pro-netics group

sourcesense
making sense of open source

The hidden gems of Spring Security

Dessì Massimiliano

Spring Meeting 27 giugno 2009 Cagliari

Author

Software Architect and Engineer
ProNetics / Sourcesense

Presidente
JugSardegna Onlus

Fondatore e coordinatore
SpringFramework Italian User Group

Autore
Spring 2.5 Aspect Oriented Programming

pro-netics group

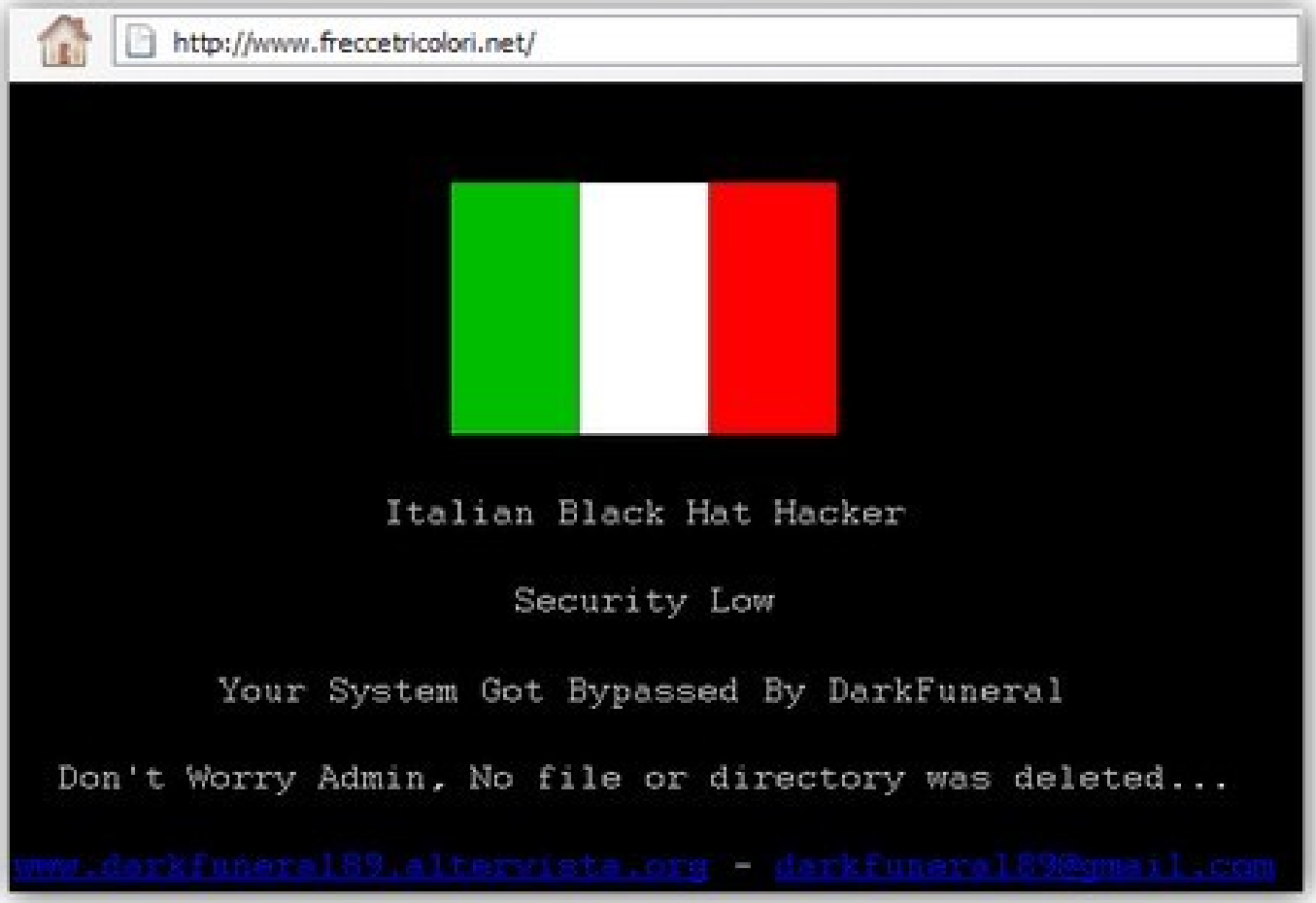
sourcesense
making sense of open source

La sicurezza non viene vista dagli utenti

pro-netics group

sourcesense
making sense of open source

La sicurezza viene percepita quando non c'è oppure è ad un livello insufficiente



pro-netics group

sourcesense
making sense of open source

I dati sono il valore delle applicazioni

pro-netics group

sourcesense
making sense of open source

E' necessario proteggere i dati

Non farci le scommesse

pro-netics group

sourcesense
making sense of open source

In questa presentazione vedremo varie facce della sicurezza con SpringSecurity

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

pro-netics group

REST

sourcesense
making sense of open source

- Tutto è una risorsa
- Ogni risorsa ha un identificativo
- Representational State Transfer
- Tutte le risorse hanno una interfaccia uniforme

  GET, POST, PUT, DELETE
- Hypermedia as the engine of application state
- Client-server, Stateless, Cacheable, Layered

http://www.magic-box.org/services/social/center/13

JAX-RS RESTeasy @

```
@Path("services/social")
@Component("socialController")
public class SocialController {

    @GET
    @Path("/{domain}/{id}")
    @Produces("application/xml")
    public Response getEntity(@PathParam("domain") String domain, @PathParam("id") String id) {
        Entity entity = null;
        try {
            entity = getEntityManager().getEntity(domain, id);
        } catch (InvalidEntityException e) {
