Java Faces

03/01/2011

Spring + Security + Hibernate

Filed under: JSF — guilhermefinotti @ 21:09

Esse post demonstra a integração do Spring 3, Spring Security 3 e Hibernate 3 numa aplicação web.
O exemplo utiliza JSF 2.0 e Primefaces.

A estrutura de pacotes é a seguinte:
* **config** -> a classe de conexão (datasource)
* **model** -> as entidades Usuario e Perfil
* **services** -> a classe service customizada para autenticar o usuário usando o hibernate
* **web** -> o managed bean acessado pela tela de login (LoginMB), um PhaseListener utilizado para capturar exceções de autenticação e uma classe utilitária para JSF.

web.xml
Aqui configuramos o filtro "org.springframework.web.filter.DelegatingFilterProxy" e definimos os arquivos lidos na inicialização ("spring-config.xml" e "spring-security.xml").
As demais configurações são relativas ao JSF e Primefaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">
    <display-name>spring + security + hibernate</display-name>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                    /WEB-INF/spring-config.xml
                    /WEB-INF/spring-security.xml
        </param-value>
    </context-param>
    <context-param>
        <param-name>primefaces.SKIN</param-name>
        <param-value>none</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <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>
        <dispatcher>FORWARD</dispatcher>
        <dispatcher>REQUEST</dispatcher>
    </filter-mapping>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>Resource Servlet</servlet-name>
        <servlet-class>org.primefaces.resource.ResourceServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Resource Servlet</servlet-name>
        <url-pattern>/primefaces_resource/*</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

faces-config.xml
Aqui configuramos o "EL Resolver" e registramos um PhaseListener responsável pelos erros de autenticação do usuário.

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config version="2.0"
              xmlns="http://java.sun.com/xml/ns/javaee"
              xmlns:xi="http://www.w3.org/2001/XInclude"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd">
    <application>
        <el-resolver>org.springframework.web.jsf.el.SpringBeanFacesELResolver</el-resolver>
    </application>
    <lifecycle>
        <phase-listener>br.com.exemploseguranca.web.seguranca.LoginErrorPhaseListener</phase-listener>
    </lifecycle>
</faces-config>
```

spring-config.xml
Aqui configuramos o suporte à anotações e as propriedades da sessionFactory do Hibernate.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop" xmlns:sec="http://www.springframework.org/schema/security"
```

```
    xmlns:tx="http://www.springframework.org/schema/tx"
    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 http://www.springframework.org/schema/context http://www.springframework.org
/schema/context/spring-context-3.0.xsd">

    <!-- habilita a configuração por annotations -->
    <context:annotation-config />

    <!-- define os pacotes/subpacotes onde serão procurados beans do spring -->
    <context:component-scan base-package="br.com.exemploseguranca" />

    <!-- Propriedades do hibernate -->
    <bean id="sessionFactory"
        class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="annotatedClasses">
            <list>
                <value>br.com.exemploseguranca.model.Usuario</value>
                <value>br.com.exemploseguranca.model.Perfil</value>
            </list>
        </property>
        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLInnoDBDialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
    </bean>

</beans>
```

spring-security.xml

Nesse arquivo definimos as regras de segurança da aplicação.
O diretório "/paginas/" é protegido e somente usuários autenticados conseguem acessá-lo.
Depois de autenticado, temos acesso às informações de autorização do usuário.
Nesse exemplo temos 2 perfis(roles) de usuário:
* Administrador (ROLE_ADM) -> possui acesso à todos os diretórios da aplicação.
* Usuário (ROLE_USER) -> possui acesso limitado ao diretório "/paginas/usuario/**".
O acesso ao diretório "/paginas/usuario/**" é permitido aos usuários dos 2 perfis.

O fluxo é o seguinte:
O usuário acessa a aplicação e é direcionado para a página "/login.jsf".
Após realizar o login ele é direcionado para "/paginas/inicio.jsf", onde encontra links para as páginas internas ("/paginas/admin/admin.jsf" e "/paginas/usuario/usuario.jsf").
A página "inicio.jsf" é liberada para qualquer usuário autenticado.

Caso ocorra algum erro na autenticação(dados incorretos), é apresentada uma mensagem de erro na própria página "/login.jsf".
Caso um usuário não autorizado(ROLE_USER) tente acessar o diretório "/paginas/admin/**", será direcionado para a página "/acessonegado.jsf"

```
<?xml version="1.0" encoding="UTF-8"?>

<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-3.0.3.xsd">

    <sec:http auto-config="true" access-denied-page="/acessonegado.jsf">

        <sec:intercept-url pattern="/login*" access="IS_AUTHENTICATED_ANONYMOUSLY" />
        <sec:intercept-url pattern="/paginas/inicio*" access="ROLE_USER, ROLE_ADM" />
        <sec:intercept-url pattern="/paginas/usuario/**" access="ROLE_USER, ROLE_ADM" />
        <sec:intercept-url pattern="/paginas/admin/**" access="ROLE_ADM" />

        <sec:form-login login-page="/login.jsf"
                        login-processing-url="/j_spring_security_check"
                        default-target-url="/paginas/inicio.jsf"
                        authentication-failure-url="/login.jsf" />

        <sec:logout logout-success-url="/login.jsf" />

    </sec:http>

    <sec:authentication-manager>
        <sec:authentication-provider user-service-ref="hibernateUserDetailsService" ref="daoAuthenticationProvider" />
    </sec:authentication-manager>

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

    <bean id="loggerListener" class="org.springframework.security.access.event.LoggerListener" />

</beans>
```

