Demoiselle Components Guide

Demoiselle Components

Bruno Gutierrez

<bruno.gutierrez@serpro.gov.br>

Sobre o Demoiselle Components

O *Demoiselle Components Guide* agrega em um único lugar toda a documentação referente a todos os componentes disponibilizados e que são compatíveis com a versão mais recente do *Demoiselle Framework*.

Parte I. Demoiselle Login Module

O componente *Demoiselle Login Module* tem como objetivo prover um conjunto de soluções para facilitar o desenvolvimento de rotinas baseadas em segurança na plataforma Java EE. A finalidade é diminuir a complexidade na implementação de mecanismos de segurança relacionados com a autenticação e a autorização de usuários.

Este documento explica como configurar o componente *Demoiselle Login Module* no servidor de aplicações JBoss 6 EAP, e como configurar as aplicações para o uso do componente *Demoiselle Login Module*.

Configuração do Demoiselle Login Module

1.1. Autenticação e Autorização

O *Demoiselle Login Module* define um módulo de login baseado em provedores de segurança. Esse módulo elimina a necessidade do desenvolvimento de um LoginModule para cada aplicação. O desenvolvedor, por sua vez, deve implementar um provedor de segurança. Esse provedor deverá ser responsável pela autenticação e autorização dos usuários de uma aplicação.

O *Demoiselle Login Module* permite que aplicações Java EE utilizem o recurso de autenticação mista no servidor de aplicações Java 6 EAP. A autenticação mista foi implementada através do uso das especificações Servlet 3.0 e JACC 1.0.2. Não será mais necessária nenhuma configuração no arquivo web.xml das aplicações para a definição do tipo de autenticação (FORM, CLIENT-CERT ou FORM-AND-CLIENT-CERT). As aplicações vão optar pelo tipo de autenticação na definição da página login.xhtml.

Um exemplo da página login.xhtml é mostrado na seção "Configuração do Login Module nas aplicações" deste documento.

O *Demoiselle Login Module* também permite que os usuários troquem a senha durante o processo de autenticação.

1.2. Configuração do Login Module no JBoss 6 EAP

Deve ser realizado o download do componente *Demoiselle Login Module* no site do SourceForge.

Neste documento, $JBoss corresponde ao diretório onde está instalado o servidor JBoss 6 EAP.

O componente *Demoiselle Login Module* baixado deve ser copiado para a pasta $JBoss/modules/br/gov/frameworkdemoiselle/loginmodule/provider/main. Se a pasta não existir, a mesma deve ser criada.

Deve ser criado o arquivo $JBoss/modules/br/gov/frameworkdemoiselle/loginmodule/provider/main/module.xml. Neste arquivo, devem ser declarados o arquivo .jar do *Demoiselle Login Module* e as suas dependências. Segue abaixo um exemplo do arquivo module.xml:

```
        <?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.0" name="br.gov.frameworkdemoiselle.loginmodule.provider">
    <resources>
        <resource-root path="demoiselle-loginmodule-1.0.0.jar"/>
        ...
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.security.jacc.api"/>
```

```
        <module name="javax.faces.api"/>

        ...

    </dependencies>

</module>
```

Deve ser adicionado um domínio de segurança da aplicação em $JBoss/standalone/configuration/standalone.xml no subsistema urn:jboss:domain:security, conforme o exemplo abaixo:

```
        <security-domain name="demoiselle-loginmodule-domain" cache-type="default">

          <authentication>

                                                                                <login-
module code="br.gov.frameworkdemoiselle.loginmodule.provider.ProviderLoginModule"
                  flag="sufficient" module="br.gov.frameworkdemoiselle.loginmodule.provider">

                                                <module-option   name="authentication-provider-
class" value="br.gov.serpro.security.provider.authentication.SampleAuthenticationProvider"/>

                                                 <module-option   name="authorization-provider-
class" value="br.gov.serpro.security.provider.authentication.SampleAuthorizationProvider"/>

                                                       <module-option   name="throwable-handler-
value="br.gov.frameworkdemoiselle.loginmodule.message.implementation.RequestThrowableHandler"/
>

                  <module-option name="system" value="sample"/>

              </login-module>

          </authentication>

                </security-domain>
```

1.3. Configuração do Login Module nas aplicações

Para vincular uma aplicação ao domínio de segurança, é necessário criar e configurar o arquivo /WEB-INF/jboss-web.xml com o nome do security-domain, definido no exemplo acima. Segue abaixo um exemplo do arquivo jboss-web.xml:

```
        <jboss-web xmlns="http://www.jboss.com/xml/ns/javaee"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.jboss.com/xml/ns/javaee http://www.jboss.org/j2ee/
schema/jboss-web_6_0.xsd"
           version="6.0">

           <security-domain>demoiselle-loginmodule-domain</security-domain>
```

```
</jboss-web>
```