            String[] args = { id, domain };
            LOG.info("Request an invalid entity with id {} and domain {}", args);
        }
        ResponseBuilder builder = (entity == null) ? Response.noContent() :
                                                     Response.ok(getEntityConverter().convert(entity));
        return builder.build();
    }

    @DELETE
    @Path("/{domain}/{id}")
    public String deleteEntity(@PathParam("domain") String domain, @PathParam("id") String id) {
        getEntityManager().deleteEntity(domain, id);
        return OK;
    }
```

REST


```
@POST
@Path("/{domain}/{id}")
@Consumes("application/xml")
public String saveEntity(InputStream is, @PathParam("domain") String domain,
     @PathParam("id") String id){

     String xml = inputStreamToString(is);
    Entity entity = getEntityConverter().unconvert(xml, id, domain);
    getEntityManager().deleteEntity(domain, id);
    getEntityManager().saveEntity(entity);
    return OK;
}

@PUT
@Path("/{domain}/{id}")
@Consumes("application/xml")
public String updateEntity(InputStream is, @PathParam("domain") String domain,
                           @PathParam("id") String id){

     String xml = inputStreamToString(is);
    Entity entity = getEntityConverter().unconvert(xml, id, domain);
    getEntityManager().saveEntity(entity);
    return OK;

..
}
```

In Ingresso e in uscita è permesso qualunque tipo , XML, JSON, text/plain, files binari, ModelAndView, String...

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

SpringSecurity

SpringSecurity è il framework per la sicurezza maggiormente scaricato su sourceforge (250k).

Permette di gestire qualsiasi aspetto della sicurezza di una webapplication in maniera molto accurata.

Sopratutto agisce in maniera trasparente senza intervenire nel funzionamento della webapplication.