HibernateUserDetailsService.java

A autenticação é feita pela interface UserDetailsService.
Para utilizar o hibernate, criamos uma implementação de UserDetailsService e sobrescrevemos o método loadUserByUsername();

```
package br.com.exemploseguranca.services;

import java.util.List;


import org.hibernate.SessionFactory;
import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Restrictions;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
```

```
15
16 import br.com.exemploseguranca.model.Usuario;
17
18 @Service("hibernateUserDetailsService")
19 public class HibernateUserDetailsService extends HibernateDaoSupport implements UserDetailsService  {
20
21     @Autowired
22     public HibernateUserDetailsService(SessionFactory sessionFactory) {
23         setSessionFactory(sessionFactory);
24     }
25
26     @SuppressWarnings("unchecked")
27     @Override
28     public UserDetails loadUserByUsername(String login) {
29         DetachedCriteria criteria = DetachedCriteria.forClass(Usuario.class, "usuario");
30         criteria.add(Restrictions.eq("usuario.login", login));
31         List resultado = getHibernateTemplate().findByCriteria(criteria);
32         if(resultado != null && resultado.size() == 0) {
33             throw new UsernameNotFoundException("Usuário não encontrado!");
34         }
35         return (Usuario)resultado.get(0);
36     }
37
38 }
```

LoginMB.java
Managed Bean que recebe a requisição da página “login.jsf”

```
01 package br.com.exemploseguranca.web.controle;
02
03 import javax.faces.context.FacesContext;
04 import javax.servlet.RequestDispatcher;
05
06
07 import org.springframework.stereotype.Controller;
08
09 import br.com.exemploseguranca.web.util.FacesUtil;
10
11 @Controller("loginMB")
12 public class LoginMB {
13
14     public LoginMB() {
15     }
16
17     public String logar() {
18         try {
19             RequestDispatcher dispatcher = FacesUtil.getServletRequest().getRequestDispatcher("/j_spring_security_check");
20             dispatcher.forward(FacesUtil.getServletRequest(), FacesUtil.getServletResponse());
21             FacesContext.getCurrentInstance().responseComplete();
22         } catch (Exception ex) {
23             FacesUtil.exibirMensagemErro(ex.getMessage());
24             return null;
25         }
26         return null;
27     }
28 }
```

LoginErrorPhaseListener.java
PhaseListener utilizado para capturar exceções de autenticação. Utiliza a classe utilitária FacesUtil para enviar mensagens de erro.

```
01 package br.com.exemploseguranca.web.seguranca;
02
03 import javax.faces.event.PhaseEvent;
04 import javax.faces.event.PhaseId;
05 import javax.faces.event.PhaseListener;
06
07
08 import org.springframework.security.authentication.BadCredentialsException;
09 import org.springframework.security.web.WebAttributes;
10
11 import br.com.exemploseguranca.web.util.FacesUtil;
12
13 @SuppressWarnings({"serial"})
14 public class LoginErrorPhaseListener implements PhaseListener {
15
16     @Override
17     public void afterPhase(PhaseEvent arg0) {
18     }
19
20     @SuppressWarnings({ "unchecked" })
21     @Override
22     public void beforePhase(PhaseEvent arg0) {
23         Exception dadosIncorretosException = (Exception) FacesUtil.getSessionMap().get(WebAttributes.AUTHENTICATION_EXCEPTION);
24         if(dadosIncorretosException instanceof BadCredentialsException) {
25             FacesUtil.getSessionMap().put(WebAttributes.AUTHENTICATION_EXCEPTION, null);
26             FacesUtil.exibirMensagemErro("Dados incorretos!");
27         }
28     }
29
30     @Override
31     public PhaseId getPhaseId() {
32         return PhaseId.RENDER_RESPONSE;
33     }
34 }
```

