrifft z.B. Microsoft ADS
- Keine Web-SSO Unterstützung

Spring

INTERFACE21

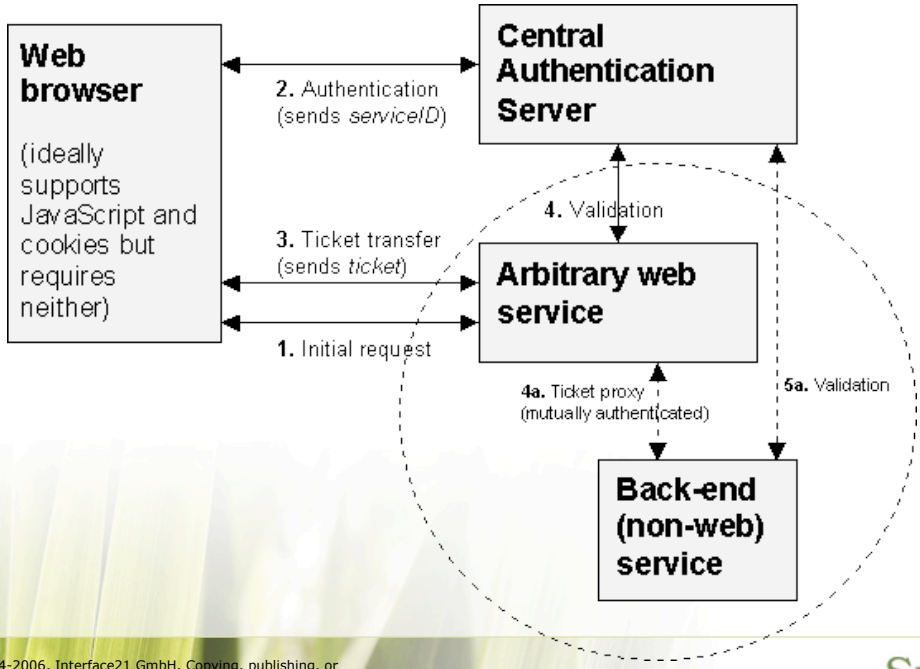
SSO mit Spring Security

- Out-Of-The-Box Unterstützung für Web-SSO mit JA-SIG CAS und Siteminder
- Integrationen möglich für:
  - Web-SSO mit Kerberos/SPNEGO
  - SSO mit Kerberos und automatischer Principal-Propagierung über RMI und HTTP
  - OpenID (Sandboxcode)