Le chiamate passano attraverso un filtro servlet

```
<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>
```

che propaga le informazioni di autenticazione al thread di esecuzione

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

Autenticazione

Processo con cui attraverso le credenziali viene verificato se l'utente è colui che dichiara di essere.

```
<sec:authentication-provider user-service-ref="sffsUserDetailservice">
     <sec:password-encoder hash="sha" />
</sec:authentication-provider>

<bean id="sffsUserDetailservice" class="it.freshfruits.security.AuthenticationJdbcDaoImpl">
     <property name="rolePrefix" value="ROLE_"/>
     <property name="dataSource" ref="dataSource"/>
     <property name="usersByUsernameQuery"
          value="SELECT username, password, enabled FROM authentication WHERE username = ?" />
     <property name="authoritiesByUsernameQuery"
          value="SELECT username, authority FROM roles WHERE username = ?" />
</bean>
```

Autorizzazione

Quali authority possono accedere e a cosa

Autorizzazione

```
<sec:http>
     <sec:intercept-url pattern="/log*.jsp" filters="none" />
      <sec:intercept-url pattern="*.htm" access="ROLE_USER,ROLE_ANONYMOUS" />
     <sec:intercept-url pattern="*.page" access="ROLE_USER,ROLE_ADMIN" />
     <sec:intercept-url pattern="*.edit" access="ROLE_USER,ROLE_ADMIN" />
     <sec:intercept-url pattern="*.admin" access="ROLE_ADMIN" />
     <sec:form-login login-page="/login.jsp" default-target-url="/"
          login-processing-url="/j_security_check"  authentication-failure-url="/loginError.jsp" />
     <sec:logout logout-url="/logout.jsp" logout-success url="/login.jsp" />
     <sec:remember-me />
</sec:http>

<bean id="accessDecisionManager" class="org.springframework.security.vote.AffirmativeBased">
     <property name="decisionVoters">
          <list>
               <bean class="org.springframework.security.vote.RoleVoter" />
               <bean class="org.springframework.security.vote.AuthenticatedVoter" />
          </list>
     </property>
</bean>

<sec:global-method-security access-decision-manager-ref="accessDecisionManager">
     <sec:protect-pointcut expression="execution(* it.freshfruits.domain.entity.*.*(..))"
          access="ROLE_USER,ROLE_ADMIN" />
</sec:global-method-security>
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

Groups

Possiamo utilizzare i gruppi, assegniamo i ruoli ai gruppi e assegniamo gli utenti a dei gruppi.

In questo modo l'utente al login si ritroverà automaticamente i ruoli dei gruppi al quale appartiene.

Modifichiamo il detailService da così:

```
<bean id="sffsUserDetailservice" class="it.freshfruits.security.AuthenticationJdbcDaoImpl">
     <property name="rolePrefix" value="ROLE_"/>
     <property name="dataSource" ref="dataSource"/>
     <property name="usersByUsernameQuery"
          value="SELECT username, password, enabled FROM authentication WHERE username = ?" />
     <property name="authoritiesByUsernameQuery"
          value="SELECT username, authority FROM roles WHERE username = ?" />
</bean>
```

Groups

A così:

```
<bean id="sffsUserDetailservice" class="it.freshfruits.security.AuthenticationJdbcDaoImpl">
        <property name="dataSource" ref="dataSource" />
        <property name="usersByUsernameQuery"
            value="SELECT username, password, enabled
                FROM authentication WHERE username = ?" />
        <property name="groupAuthoritiesByUsernameQuery"
            value="SELECT g.id, g.group_name, ga.name
                FROM groups g, group_members gm, authorities ga
                WHERE gm.username = ? AND g.id = ga.group_id AND g.id = gm.group_id"/>
        <property name="enableGroups" value="true"/>
        <property name="enableAuthorities" value="false"/>
</bean>
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

OpenID

“Le password sono come le mutande...