FacesUtil.java
Classe utilitária para desenvolvimento JSF.

```
01 package br.com.exemploseguranca.web.util;
02
03 import java.util.Map;
04
05 import javax.faces.application.FacesMessage;
06 import javax.faces.context.ExternalContext;
07 import javax.faces.context.FacesContext;
08 import javax.servlet.ServletContext;
09 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
```

```java

/**
 * Classe utilitária para desenvolvimento JSF
 */
public class FacesUtil {


    public static String getRequestParameter(String name) {
        return (String)FacesContext.getCurrentInstance().getExternalContext().getRequestParameterMap().get(name);
    }

    public static void exibirMensagemSucesso(String mensagem) {
        exibirMensagem(FacesMessage.SEVERITY_INFO, mensagem);
    }

    public static void exibirMensagemAlerta(String mensagem) {
        exibirMensagem(FacesMessage.SEVERITY_WARN, mensagem);
    }

    public static void exibirMensagemErro(String mensagem) {
        exibirMensagem(FacesMessage.SEVERITY_ERROR, mensagem);
    }

    private static void exibirMensagem(FacesMessage.Severity severity, String mensagem) {
        FacesMessage facesMessage = new FacesMessage(severity, "", mensagem);
        FacesContext.getCurrentInstance().addMessage(null, facesMessage);
    }

    public static ExternalContext getExternalContext() {
        return FacesContext.getCurrentInstance().getExternalContext();
    }

    public static Map getSessionMap() {
        return FacesContext.getCurrentInstance().getExternalContext().getSessionMap();
    }

    public static ServletContext getServletContext() {
        return (ServletContext)FacesContext.getCurrentInstance().getExternalContext().getContext();
    }

    public static HttpServletRequest getServletRequest() {
        return (HttpServletRequest)FacesContext.getCurrentInstance().getExternalContext().getRequest();
    }

    public static HttpServletResponse getServletResponse() {
        return (HttpServletResponse)FacesContext.getCurrentInstance().getExternalContext().getResponse();
    }


}
```

Usuario.java

A entidade Usuario implementa a interface UserDetails e deve ter o método getAuthorities().
Foi criada uma tabela de associação (usuario_perfil) para armazenar o perfil de acesso(autorização) dos usuários.

```java
package net.finottisistemas.modelo;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.GrantedAuthorityImpl;
import org.springframework.security.core.userdetails.UserDetails;

@Entity
@Table(name = "usuario")
public class Usuario implements java.io.Serializable, UserDetails {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_usuario")
    private Long id;

    @Column(name = "login")
    private String login;

    @Column(name = "senha")
    private String senha;

    private boolean ativo = true;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "usuario_perfil", joinColumns = @JoinColumn(name = "id_usuario"), inverseJoinColumns = @JoinColumn(name = "id_perfil"))
    private List<Perfil> perfis = new ArrayList<Perfil>();

    @Transient
    public Collection<GrantedAuthority> getAuthorities() {
        List<GrantedAuthority> lista = new ArrayList<GrantedAuthority>();
        for(Perfil perfil : getPerfis()) {
            lista.add(new GrantedAuthorityImpl(perfil.getAuthority()));
        }
        return lista;
    }

    @Transient
    public String getPassword() {
```

```
        return this.senha;
    }

    @Transient
    public String getUsername() {
        return this.login;
    }

    @Transient
    public boolean isAccountNonExpired() {
        return true;
    }

    @Transient
    public boolean isAccountNonLocked() {
        return true;
    }

    @Transient
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Transient
    public boolean isEnabled() {
        return this.ativo;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public boolean isAtivo() {
        return ativo;
    }

    public void setAtivo(boolean ativo) {
        this.ativo = ativo;
    }

    public List<Perfil> getPerfis() {
        return perfis;
    }

    public void setPerfis(List<Perfil> perfis) {
        this.perfis = perfis;
    }

}
```

Perfil.java
A entidade Perfil implementa a interface GrantedAuthority.

```
package net.finottisistemas.modelo;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.persistence.Transient;

import org.springframework.security.core.GrantedAuthority;

@Entity
@Table(name = "perfil")
public class Perfil implements java.io.Serializable, GrantedAuthority {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id_perfil")
    private Long id;

    @Column(name = "descricao")
    private String descricao;

    @ManyToMany(fetch=FetchType.EAGER)
    @JoinTable(name = "usuario_perfil", joinColumns = @JoinColumn(name = "id_perfil"), inverseJoinColumns = @JoinColumn(name = "id_usuario"))
    private List<Usuario> usuarios = new ArrayList<Usuario>();


    @Transient
    public String getAuthority() {
        return this.descricao;
    }

    @Transient
    public int compareTo(Object o) {
        return this.compareTo(o);
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public List<Usuario> getUsuarios() {
        return usuarios;
    }

    public void setUsuarios(List<Usuario> usuarios) {
        this.usuarios = usuarios;
    }
}
```

Conexao.java

Propriedades da conexão com o banco de dados (datasource)

```
package br.com.exemploseguranca.config;

import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.stereotype.Component;


@Component("dataSource")
public class Conexao extends DriverManagerDataSource {

    public Conexao(){
        this.setDriverClassName("com.mysql.jdbc.Driver");
        this.setUrl("jdbc:mysql://localhost/seguranca");
        this.setUsername("root");
        this.setPassword("root");
    }

}
```

login.xhtml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:p="http://primefaces.prime.com.tr/ui"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

   <h:head>

       <title>
            Spring + Security + Hibernate
       </title>

       <link type="text/css" rel="stylesheet" href="#{facesContext.externalContext.requestContextPath}/css/redmond/skin.css" />

   </h:head>

   <h:body>

       <h:form prependId="false">

            <p:dialog header="Área restrita"
                      modal="true"
                      closable="false"
                      position="center"
                      widgetVar="modalLogin"
                      minWidth="300"
                      width="300"
                      showEffect="slide"
                      draggable="false"
                      resizable="false"
                      visible="true">

