**Configuring Spring Security to support Group-Based Access Control**

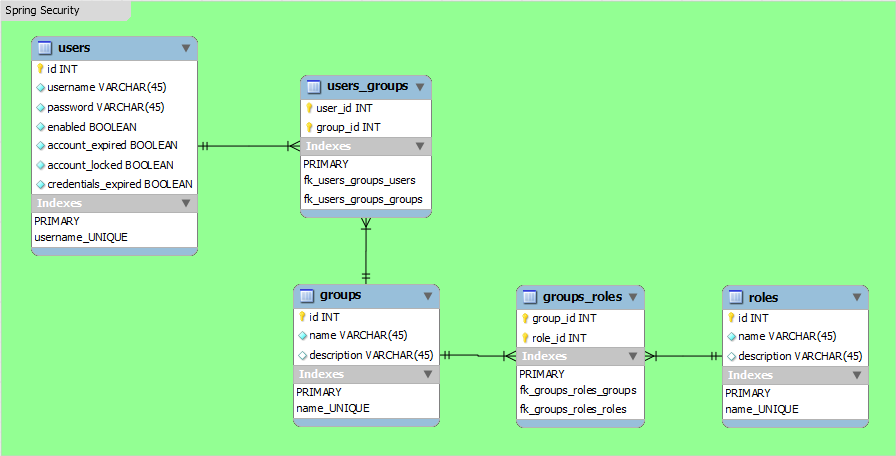
**with a custom database schema and authentication based in Hibernate**

Jose Domingo López López ([lopez.josedomingo@gmail.com](mailto:lopez.josedomingo@gmail.com))

19 May, 2011

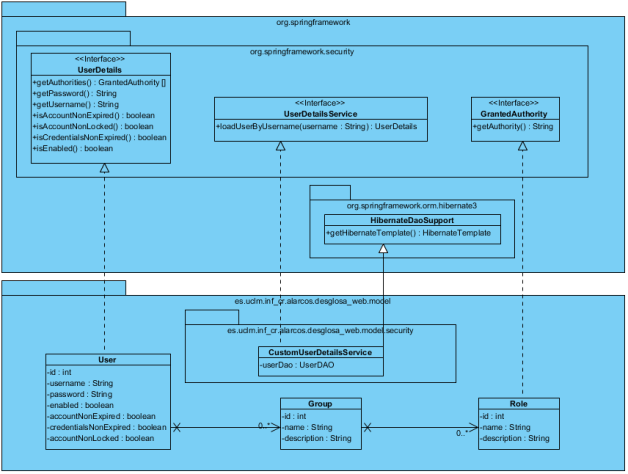
In this tutorial we will learn how to set up Spring Security in order to build a webapp that supports Group-Based Access Control (GBAC). However, it is easy to customize it to support a simple role-based access control. In addition, the authentication will be Hibernate based instead of a simple Jdbc connection.

The first step is to design the database schema. As explained above, we are going to design a GBAC, so we need a *users* table, a *groups* table, a *roles* table, and two additional tables (*users\_groups* and *groups\_roles*) to link them as *many­-to-many­* relationships (Fig. xxx).



To understand the implementation of the Java classes, it is recommended to read the Spring Security documentation about the *UserDetails*, *UserDetailsService* and *GrantedAuthority* interfaces. By the way, Fig. xxx illustrates a class diagram about how to do the required implementation.

Note that if Hibernate support is not desired, but Jdbc support, then *CustomUserDetailsService* should extend from *JdbcDaoImpl* or *JdbcDaoSupport*, whatever are your needs, not from *HibernateDaoSupport*. In this case you would mind to rewrite the *LoadUser****s****ByName*, *CreateUserDetails* and *UsersByUsernameMapping* methods to fit your custom user needs.



The second step is to implement our custom Java classes (*User*, *Group*, *Role*, and *CustomUserDetailsService*) adding their JPA (Java Persistence Api) annotations to map them with the database tables.