Non devi farle vedere a nessuno

Devi cambiarle con regolarità

Non devi prestarle a nessuno”

OpenID è un meccanismo decentralizzato per il single sign on.
La nostra applicazione non contiene più la password, di cui si occupa un provider OpenID.
Basta fornire l' username del tipo:

http://max.myopenid.com

OpenID

L'autenticazione avviene esternamente sul provider openid dove verrà richiesta la password

```
    <sec:http>
        <sec:concurrent-session-control max-sessions="1" expired-url="/openidlogin.jsp*"/>
        <sec:intercept-url pattern="/openidlogin.jsp*" filters="none"/>
        <sec:intercept-url pattern="/" access="ROLE_USER" />
        <sec:intercept-url pattern="/**" access="ROLE_USER" />
        <sec:openid-login login-page="/openidlogin.jsp" default-target-url="/" />
    </sec:http>
```

```
...
<h3>Please Enter Your OpenID Identity</h3>
<form name="f" action="<c:url value='j_spring_openid_security_check'/>" method="POST">
  <table>
    <tr><td>OpenID Identity:</td><td><input type='text' name='j_username'value=''/></td></tr>
    <tr><td colspan='2'><input name="submit" type="submit"></td></tr>
    <tr><td colspan='2'><input name="reset" type="reset"></td></tr>
  </table>
</form>
</body>
</html>
```

La nostra applicazione a questo punto manterrà solo gli username e gli specifici ruoli applicativi, possiamo implementarci così il nostro UserDetailService

```
public class FreshFruitDetailsService implements UserDetailsService {

    private String defaultRole;
    private DataSource dataSource;

    public UserDetails loadUserByUsername(String username) throws
             UsernameNotFoundException, DataAccessException {

      UsersByUsernameMapping mapping = new UsersByUsernameMapping(dataSource);
      List users = mapping.execute(username);
      return (UserDetails) users.get(0);
    }


...
```

OpenID


```
  private class UsersByUsernameMapping extends MappingSqlQuery {

   protected UsersByUsernameMapping(DataSource ds) {
    super(ds,"SELECT username, authority, enabled FROM users WHERE username = ? ;");
    declareParameter(new SqlParameter(Types.VARCHAR));
    compile();
   }

   protected Object mapRow(ResultSet rs, int rownum) throws
                                        SQLException {
       String username = rs.getString(1);
       String password = rs.getString(2);
       boolean enabled = rs.getBoolean(3);
       UserDetails user = new User(username, password, enabled, true, true, true,
            New GrantedAuthority[] { new GrantedAuthorityImpl("HOLDER") });
       return user;
   }
  }


  public void setDatasource(DataSource dataSource) {
     this.dataSource = dataSource;
  }

}
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

pro-netics group

OAuth

sourcesense
making sense of open source

"An open protocol to allow secure API authorization in a simple and standard method from desktop and web applications."

Una applicazione ne autorizza un altra ad utilizzare i dati di un suo utente autenticato.

OAuth


```
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-2.0.4.xsd
http://spring-security-oauth.codehaus.org/2.0
http://spring-security-oauth.codehaus.org/schema/spring-security-oauth-2.0.xsd">

    <oauth:provider consumer-details-service-ref="consumerDetails"
                    token-services-ref="tokenServices"
                    request-token-url="/oauth/request_token"
                    authenticate-token-url="/oauth/authorize"
                    authentication-failed-url="/oauth/confirm_access"
                    access-granted-url="/request_token_authorized.jsp"
                    access-token-url="/oauth/access_token"/>

  <oauth:consumer-details-service id="consumerDetails">
    <oauth:consumer name="Tonr.com" key="tonr-consumer-key"
          secret="SHHHHH!!!!!!!!!!" resourceName="Your Photos"
          resourceDescription="Your photos that you have uploaded to sparklr.com."/>
  </oauth:consumer-details-service>

  <oauth:token-services id="tokenServices"/>
```

pro-netics group

OAuth

sourcesense
making sense of open source

"An open protocol to allow secure API authorization in a simple and standard method from desktop and web applications."

Una applicazione ne autorizza un altra ad utilizzare i dati di un suo utente autenticato.

OAuth