               <center>

                    <p:messages id="mensagens" showDetail="true" showSummary="false" />

                    <h:panelGrid columns="2">

                        <h:outputLabel value="Login" />
                        <h:inputText id="j_username" size="15" />

                        <h:outputLabel value="Senha" />
                        <h:inputSecret id="j_password" size="15" />

                    </h:panelGrid>

                    <br />

                    <h:commandButton value="Entrar" action="#{loginMB.logar}" />

               </center>

            </p:dialog>

       </h:form>

    </h:body>

</html>
```

/paginas/inicio.xhtml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
03
04 <html xmlns="http://www.w3.org/1999/xhtml"
05       xmlns:h="http://java.sun.com/jsf/html"
06       xmlns:f="http://java.sun.com/jsf/core"
07       xmlns:ui="http://java.sun.com/jsf/facelets"
08       xmlns:p="http://primefaces.prime.com.tr/ui">
09
10    <h:head>
11
12        <title>
13             Spring + Security + Hibernate
14        </title>
15
16        <link href="#{facesContext.externalContext.requestContextPath}/css/redmond/skin.css" rel="stylesheet" type="text/css" />
17
18    </h:head>
19
20    <h:body>
21
22    <h:form prependId="false">
23
24         <p:dialog header="Spring + Security + Hibernate"
25                   modal="true"
26                   closable="false"
27                   position="center"
28                   widgetVar="modalInicio"
29                   width="500"
30                   height="400"
31                   draggable="false"
32                   resizable="false"
33                   visible="true">
34
35            <center>
36                <img src="#{facesContext.externalContext.requestContextPath}/img/spring_security.jpg" />
37            </center>
38
39            <br />
40            <br />
41            <br />
42
43            <a href="#{facesContext.externalContext.requestContextPath}/paginas/admin/admin.jsf">
44                Página do administrador
45            </a>
46            <br />
47            <a href="#{facesContext.externalContext.requestContextPath}/paginas/usuario/usuario.jsf">
48                Página do usuário
49            </a>
50            <br />
51            <br />
52            <a href="#{facesContext.externalContext.requestContextPath}/j_spring_security_logout">
53                Sair
54            </a>
55
56        </p:dialog>
57
58     </h:form>
59
60     </h:body>
61
62 </html>
```

/paginas/admin/admin.xhtml

```
01 <?xml version="1.0" encoding="ISO-8859-1"?>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03
04 <html xmlns="http://www.w3.org/1999/xhtml"
05       xmlns:h="http://java.sun.com/jsf/html"
06       xmlns:f="http://java.sun.com/jsf/core"
07       xmlns:ui="http://java.sun.com/jsf/facelets"
08       xmlns:p="http://primefaces.prime.com.tr/ui">
09
10    <h:head>
11
12        <title>
13             Spring + Security + Hibernate
14        </title>
15
16        <link href="#{facesContext.externalContext.requestContextPath}/css/redmond/skin.css" rel="stylesheet" type="text/css" />
17
18    </h:head>
19
20    <h:body>
21
22         <p:dialog header="Página do administrador"
23                   modal="true"
24                   closable="true"
25                   position="center"
26                   widgetVar="modalAdmin"
27                   minWidth="400"
28                   width="400"
29                   draggable="true"
30                   resizable="true"
31                   visible="true">
32
33            <center>
34
35                <img src="#{facesContext.externalContext.requestContextPath}/img/spring_security.jpg" />
36
37                <br />
38                <br />
39                <br />
40                # Página do Administrador #
41                <br />
42                <br />
43                <br />
44
45                <a href="#{facesContext.externalContext.requestContextPath}/paginas/inicio.jsf">
46                 Voltar
47                </a>
48                <br />
```

```
                <br />
                <a href="#{facesContext.externalContext.requestContextPath}/j_spring_security_logout">
                    Sair
                </a>

            </center>

        </p:dialog>

    </h:body>

</html>
```

/paginas/usuario/usuario.xhtml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.prime.com.tr/ui">

   <h:head>

       <title>
            Spring + Security + Hibernate
       </title>

       <link href="#{facesContext.externalContext.requestContextPath}/css/redmond/skin.css" rel="stylesheet" type="text/css" />

   </h:head>

   <h:body>

        <p:dialog header="Página do Usuário"
                  modal="true"
                  closable="false"
                  position="center"
                  widgetVar="modalUsuario"
                  minWidth="400"
                  width="400"
                  draggable="false"
                  resizable="false"
                  visible="true">

            <center>

                <img src="#{facesContext.externalContext.requestContextPath}/img/spring_security.jpg" />

                <br />
                <br />
                <br />
                # Página do usuário #
                <br />
                <br />
                <br />

                <a href="#{facesContext.externalContext.requestContextPath}/paginas/inicio.jsf">
                 Voltar
                </a>
                <br />
                <br />
                <a href="#{facesContext.externalContext.requestContextPath}/j_spring_security_logout">
                    Sair
                </a>