A aplicação utiliza as classes do componente *Demoiselle Login Module* em tempo de execução. Dessa forma, é necessário criar e configurar o arquivo /WEB-INF/jboss-deployment-structure.xml, conforme o exemplo abaixo:

```
        <jboss-deployment-structure>
    <deployment>
        <dependencies>
            <module name="br.gov.frameworkdemoiselle.loginmodule.provider" slot="main"/>
        </dependencies>
    </deployment>
</jboss-deployment-structure>
```

O componente *Demoiselle Login Module* criou um Managed Bean, chamado LoginMB, responsável pela definição das ações de login (autenticação e autorização) e logout. A classe LoginMB elimina a necessidade do desenvolvimento de um Managed Bean de login e logout para cada aplicação.

A aplicação deve declarar a classe LoginMB no arquivo /WEB-INF/faces-config.xml, conforme o exemplo abaixo:

```
        <?xml version="1.0" encoding="UTF-8"?>
<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    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"
    version="2.0">
        ...
        <managed-bean>
            <managed-bean-name>login</managed-bean-name>
             <managed-bean-class>br.gov.frameworkdemoiselle.loginmodule.LoginMB</managed-bean-
class>
            <managed-bean-scope>session</managed-bean-scope>
            <managed-property>
                <property-name>successPageRedirect</property-name>
                <value>default</value>
            </managed-property>
            <managed-property>
                <property-name>errorPageRedirect</property-name>
                <value>error</value>
            </managed-property>
        </managed-bean>
        <navigation-rule>
            <navigation-case>
```

```
                <from-outcome>default</from-outcome>
                <to-view-id>/pages/default.xhtml</to-view-id>
                <redirect/>
            </navigation-case>
        </navigation-rule>

        <navigation-rule>
            <navigation-case>
                <from-outcome>error</from-outcome>
                <to-view-id>/error.jsf</to-view-id>
            </navigation-case>
        </navigation-rule>
        ...

</faces-config>
```

É necessária a criação de uma página de login, conforme o exemplo abaixo:

```
        <!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" lang="pt-BR"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
           <link   href="#{facesContext.externalContext.requestContextPath}/resources/styles/
default.css" rel="stylesheet" type="text/css"/>
    <title>Tirando Onda JEE - Exemplo Web</title>
</head>

<body>

    <h:form id="frmLogin">

        <h:panelGrid id="pnlGrid" columns="1">

            <h:panelGroup>
                <h:outputLabel for="inputLogin" value="Username:"/>
                <h:inputText id="inputLogin" value="#{login.username}"/><br/>
            </h:panelGroup>

            <h:panelGroup>
                <h:outputLabel for="inputPassword" value="Password:"/>
                <h:inputSecret id="inputPassword" value="#{login.password}"/><br/>
            </h:panelGroup>

            <h:panelGroup>
                <h:inputSecret id="inputNewPassword" value="#{login.newPassword}"/><br/>
            </h:panelGroup>

      <h:commandButton id="submit" value="Login" action="#{login.autenticateUsernamePassword}"/
><br/>

            <a href="https://localhost:8443/sample-web/login_cert.jsf">
                <img src="resources/images/certificado.gif"></img>
```

```
            </a>

        </h:panelGrid>

    </h:form>

</body>

</html>
```

O exemplo acima, que será chamado de login.xhtml, utiliza propriedades e um método do Managed Bean LoginMB.

O managed-bean-name "login", definido no arquivo /WEB-INF/faces-config.xml, identifica o Managed Bean LoginMB no exemplo acima.

As propriedades utilizadas são as seguintes:

- #{login.username}: identificação do usuário. Ex: CPF.
- #{login.password}: senha do usuário.
- #{login.newPassword}: nova senha do usuário.

O método utilizado é o #{login.autenticateUsernamePassword}, que realiza a autenticação por usuário e senha, e com recurso de troca de senha.

Para autenticação por certificado digital, é necessário clicar na imagem identificada pela tag img do HTML do exemplo acima. O clique na imagem faz o servidor JBoss 6 EAP direcionar a aplicação, chamada de sample-web, para a porta 8443, configurada no JBoss 6 EAP para solicitar o certificado digital do usuário.

Após ler o certificado digital do usuário, o servidor JBoss 6 EAP direciona a aplicação para a página login_cert.jsf, que deve ser criada e cujo conteúdo é mostrado abaixo:

```
        <!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" lang="pt-BR"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
</head>

<f:metadata>
    <f:event type="preRenderView" listener="#{login.onPageLoad}"/>
</f:metadata>

</html>
```

O exemplo acima, que será chamado de login_cert.xhtml, utiliza um método do Managed Bean LoginMB.