```
http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security-2.0.4.xsd
http://spring-security-oauth.codehaus.org/2.0
http://spring-security-oauth.codehaus.org/schema/spring-security-oauth-2.0.xsd">

    <oauth:provider consumer-details-service-ref="consumerDetails"
                    token-services-ref="tokenServices"
                    request-token-url="/oauth/request_token"
                    authenticate-token-url="/oauth/authorize"
                    authentication-failed-url="/oauth/confirm_access"
                    access-granted-url="/request_token_authorized.jsp"
                    access-token-url="/oauth/access_token"/>

  <oauth:consumer-details-service id="consumerDetails">
    <oauth:consumer name="Tonr.com" key="tonr-consumer-key"
          secret="SHHHHH!!!!!!!!!!" resourceName="Your Photos"
          resourceDescription="Your photos that you have uploaded to sparklr.com."/>
  </oauth:consumer-details-service>

  <oauth:token-services id="tokenServices"/>
```

OAuth

Oauth for Spring security http://spring-security-oauth.codehaus.org non è parte della distribuzione di Spring,

ma funziona senza problemi :)

Se usato con OpenID bisogna tenere conto che OpenID come `Auhtentication` usa un `OpenIDAuthenticationToken` perciò il servizio autorizzante deve tenerne conto

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

ACL

Una Access Control list permette di gestire i permessi sui singoli oggetti di dominio, definendo chi può fare cosa su quell' oggetto.

```
ObjectIdentity oid = new ObjectIdentityImpl(MyClass.class, 13);
MutableAcl acl = mutableAclService.createAcl(oid);
acl.insertAce(0, BasePermission.READ, new GrantedAuthoritySid(role.getName()), true);
acl.insertAce(1, BasePermission.WRITE, new GrantedAuthoritySid(“ROLE_WRITE”), true);
acl.insertAce(2, BasePermission.CREATE, new GrantedAuthoritySid(“ROLE_CREATE”), true);
acl.insertAce(3, BasePermission.DELETE, new GrantedAuthoritySid(“ROLE_DELETE”), true);
acl.insertAce(4, BasePermission.ADMINISTRATION, new PrincipalSid(“ROLE_ADMIN”), true);
mutableAclService.updateAcl(acl);
```

```
<bean id="aclService" class="it.pronetics.acl.repository.jdbc.PostgreSqlMutableAclService">
    <constructor-arg ref="dataSource" />
    <constructor-arg ref="lookupStrategy" />
    <constructor-arg ref="jdbcAclCache" />
    <property name="sidIdentityQuery" value="select currval('acl_sid_id_seq')" />
    <property name="classIdentityQuery" value="select currval('acl_class_id_seq')" />
</bean>
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

```
    <!-- Securing Methods with Security Interceptor -->

<bean class="org.springaop.chapter.five.security.FooServiceImpl">
    <security:intercept-methods access-decision-manager-ref="accessDecisionManager">
        <security:protect method="org.springaop.chapter.five.security.FooService.getBalance"
                  access="ROLE_USER" />
        <security:protect
                  method="org.springaop.chapter.five.security.FooService.setBalanceAccount"
                  access="ROLE_ACCOUNTING,ROLE_ADMIN" />
        <security:protect method="org.springaop.chapter.five.security.FooService.suspendAccount"
                  access="ROLE_ADMIN" />
    </security:intercept-methods>
</bean>

    <!-- Securing Methods with Pointcut-->

<global-method-security access-decision-manager-ref="accessDecisionManager">
   <protect-pointcut
        expression="execution(* org.springaop.chapter.five.security.FooService.getBalance(..))"
        access="ROLE_USER" />
   <protect-pointcut
        expression="execution(* org.springaop.chapter.five.security.FooService.set*(..))"
        access="ROLE_ACCOUNTING,ROLE_ADMIN" />
   <protect-pointcut
        expression="execution(*org.springaop.chapter.five.security.FooService.suspendAccount(..))"
        access="ROLE_ADMIN" />
</global-method-security>
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

Annotations