            </center>

        </p:dialog>

    </h:body>

</html>
```

/acessonegado.xhtml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:p="http://primefaces.prime.com.tr/ui">

   <h:head>

       <title>
            <h:outputText value="Spring + Security + Hibernate" />
       </title>

       <link href="#{facesContext.externalContext.requestContextPath}/css/redmond/skin.css" rel="stylesheet" type="text/css" />

   </h:head>

   <h:body style="font-size: 10pt">

        <p:dialog header="Ops! Acesso Negado!"
                  modal="true"
                  closable="true"
                  position="center"
                  widgetVar="modalNegado"
                  minWidth="500"
                  width="400"
                  showEffect="shake"
                  draggable="true"
                  resizable="true"
                  visible="true">

```

```
            <center>

                <img src="#{facesContext.externalContext.requestContextPath}/img/acesso_negado.gif" />

                <br />
                <br />
                <br />
                # Acesso Negado #
                <br />
                <br />
                <br />

                <a href="#{facesContext.externalContext.requestContextPath}/paginas/inicio.jsf">
                 Voltar
                </a>

            </center>

        </p:dialog>

    </h:body>

</html>
```

seguranca.sql
Script utilizado para o banco de dados (Mysql)

```
SET FOREIGN_KEY_CHECKS=0;
-- ----------------------------
-- Table structure for perfil
-- ----------------------------
DROP TABLE IF EXISTS `perfil`;
CREATE TABLE `perfil` (
  `id_perfil` bigint(20) NOT NULL AUTO_INCREMENT,
  `descricao` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_perfil`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Table structure for usuario
-- ----------------------------
DROP TABLE IF EXISTS `usuario`;
CREATE TABLE `usuario` (
  `id_usuario` bigint(20) NOT NULL AUTO_INCREMENT,
  `ativo` bit(1) NOT NULL,
  `login` varchar(255) DEFAULT NULL,
  `senha` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_usuario`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;

-- ----------------------------
-- Table structure for usuario_perfil
-- ----------------------------
DROP TABLE IF EXISTS `usuario_perfil`;
CREATE TABLE `usuario_perfil` (
  `id_perfil` bigint(20) NOT NULL,
  `id_usuario` bigint(20) NOT NULL,
  KEY `FK57CD23FD9DB9E69B` (`id_perfil`),
  KEY `FK57CD23FD5A9933B9` (`id_usuario`),
  KEY `FK57CD23FD67F47FA6` (`id_perfil`),
  KEY `FK57CD23FDD7B1BC0E` (`id_usuario`),
  KEY `FK57CD23FD28906DBF` (`id_perfil`),
  KEY `FK57CD23FD2A939115` (`id_usuario`),
  CONSTRAINT `FK57CD23FD2A939115` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id_usuario`),
  CONSTRAINT `FK57CD23FD28906DBF` FOREIGN KEY (`id_perfil`) REFERENCES `perfil` (`id_perfil`),
  CONSTRAINT `FK57CD23FD5A9933B9` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id_usuario`),
  CONSTRAINT `FK57CD23FD67F47FA6` FOREIGN KEY (`id_perfil`) REFERENCES `perfil` (`id_perfil`),
  CONSTRAINT `FK57CD23FD9DB9E69B` FOREIGN KEY (`id_perfil`) REFERENCES `perfil` (`id_perfil`),
  CONSTRAINT `FK57CD23FDD7B1BC0E` FOREIGN KEY (`id_usuario`) REFERENCES `usuario` (`id_usuario`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- ----------------------------
-- Records
-- ----------------------------
INSERT INTO `perfil` VALUES ('1', 'ROLE_ADM');
INSERT INTO `perfil` VALUES ('2', 'ROLE_USER');
INSERT INTO `usuario` VALUES ('1', true, 'admin', '12345');
INSERT INTO `usuario` VALUES ('2', true, 'user', '12345');
INSERT INTO `usuario_perfil` VALUES ('1', '1');
INSERT INTO `usuario_perfil` VALUES ('2', '2');
```

O código fonte do projeto está disponível no github e no 4shared.

Seja o primeiro a gostar disso post.

Comentários (40)

40 Comentários »

1. 

   Está dando problemas nos imports

   import javax.servlet.ServletContext;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;
   import javax.servlet.RequestDispatcher;

   Será que está com todas as dependências?
   O que pode ser?

Comentário por Pedro — 03/01/2011 @ 21:56

Responder

- 

  Está com todas as dependências sim.
  Esses imports são das libs do container web (no exemplo utilizei o Tomcat 6).
  Caso esteja utilizando alguma IDE como o Eclipse, adicione a library do seu servidor configurado.
  Caso contrário, copie os arquivos jar da pasta TOMCAT_HOME/lib para o diretório WEB-INF/lib do projeto.

  Comentário por guilhermefinotti — 03/01/2011 @ 22:00

  Responder

2. 

   Excelente Post, Guilherme. Parabéns!

   Comentário por Rafael Francelino — 04/01/2011 @ 14:27

   Responder

3. 