|  |
| --- |
| **package** es.uclm.inf\_cr.alarcos.desglosa\_web.model;  **import** java.util.ArrayList;  **import** java.util.HashSet;  **import** java.util.List;  **import** java.util.Set;  **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.GrantedAuthority;  **import** org.springframework.security.userdetails.UserDetails;  @Entity  @Table(name="users")  **public** **class** User **implements** UserDetails {  **private** **int** id;  **private** String username;  **private** String password;  **private** **boolean** enabled;  **private** **boolean** accountNonExpired;  **private** **boolean** credentialsNonExpired;  **private** **boolean** accountNonLocked;  **private** Set<Group> groups = **new** HashSet<Group>();    **public** User() {  }  @Id @GeneratedValue(strategy=GenerationType.*AUTO*)  **public** **int** getId() {  **return** id;  }  @Column(nullable=**false**,length=45,unique=**true**)  **public** String getUsername() {  **return** username;  }  @Column(nullable=**false**)  **public** String getPassword() {  **return** password;  }  @Column(name="enabled")  **public** **boolean** isEnabled() {  **return** enabled;  }  @Transient  **public** **boolean** isAccountNonExpired() {  //return accountNonExpired;  **return** **true**;  }  @Transient  **public** **boolean** isCredentialsNonExpired() {  //return credentialsNonExpired;  **return** **true**;  }  @Transient  **public** **boolean** isAccountNonLocked() {  //return accountNonLocked;  **return** **true**;  }  @ManyToMany(fetch = FetchType.*EAGER*)  @JoinTable(  name="users\_groups",  joinColumns = { @JoinColumn( name="user\_id") },  inverseJoinColumns = @JoinColumn( name="group\_id")  )  **public** Set<Group> getGroups() {  **return** groups;  }  @Transient  **public** GrantedAuthority[] getAuthorities() {  //return authorities;  List<GrantedAuthority> authorities = **new** ArrayList<GrantedAuthority>();  **for** (Group g : **this**.groups) {  authorities.addAll(g.getAuthorities());  }  //return new GrantedAuthority[]{new GrantedAuthorityImpl("ROLE\_ADMIN")};  **return** authorities.toArray(**new** GrantedAuthority[0]);  }    **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** **void** setUsername(String username) {  **this**.username = username;  }  **public** **void** setPassword(String password) {  **this**.password = password;  }  **public** **void** setEnabled(**boolean** enabled) {  **this**.enabled = enabled;  }  **public** **void** setAccountNonExpired(**boolean** accountNonExpired) {  **this**.accountNonExpired = accountNonExpired;  }  **public** **void** setCredentialsNonExpired(**boolean** credentialsNonExpired) {  **this**.credentialsNonExpired = credentialsNonExpired;  }  **public** **void** setAccountNonLocked(**boolean** accountNonLocked) {  **this**.accountNonLocked = accountNonLocked;  }  **public** **void** setGroups(Set<Group> groups) {  **this**.groups = groups;  }  } |

|  |
| --- |
| **package** es.uclm.inf\_cr.alarcos.desglosa\_web.model;  **import** java.util.ArrayList;  **import** java.util.HashSet;  **import** java.util.List;  **import** java.util.Set;  **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.GrantedAuthority;  **import** org.springframework.security.GrantedAuthorityImpl;  @Entity  @Table(name="groups")  **public** **class** Group {  **private** **int** id;  **private** String name;  **private** String description;  **private** Set<Role> roles = **new** HashSet<Role>();    **public** Group() {  }  @Id @GeneratedValue(strategy=GenerationType.*AUTO*)  **public** **int** getId() {  **return** id;  }  @Column(nullable=**false**,length=45,unique=**true**)  **public** String getName() {  **return** name;  }  @Column(nullable=**true**,length=45,unique=**false**)  **public** String getDescription() {  **return** description;  }  @ManyToMany(fetch = FetchType.*EAGER*)  @JoinTable(  name="groups\_roles",  joinColumns = { @JoinColumn( name="group\_id") },  inverseJoinColumns = @JoinColumn( name="role\_id")  )  **public** Set<Role> getRoles() {  **return** roles;  }    @Transient  **public** List<GrantedAuthority> getAuthorities() {  //return authorities;  List<GrantedAuthority> authorities = **new** ArrayList<GrantedAuthority>();  **for** (Role r : **this**.roles) {  authorities.add(**new** GrantedAuthorityImpl(r.getName()));  }  **return** authorities;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **void** setDescription(String description) {  **this**.description = description;  }  **public** **void** setRoles(Set<Role> roles) {  **this**.roles = roles;  }  **public** **int** compareTo(Object o) {  // **TODO** Auto-generated method stub  **return** 0;  }    } |