```
import org.springframework.security.annotation.Secured;

public class FooServiceImplWithAnnotations implements FooService {

    @Secured("ROLE_USER")
    public Integer getBalance(Integer idAccount) {
        Integer result = 0;
        ...
        return result;
    }

    @Secured( { "ROLE_ACCOUNTING", "ROLE_ADMIN" })
    public void setBalanceAccount(Integer id, Integer balance) {
        ...
    }

    @Secured("ROLE_ADMIN")
    public boolean suspendAccount(Integer id) {
        boolean result = false;
        ...
        return result;
    }
}
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

Spring 3

Oltre alle modifiche REST alla parte MVC, Spring 3 introduce una importante novita:

Spring Expression Language (SpEL)

Si tratta di un linguaggio molto espressivo che supporta l'interrogazione e la modifica di un grafo di oggetti a runtime, utilizzabile nella configurazione xml, nelle annotazioni nelle firme dei metodi

Nei bean sull' xml

```
<bean id="numberGuess" class="org.spring.samples.NumberGuess">
    <property name="randomNumber" value="#{ T(java.lang.Math).random() * 100.0 }"/>

</bean>


<bean id="shapeGuess" class="org.spring.samples.ShapeGuess">
    <property name="initialShapeSeed" value="#{ numberGuess.randomNumber }"/>

</bean>
```

e posso anche riferirli tra loro

Nel codice

```
public class SimpleMovieLister {

    private MovieFinder movieFinder;
    private String defaultLocale;

    @Value("#{ systemProperties['user.region'] }")
    public void setDefaultLocale(String defaultLocale){
        this.defaultLocale = defaultLocale;
    }

    @Autowired
    public void configure(MovieFinder movieFinder,
                          @Value("#{ systemProperties['user.region'] } String defaultLocale){
        this.movieFinder = movieFinder;
        this.defaultLocale = defaultLocale;
    }
...
}
```

pro-netics group

Agenda

sourcesense
making sense of open source

- REST
- SpringSecurity
- Authentication/Authorization
- Groups
- OpenID
- OAuth
- ACL
- AOP
- @ Annotations
- Spring 3
- SpringSecurity 3

Posso quindi usare SpEL negli url da intercettare

```
<http use-expressions="true">
    <intercept-url pattern="/secure/**" access="hasRole('ROLE_ADMIN') and
        hasIpAddress('192.168.1.0/24')" />
...
</http>
```

Posso usare delle @Pre e @Post annotazioni che lavorano insieme alla ACL e usare anche SpEL

```
<global-method-security pre-post-annotations="enabled"/>

@PreAuthorize("hasRole('ROLE_ADMIN')")
public void create(Contact contact);

@PreAuthorize("hasPermission(#contact, 'admin')")
public void deletePermission(Contact contact, Sid recipient, Permission permission);

@PreAuthorize("#contact.name == principal.name)")
public void doSomething(Contact contact);
```

Oppure posso usare SpEL per

filtrare le collezioni in base ai ruoli

```
@PreAuthorize("hasRole('ROLE_USER')")
@PostFilter("hasPermission(filterObject,'read')or
             hasPermission(filterObject,'admin')")
Public List getAll();
```

pro-netics group

sourcesense
making sense of open source

Domande ?

pro-netics group

sourcesense
making sense of open source

Grazie per l'attenzione !

Massimiliano Dessì
desmax74 at yahoo.it
massimiliano.dessi at pronetics.it

http://jroller.com/desmax
http://jroller.com/desmax
http://www.linkedin.com/in/desmax74
http://wiki.java.net/bin/view/People/MassimilianoDessi
http://www.jugsardegna.org/vqwiki/jsp/Wiki?MassimilianoDessi

Spring Framework Italian User Group
http://it.groups.yahoo.com/group/SpringFramework-it