   Vc esta usando o NetBeans ou Eclipse?

   Comentário por Luciano — 12/02/2011 @ 22:09

   Responder

   - 

     Eclipse

     Comentário por guilhermefinotti — 12/02/2011 @ 22:26

     Responder

4. 

   Parabéns pelo post. Muito bom! Tenho uma dúvida: na classe HibernateUserDetailsService.java, no método public UserDetails loadUserByUsername(String login) recebo apenas o login. Como faço para receber tbm a senha? Um abraço!

   Comentário por Donizete Waterkemper — 25/02/2011 @ 16:37

   Responder

   - 

     Isso é uma particularidade do framework Spring Security.
     Para utilizá-lo, vc deve usar ou sobrescrever o método loadUserByUsername(String username).
     http://static.springsource.org/spring-security/site/apidocs/index.html

     Comentário por guilhermefinotti — 25/02/2011 @ 16:44

     Responder

     - 

       Eu conseguiria, via Spring, ter o bean Usuario carregado na classe HibernateUserDetailsService? Isso tbm resolveria o meu problema.

       Comentário por Donizete Waterkemper — 25/02/2011 @ 18:53

5. 

   Parabéns pelo tutorial! Uma dúvida:

   Você está usando FetchType.EAGER no @ManyToMany, o que não é uma boa prática. Trocando para LAZY, acontece o famigerado "LazyInitializationException".

   Tentei de tudo que consegui para resolver, e nada adiantou, tu já passou pelo mesmo, sabe como resolver?

   Grato,
   Uilian.

   @ManyToMany(fetch = FetchType.LAZY)
   @JoinTable(name = "usuario_perfil", joinColumns = @JoinColumn(name = "id_usuario"), inverseJoinColumns = @JoinColumn(name = "id_perfil"))
   private List perfis = new ArrayList();

   Comentário por Uilian — 10/04/2011 @ 16:42

   Responder

6. 

   Uilian, vc está certo. Realmente o FetchType.EAGER não é uma boa prática. Mas o objetivo do tutorial é apenas demonstrar a integração dos frameworks. Não deve ser utilizado como base para aplicações "sérias".
   Quanto ao problema do LazyInitializeException, é possível resolvê-lo com o padrão OpenSessionInViewFilter. Aqui tem um exemplo http://www.guj.com.br

/java/233706-opensessioninview-do-spring-resolvido#1202712

[]‘s

Comentário por guilhermefinotti — 10/04/2011 @ 23:13

Responder

7. Boa noite Guilherme!! Seu post está SENSACIONAL. Consegui utilizá-lo perfeitamente. Que DEUS abençoe você. Parabéns. Obrigado!!

   Comentário por Jonas — 14/04/2011 @ 22:34

   Responder

   - Valeu Jonas!
     Obrigado!

     Comentário por guilhermefinotti — 15/04/2011 @ 9:10

     Responder

8. boa noite,

   sou iniciante e está apresentando o seguinte erro para varias importações do pacote javax.persistence.Table , column etc… esta faltando alguma biblioteca ?
   Estou utilizando o netbeans 6.9.1 e Java: 1.6.0_24
   obrigado

   Comentário por James Santos — 30/04/2011 @ 21:58

   Responder

   - Olá James, não está faltando nenhuma biblioteca.
     O arquivo que contém esse pacote é o ejb3-persistence.jar.
     Verifique se ele está adicionado na configuração do seu projeto.
     []‘s

     Comentário por guilhermefinotti — 01/05/2011 @ 15:39

     Responder

9. boa noite, adicionei o jar que voce me mostrou e desapareceu os erros de referencia.
   Agora estou com o seguinte problema: You cannot use a spring-security-2.0.xsd or spring-security-3.0.xsd schema with Spring Security 3.1.
   Please update your schema declarations to the 3.1 schema.

   Tentei alterar apenas o schema para 3.1.xsd, mas começou a apresentar um novo erro: Configuration problem: authentication-provider element cannot be used with other attributes when using ‘ref’ attribute
   voce sabe como posso resolver
   obrigado.

   Comentário por James Santos — 01/05/2011 @ 21:54

   Responder

   - O exemplo foi feito com o spring security 3.0.5. Não fiz nenhum teste com a versão 3.1.
     No fim do post tem o link para baixar o projeto com todas as bibliotecas necessárias.
     []‘s

     Comentário por guilhermefinotti — 02/05/2011 @ 0:12

     Responder

10. Guilherme bom dia, estou tentando colocar o security na minha “Aplicação” tô tentando aprender o Java… bem
    estou usando tomcat 7, JSF 2 Spring e o Security 3 e JPA 2 não estou usando o Hibernate mais o EclipseLink
    bem a minha aplicação não apresenta erros, mais não consigo autenticar ou seja recebo a mensagem que o
    usuario ou a senha são invalidos…. embora eu sei que esta certo, pelo que vi de diferente do seu projeto é que
    na entidade usuario eu não implementei a Interface UserDetails, fiz o trabalho na implementação da classe
    UserDetailsService será que este foi o meu Erro? eu instanciei o usuario la e no metodo loadUserByUsername
    obtive o objeto usuario Usuario u = usuarioDao.findByLogin(username)
    criei o objeto user do spring (org.springframework.security.core.userdetails.User user)
    … security.core.userdetails.User(u.getUsrLogin(), u.getUsrSenha(), enabled, …..
    Tambem fiz um laço para pegar as informações de papeis
    for (Papel role : papels) {
    grantedAuthorities.add(new GrantedAuthorityImpl(role.getPaPapel()));
    }