|  |
| --- |
| **package** es.uclm.inf\_cr.alarcos.desglosa\_web.model;  **import** javax.persistence.Column;  **import** javax.persistence.Entity;  **import** javax.persistence.GeneratedValue;  **import** javax.persistence.GenerationType;  **import** javax.persistence.Id;  **import** javax.persistence.Table;  **import** javax.persistence.Transient;  **import** org.springframework.security.GrantedAuthority;  @Entity  @Table(name="roles")  **public** **class** Role **implements** GrantedAuthority {  **private** **int** id;  **private** String name;  **private** String description;    **public** Role () {  }  @Id @GeneratedValue(strategy=GenerationType.*AUTO*)  **public** **int** getId() {  **return** id;  }  @Column(nullable=**false**,length=45,unique=**true**)  **public** String getName() {  **return** name;  }  @Column(nullable=**true**,length=45,unique=**false**)  **public** String getDescription() {  **return** description;  }    @Transient  **public** String getAuthority() {  **return** getName();  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** **void** setName(String name) {  **this**.name = name;  }  **public** **void** setDescription(String description) {  **this**.description = description;  }    **public** **boolean** equals(Object o) {  **if** (**this** == o) {  **return** **true**;  }  **if** (!(o **instanceof** Role)) {  **return** **false**;  }  **final** Role role = (Role) o;  **return** !(name != **null** ? !name.equals(role.name) : role.name != **null**);  }  **public** **int** compareTo(Object o) {  **return** (equals(o) ? 0 : -1);  }  } |

|  |
| --- |
| **package** es.uclm.inf\_cr.alarcos.desglosa\_web.security;  **import** java.util.List;  **import** org.springframework.dao.DataAccessException;  **import** org.springframework.orm.hibernate3.support.HibernateDaoSupport;  **import** org.springframework.security.userdetails.UserDetails;  **import** org.springframework.security.userdetails.UserDetailsService;  **import** org.springframework.security.userdetails.UsernameNotFoundException;  **public** **class** CustomUserDetailsService **extends** HibernateDaoSupport **implements** UserDetailsService {    **public** UserDetails loadUserByUsername(String username) **throws** UsernameNotFoundException, DataAccessException {  List users = getHibernateTemplate().find("from User where username=?", username);  **if** (users == **null** || users.isEmpty()) {  **throw** **new** UsernameNotFoundException("user '" + username + "' not found...");  } **else** {  **return** (UserDetails) users.get(0);  }  }  } |

In the third step, the Spring Security’s XML file must be configured. In this case, it is called applicationContext-security.xml (Fig. xxx). It is important to note that:

* *userDetailsService* bean points to our *CustomUserDetailsService*.
* *dataSource* bean specifies the driver, url, user and password database.
* *sessionFactory* bean, which is pointed by *hibernateTemplate* bean, specifies Hibernate configuration. Note that it points to a file named *hibernate.cfg.xml* also. This file maps all hibernate annotated java classes (Fig. xxx).