Spring

INTERFACE21

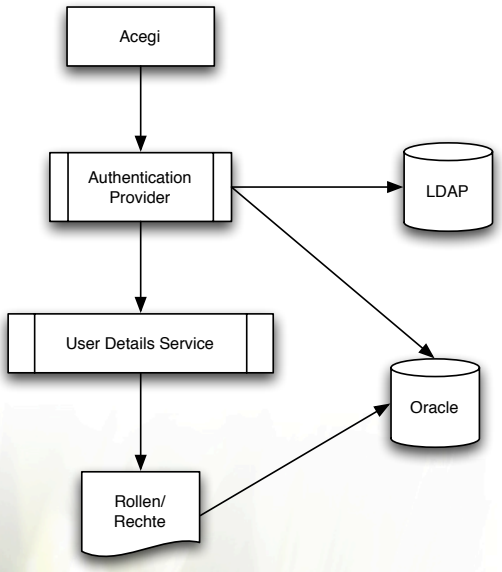
CAS mit Spring Security



Spring

INTERFACE21

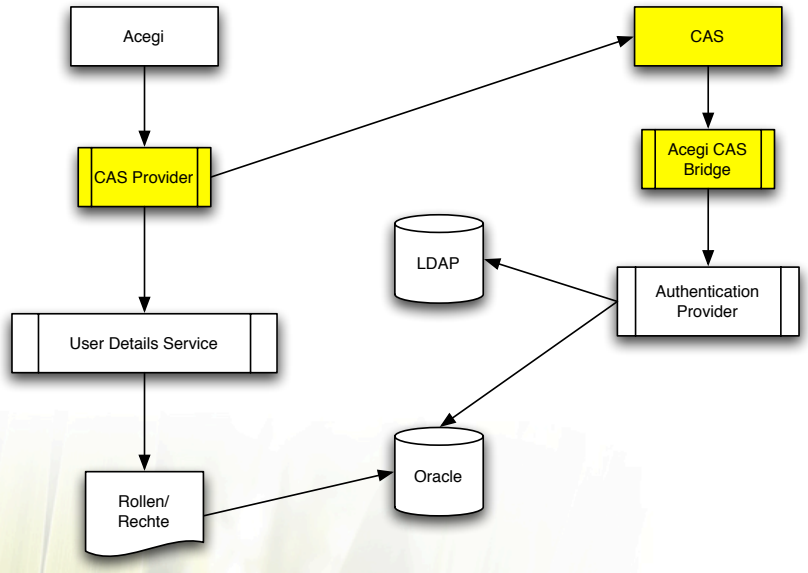
Spring Security ohne CAS



Spring

INTERFACE21

Spring Security mit CAS



Spring

INTERFACE21

CAS mit Spring Security

- CAS-Client integriert in Spring Security
- CAS Authentication Provider Bridge
  - Spring Security Provider in CAS verwendbar
- CAS führt nur Authentifizierung durch
- Autorisierung über UserDetailsService

Spring

INTERFACE21

Kerberos/SPNEGO mit Spring Security

- SPNEGO = Single Sign On über den Browser
  - z.B. in Windows Netzwerken
- SPNEGO Authentication Provider
- SPNEGO Entry Point
- SPNEGO ServletFilter
- Autorisierung über UserDetailsService

Spring

INTERFACE21

Agenda

- Acegi vs. Spring Security
- Authentifizierung
- Web-Sicherheit
- Methoden-Sicherheit
- Instanz-Sicherheit
- Single-Sign-On
- Testing
- Fazit

Spring

INTERFACE21

Testing mit JAAS

- Businesscode-Tests:
  - Security-Code wird immer mit getestet
  - Login erforderlich
- Security-Tests:
  - Kein getrenntes Testen von Autorisierung und Authentifizierung möglich
  - Keine fertigen Test Login Modules
  - Businesscode wird immer mit getestet

➡ Kein separates Testen möglich da Code vermischt ist.

Spring

INTERFACE21

Testing mit Acegi

- Businesscode-Tests:
  - Security kann vollständig abgeschaltet werden (Aspekte entfernen)
  - Kein Login Erforderlich
- Securitycode-Tests:
  - Getrenntes Testen von Authentifizierung und Autorisierung (TestingAuthenticationProvider)
  - Businesscode muss nicht ausgeführt werden

➡ Getrenntes testen durch Code-Trennung möglich

Spring

INTERFACE21

Fazit

- Spring Security...
  - setzt kein Spring im Anwendungscode voraus
  - ist abhängig vom Businesscode und nicht andersrum
  - ist durch die feingranulare Struktur leicht anpassbar und erweiterbar
  - bietet bereits sehr viel Out-Of-The-Box
  - vereinfacht das Testen
  - kann gemeinsam mit JAAS eingesetzt werden
  - funktioniert in jeder Umgebung
  - ist komplett Open Source (Apache Lizenz)
  - professioneller Support vorhanden

Spring

INTERFACE21

Wie geht‘s weiter?

- Spring Security 2.0 (M1 im Juli 2007)
  - Spring Security Namespace
  - Hierarchische Rollen
  - Single Sign On Verbesserungen
  - Spring LDAP Integration

Spring

INTERFACE21

Security Namespace

```
<bean id="authenticationProcessingFilter"
    class="org.acegisecurity.ui.webapp.AuthenticationProcessingFilter">
    <property name="authenticationManager"
    ref="authenticationManager"/>
    <property name="authenticationFailureUrl" value="/login.jsp"/>
    <property name="defaultTargetUrl" value="/"/>
</bean>

<security:authentication-form id="authenticationProcessingFilter"
      authenticationManager="authenticationManager"
      errorFormUrl="/login.jsp"
      defaultTargetUrl="/"/>
```


Spring

INTERFACE21

Wo geht‘s weiter?

- Homegage:
  http://www.springsecurity.org
- Spring User Group Germany
  http://groups.google.com/group/sugg
- Forum
  http://forum.springframework.org

Spring

INTERFACE21

Buch zum Film

Spring Security

Das Acegi Security System professionell einsetzen

ISBN: 3-939084-26-3

Mike Wiesner

Herbst 2007

Spring