Bem eu tô achando que esta classe não esta sendo executada… porque eu coloquei um println
como forma de depurar (desculpem-me é que eu nunca fiz um teste) mais não chego a ver nada
mais também não há erros

Será que vc poderia comentar ?

Comentário por Robson — 24/10/2011 @ 12:43

Responder

- 

  Bom dia Robson. Não consegui enxergar nada de errado em sua aplicação. Inclusive, me disseram que é mais recomendado fazer a implementação da segurança fora da entidade.
  Verifique seu arquivo de configuração (spring-config.xml) e sua tabela “Roles”, pra confirmar que existe um papel para o usuário que está tentando logar.
  Estou no trabalho agora e não tenho como testar. Assim que chegar em casa tento te ajudar.
  Abraço

  Comentário por guilhermefinotti — 24/10/2011 @ 13:01

  Responder

  - 

    Prezado obrigado mesmo , bem so por ter comentado algo já ajuda… sabe como é dificil vir do MS-VFP pegar Java pela frente…
    mais então no arquivo de configuração eu não coloquei nenhum papel so deixei configurado para autenticado

    Será que tá ai o problema. ? já tô olhando a documentação do Spring… mais ainda to meio achando que o problema pode ser outro.

    Comentário por Robson — 24/10/2011 @ 15:10

    - 

      Guilherme bom dia!
      Ontem passei uma revista no projeto, e realmente não consegui saber o que esta ocorrendo, olhei o xml do spring e os dados da tabela Papel como vc sugeriu, ta tudo ok , notei que o UsuarioDao que é onde eu vou obter a informação
      do usuario eu coloquei um “Alerta” antes e depois da consulta e no Eclipse na area onde vemos o que esta sendo processado
      so aparece o primeiro Alerta (dizendo que entrou no metodo) ja o segundo alerta não é mostrado, como se o programa parasse ali mais não aparece nenhum erro o que é ruim porque se houvesse erro eu corria atras,
      Olha sei que é dificil estamos no trabalho, mais se vc tiver alguma dica pode me contactar no meu E-mail se for possivel
      robsonlira_pe@hotmail.com

      Um abraço

      Comentário por Anônimo — 25/10/2011 @ 12:27

11. 

    Caro Guilherme boa tarde, cara ontem fiz o que vc pediu olhei o arquivo do Spring, a Tabela Papel, o arquivo de Serviço e o UsuarioDao, mais não vi nda, no UsuarioDao coloquei um alerta antes e depois da consulta. ao executar no Eclipse eu posso ver que qdo o metodo findByLogin o primeiro alerta antes da consulta aparece no monito, já o segundo não ou seja o Repositorio esta sendo chamado mais … realmente não sei o que há, o bom seria que houvesse erro pra correr atras.
    Se vc puder sugerir alguma dica, ou melhor outra dica agradeço.

    Um Abraço
    Robson Lira
    robsonlira_pe@hotmail.com

    Comentário por Robson — 25/10/2011 @ 13:33

    Responder

12. 

    Caro Róbson tenho um template de Facelets e o mesmo gera o seguinte erro no stacktrace:

    Invalid path : /exemploseguranca/paginas/template.jsf

    tirando isso roda perfeitamente o q pode ser feito?

    Comentário por Luxu — 28/10/2011 @ 0:20

    Responder

    - 

      Bom dia Luxu.
      Troque a extensão do arquivo para .xhtml que funciona.
      /exemploseguranca/paginas/template.xhtml
      []‘s
      Guilherme

      Comentário por guilhermefinotti — 28/10/2011 @ 7:57

      Responder

Funciona naum, pq o problema são com todos componentes pq comentando os mesmos roda sem problemas sak? é como se o spring não se desse com esses componentes…

Comentário por Luxu — 28/10/2011 @ 8:01

Estranho esse problema. Nunca ouvi falar dessa restrição. Vc está usando JSF 2.0?

Comentário por guilhermefinotti — 28/10/2011 @ 8:10

Estou sim tenho uma página com menu e sem componente aparece certinho, bem estranho msm!

Comentário por Luxu — 28/10/2011 @ 8:16

Vou tentar fazer uns testes e reproduzir esse problema.
Se encontrar alguma solução, favor me avisar, ok?

Comentário por guilhermefinotti — 28/10/2011 @ 8:21

OK, postarei aki e vc saberá, vlw por enquanto!

Comentário por Luxu — 28/10/2011 @ 8:24

13. 

Boa noite amigo, meu nome é Luiz Paulo.