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans:beans xmlns=*"http://www.springframework.org/schema/security"*  xmlns:beans=*"http://www.springframework.org/schema/beans"*  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-2.0.xsd*  *http://www.springframework.org/schema/security*  *http://www.springframework.org/schema/security/spring-security-2.0.1.xsd"*>  <global-method-security secured-annotations=*"enabled"* />  <http auto-config=*"true"*>  <intercept-url pattern=*"/index.jsp"* filters=*"none"*/>  <intercept-url pattern=*"/login.jsp"* access=*"IS\_AUTHENTICATED\_ANONYMOUSLY"*/>  <intercept-url pattern=*"/logout.jsp"* access=*"ROLE\_ADMIN,ROLE\_EXECUTIVE,ROLE\_MANAGER,ROLE\_USER"*/>  <intercept-url pattern=*"/jsp/admin/\*"* access=*"ROLE\_ADMIN"*/>  <intercept-url pattern=*"/jsp/myAccount.jsp"* access=*"ROLE\_ADMIN,ROLE\_EXECUTIVE,ROLE\_MANAGER,ROLE\_USER"*/>  <intercept-url pattern=*"/jsp/visualization.jsp"* access=*"ROLE\_ADMIN,ROLE\_EXECUTIVE,ROLE\_MANAGER,ROLE\_USER"*/>  <intercept-url pattern=*"/\*\*"* access=*"IS\_AUTHENTICATED\_ANONYMOUSLY"*/>  <form-login login-page=*"/login.jsp"* default-target-url=*"/login.jsp?result=success"* always-use-default-target=*"true"* authentication-failure-url=*"/login.jsp?result=failed"* />  <anonymous />  <http-basic />  <logout logout-url=*"/logout.jsp"* logout-success-url=*"/"* invalidate-session=*"true"*/>  <remember-me />  </http>    <authentication-provider user-service-ref=*'userDetailsService'*>  <!-- <password-encoder hash="md5"/> -->  </authentication-provider>  <beans:bean id=*"userDetailsService"* class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.security.CustomUserDetailsService"*>  <beans:property name=*"hibernateTemplate"* ref=*"hibernateTemplate"* />  </beans:bean>    <beans:bean id=*"dataSource"* class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>  <beans:property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />  <beans:property name=*"url"* value=*"jdbc:mysql://localhost:3306/desglosadb"* />  <beans:property name=*"username"* value=*"desglosaadmin"* />  <beans:property name=*"password"* value=*"nimdaasolgsed"* />  </beans:bean>      <!-- HIBERNATE CONFIGURATION -->    <beans:bean id=*"sessionFactory"* class=*"org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"*>  <beans:property name=*"dataSource"* ref=*"dataSource"* />  <beans:property name=*"configLocation"* value=*"classpath:hibernate.cfg.xml"*/>  <!-- Annotated classes are mapped in hibernate.cfg.xml -->  <beans:property name=*"hibernateProperties"*>  <beans:props>  <beans:prop key=*"hibernate.dialect"*>org.hibernate.dialect.MySQLDialect</beans:prop>  <beans:prop key=*"hibernate.show\_sql"*>true</beans:prop>  <beans:prop key=*"hibernate.hbm2ddl.auto"*>update</beans:prop>  </beans:props>  </beans:property>  </beans:bean>    <beans:bean id=*"hibernateTemplate"* class=*"org.springframework.orm.hibernate3.HibernateTemplate"*>  <beans:property name=*"sessionFactory"* ref=*"sessionFactory"* />  </beans:bean>    <beans:bean id=*"userDao"* class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.dao.hibernate.UserDAOHibernate"*>  <beans:constructor-arg value=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.User"*/>  <beans:property name=*"hibernateTemplate"* ref=*"hibernateTemplate"* />  </beans:bean>    <beans:bean id=*"groupDao"* class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.dao.hibernate.GroupDAOHibernate"*>  <beans:constructor-arg value=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.Group"*/>  <beans:property name=*"hibernateTemplate"* ref=*"hibernateTemplate"* />  </beans:bean>    <beans:bean id=*"roleDao"* class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.dao.hibernate.RoleDAOHibernate"*>  <beans:constructor-arg value=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.Role"*/>  <beans:property name=*"hibernateTemplate"* ref=*"hibernateTemplate"* />  </beans:bean>    </beans:beans> |

|  |
| --- |
| <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">  <hibernate-configuration>  <session-factory>  <mapping class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.User"*/>  <mapping class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.Group"*/>  <mapping class=*"es.uclm.inf\_cr.alarcos.desglosa\_web.model.Role"*/>  </session-factory>  </hibernate-configuration> |

The fourth step is to add Spring Security to the *web.xml*. The file web.xml shown in this document shows how to integrate Spring, Spring Security, Internationalization support, Struts2, Struts-menu and SiteMesh. Note that the *filter-mapping* order is critical in order to check get everything finely working together (p.e. *springSecurityFilterChain* before *sitemesh* in to update correctly menus configured with *struts-menu*).

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app id=*"starter"* version=*"2.4"*  xmlns=*"http://java.sun.com/xml/ns/j2ee"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>    <display-name>Struts 2 - Maven Archetype - Starter</display-name>    <!-- Define the basename for a resource bundle for I18N -->  <context-param>  <param-name>javax.servlet.jsp.jstl.fmt.localizationContext</param-name>  <param-value>ApplicationResources</param-value>  </context-param>  <!-- Fallback locale if no bundles found for browser's preferred locale -->  <!-- Force a single locale using param-name 'javax.servlet.jsp.jstl.fmt.locale' -->  <context-param>  <param-name>javax.servlet.jsp.jstl.fmt.fallbackLocale</param-name>  <param-value>en</param-value>  </context-param>  <!-- Context Configuration locations for Spring XML files -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath\*:/applicationContext\*.xml</param-value>  </context-param>    <!-- Filters -->  <filter>  <filter-name>struts-cleanup</filter-name>  <filter-class>org.apache.struts2.dispatcher.ActionContextCleanUp</filter-class>  </filter>  <filter>  <filter-name>sitemesh</filter-name>  <filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>  </filter>  <filter>  <filter-name>action2</filter-name>  <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>  </filter>  <filter>  <filter-name>springSecurityFilterChain</filter-name>  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  </filter>    <!-- The filter-mapping order is critical -->  <filter-mapping>  <filter-name>springSecurityFilterChain</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <filter-mapping>  <filter-name>struts-cleanup</filter-name>  <url-pattern>/\*</url-pattern>  <dispatcher>REQUEST</dispatcher>  <dispatcher>FORWARD</dispatcher>  </filter-mapping>  <filter-mapping>  <filter-name>sitemesh</filter-name>  <url-pattern>/\*</url-pattern>  <dispatcher>REQUEST</dispatcher>  <dispatcher>FORWARD</dispatcher>  <dispatcher>INCLUDE</dispatcher>  </filter-mapping>  <filter-mapping>  <filter-name>action2</filter-name>  <url-pattern>/\*</url-pattern>  <dispatcher>REQUEST</dispatcher>  <dispatcher>FORWARD</dispatcher>  </filter-mapping>  <!-- Listeners -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <listener>  <listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>  </listener>  <listener>  <listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>  </listener>  <!--  - Loads the menu-config.xml for struts-menu at startup,  - by default from "/WEB-INF/menu-config.xml".  - To override this, add a context-param named "menuConfigLocation"  - web.xml file.  -->  <listener>  <listener-class>net.sf.navigator.menu.MenuContextListener</listener-class>  </listener>      <!-- Servlets -->  <servlet>  <servlet-name>dwr</servlet-name>  <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>  <init-param>  <param-name>debug</param-name>  <param-value>true</param-value>  </init-param>  </servlet>  <servlet>  <servlet-name>jspSupportServlet</servlet-name>  <servlet-class>org.apache.struts2.views.JspSupportServlet</servlet-class>  <load-on-startup>5</load-on-startup>  </servlet>    <servlet-mapping>  <servlet-name>dwr</servlet-name>  <url-pattern>/dwr/\*</url-pattern>  </servlet-mapping>    <session-config>  <session-timeout>10</session-timeout>  </session-config>    <!-- Welcome file lists -->  <welcome-file-list>  <welcome-file>index.jsp</welcome-file>  </welcome-file-list>  <error-page>  <error-code>500</error-code>  <location>/error.jsp</location>  </error-page>  <error-page>  <error-code>400</error-code>  <location>/index.jsp</location>  </error-page>  <error-page>  <error-code>403</error-code>  <location>/403.jsp</location>  </error-page>  <error-page>  <error-code>404</error-code>  <location>/404.jsp</location>  </error-page>  </web-app> |

Finally, here is an example of the login page. Note that the action url is */j\_spring\_security\_check*.

|  |
| --- |
| <!-- Evaluate login operation -->  <c:choose>  <c:when test=*"*${param.result == 'success'}*"*>  <p><fmt:message key=*"message.login.sucess"*/>  </br><fmt:message key=*"message.redirect.index"*/></p>  <c:url var=*"url"* value=*"/index.action"*></c:url>  <script> setTimeout("window.location.href='<c:out value="${url}"/>'",5000); </script>  </c:when>  <c:otherwise>  <c:if test=*"*${param.result == 'failed'}*"*>  <p><fmt:message key=*"message.login.fail"*/>  </br><c:out value=*"*${SPRING\_SECURITY\_LAST\_EXCEPTION.message}*"*/>.</pa>  <!-- "Bad credentials" -->  </c:if>  <form method=*"post"* id=*"loginForm"* action=*"*<c:url value=*'/j\_spring\_security\_check'*/>*"* onsubmit="saveUsername(this);return validateForm(this)">  <ul>  <li>  <label for=*"j\_username"*>  <fmt:message key=*"label.username"* />  </label>  <s:textfield id=*"j\_username"* name=*"j\_username"* value=*""* tabindex=*"1"* />  </li>  <li>  <label for=*"j\_password"*>  <fmt:message key=*"label.password"* />  </label>  <s:password id=*"j\_password"* name=*"j\_password"* value=*""* tabindex=*"2"* />  </li>  <li>  <input type=*"submit"* name=*"login"* value=*"*<fmt:message key=*'button.login'*/>*"* tabindex=*"3"* />  </li>  </ul>  </form>  </c:otherwise>  </c:choose> |