Você sabe se consigo usar este exemplo em u projeto EJB?

Comentário por Anônimo — 14/01/2012 @ 21:31

Responder

Olá Luiz Paulo.
Vc consegue usar esse exemplo num projeto EJB.
Aqui tem um exemplo http://forum.springsource.org/showthread.php?115626-Spring-Security-3-credentials-from-context-to-access-remote-EJB

Comentário por guilhermefinotti — 15/01/2012 @ 21:24

Responder

14. 

Olá amigo, meu nome é Lucas,

Primeiramente parabéns pelo post porque pra quem é iniciante em spring-security esse tutorial foi uma verdadeira mão na roda.
Bem no projeto que to fazendo eu preciso usar os facelets do spring-security. tentei usar em um menu (pra renderizar ou nao) e deu erro no porque ele pede um método Authority().. no caso do seu projeto como seria o uso desses facelets?

valeu pelo tuto!

Comentário por Lucas Feitozas — 01/03/2012 @ 18:31

Responder

Boa tarde Lucas!
Que bom que o tutorial foi útil pra você. Para utilizar spring security + facelets, sugiro a utilização da seguinte taglib (http://www.dominikdorn.com/facelets/). No final da página tem um exemplo.
Pretendo arrumar mais tempo para escrever novos tutoriais e "spring security + facelets" é um bom tema.
Inclusive, se você conseguir fazer essa integração e quiser disponibilizar aqui no blog é só avisar.
[]'s

Comentário por guilhermefinotti — 02/03/2012 @ 15:24

Responder

Opa, eu usei as taglibs deste site e + configurei o springsecurity.taglib.xml(http://static.springsource.org/spring-webflow/docs/2.3.x/reference/html/ch13s11.html) e enfim.. ele me retornou o seguinte erro:

Grave: Critical error during deployment:
com.sun.faces.config.ConfigurationException: java.lang.ClassNotFoundException:
org.springframework.faces.security.FaceletsAuthorizeTagHandler
…. 15More

procurei um material na internet e nao achei.. meu minha configuraçao do spring-security.xml está igual a sua.. (apenas adequada ao projeto que estou usando).. se vc puder me ajudar seria mto bom.. qualquer coisa posso te passar como eu estou usando isso ai..

=)

Comentário por Lucas Feitozas — 02/03/2012 @ 15:41

15.

Bom tarde … estou tentando resolver um problema que pode ate ser simples.. mas não estou conseguindo….
como eu faço para recuperar a sessão em outras partes do programa para usar o Hibernate, por exemplo para criar uma criteira…
onde eu recupero a sessao… pois tentei criar uma nova mas dai da erro!!

Comentário por Helinton Veiverber — 03/03/2012 @ 15:06

Responder

-

  Pesquisa sobre HibernateUtil e o o EntityManagerUtil (getSession())

  Comentário por Lucas Feitozas — 03/03/2012 @ 15:17

  Responder

-

  Helinton, para utilizar a sessão do hibernate em outras classes, basta que essas classes extendam HibernateDaoSupport. A classe HibernateUserDetailsService é um exemplo.
  Outro exemplo pode ser visto na classe UsuarioDAO desse post (http://javafaces.wordpress.com/2010/12/02/exemplo-jsf-2-hibernate-3-spring-3/)
  []'s

  Comentário por guilhermefinotti — 03/03/2012 @ 15:18

  Responder

  -

    Já havia tentado isso fiz a classe da seguinte maneira
    *****Classe******

    public class teste extends HibernateDaoSupport{

    /** Creates a new instance of teste */
    public teste() {
    }

    public List getEnderecos() {
    List ends = getHibernateTemplate().loadAll(Endereco.class);
    return ends;
    }
    }

    e me da o seguinte erro na hora que invoco o metodo getEnedreco para litar os mesmo em uma tabela jsf

    HTTP Status 500 -

    type Exception report

    message

    description The server encountered an internal error () that prevented it from fulfilling this request.

    exception

    javax.servlet.ServletException
    javax.faces.webapp.FacesServlet.service(FacesServlet.java:606)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:368)
    org.springframework.security.web.access.intercept.FilterSecurityInterceptor.invoke(FilterSecurityInterceptor.java:109)
    org.springframework.security.web.access.intercept.FilterSecurityInterceptor.doFilter(FilterSecurityInterceptor.java:83)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:380)
    org.springframework.security.web.access.ExceptionTranslationFilter.doFilter(ExceptionTranslationFilter.java:97)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:380)
    org.springframework.security.web.session.SessionManagementFilter.doFilter(SessionManagementFilter.java:100)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:380)
    org.springframework.security.web.authentication.AnonymousAuthenticationFilter.doFilter(AnonymousAuthenticationFilter.java:78)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:380)
    org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter.doFilter(SecurityContextHolderAwareRequestFilter.java:54)
    org.springframework.security.web.FilterChainProxy$VirtualFilterChain.doFilter(FilterChainProxy.java:380)
    org.springframework.security.web.savedrequest.RequestCacheAwareFilter.doFilter(RequestCacheAwareFilter.java:35)