O managed-bean-name "login", definido no arquivo /WEB-INF/faces-config.xml, identifica o Managed Bean LoginMB no exemplo acima.

O método utilizado é o #{login.onPageLoad}, que internamente chama o método responsável por realizar a autenticação por certificado digital.

Importante

Independentemente do tipo de autenticação, o usuário autenticado com sucesso vai acessar a página da aplicação identificada pelo property-name "successPageRedirect", definido no arquivo /WEB-INF/faces-config.xml.

Segue abaixo um exemplo da página inicial da aplicação, a index.jsp:

```
        <%@ page contentType="text/html; charset=UTF-8" %>

<%response.sendRedirect("./pages/default.jsf");%>
```

No exemplo acima, a aplicação é redirecionada para a página default.jsf, que é a primeira página da aplicação a ser acessada por usuários autenticados. Caso o usuário ainda não tenha se autenticado, o browser lança um erro 403. Nesse caso, deve ser criada uma página que trate esse erro, conforme o exemplo abaixo:

```
        <!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" lang="pt-BR">

<head>
    <title>403</title>

    <meta http-equiv="content-type" content="text/html; charset=utf-8" />

    <meta http-equiv="refresh" content="0; #{facesContext.externalContext.requestContextPath}/
login.jsf" />

</head>

</html>
```

No exemplo acima, que será chamado de 403.xhtml, a aplicação é redirecionada para a página login.jsf, cujo conteúdo foi mostrado anteriormente neste documento. A idéia é que os usuários se autentiquem antes de acessarem a página default.jsf.

Segue um exemplo do arquivo /WEB-INF/web.xml:

```
        <?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_3_0.xsd"
    version="3.0">
```

```
    <display-name>sample-web</display-name>

    <context-param>
        <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
        <param-value>server</param-value>
    </context-param>

    <context-param>
        <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
        <param-value>.xhtml</param-value>
    </context-param>

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <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>

    <error-page>
        <error-code>403</error-code>
        <location>/403.jsf</location>
    </error-page>

    <error-page>
        <exception-type>java.lang.Throwable</exception-type>
        <location>/error.jsf</location>
    </error-page>

    <security-constraint>
        <web-resource-collection>
            <web-resource-name>private</web-resource-name>
            <description>private resources</description>
            <url-pattern>/pages/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>*</role-name>
        </auth-constraint>
    </security-constraint>

    <security-role>
        <role-name>*</role-name>
    </security-role>
</web-app>
```

No exemplo acima, foi definida uma restrição de segurança (security-constraint) para a aplicação. De acordo com essa restrição, as páginas protegidas (páginas que podem ser acessadas somente por usuários autenticados) são as páginas localizadas dentro da pasta "pages".

Neste documento, a página que exibe mensagens de erro da aplicação, inclusive erros de autenticação, é a página error.jsf. Esta página é identificada pelo property-name "errorPageRedirect", definido no arquivo /WEB-INF/faces-config.xml. Um exemplo desta página, que deve ser criada, é mostrado abaixo:

```
        <!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" lang="pt-BR"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core">

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>Pagina de erro</title>
</head>

<body>
    The error message is: #{requestScope['javax.servlet.error.message']}

    <h:messages globalOnly="true"/>

    <br/><br/>

    <a href="http://localhost:8080/sample-web">
        Nova tentativa de login
    </a>
</body>

</html>
```

Finalmente, o servidor JBoss 6 EAP pode ser iniciado e a aplicação sample-web pode ser chamada através do endereço http://localhost:8080/sample-web.

1.4. Implementação de provedores de segurança

O *Demoiselle Login Module* define duas interfaces: IAuthenticationProvider e IAuthorizationProvider.

A interface IAuthenticationProvider é uma abstração para a operação de identificação do usuário. O exemplo abaixo mostra a implementação da interface IAuthenticationProvider:

```
        public class SampleAuthenticationProvider implements IAuthenticationProvider
        {

            public void initialize(Properties props)
            {
                    ...
            }

            public Principal authenticate(String user, String password) throws LoginException
            {
                    ...
            }
```

```
            public Principal authenticate(String user, String password, String newPassword)
                    throws LoginException
            {
                    ...
            }

            public Principal authenticate(X509Certificate x509) throws LoginException
            {
                    ...
            }
        }
```

A interface IAuthorizationProvider é uma abstração para a operação de carregar papéis do usuário e através deles definir se o usuário está ou não autorizado a realizar uma operação. O exemplo abaixo mostra a implementação da interface IAuthorizationProvider:

```
        public class SampleAuthorizationProvider implements IAuthorizationProvider
        {

            public void initialize(Properties props) {
                    ...
            }

           public Collection<Role> authorize(Principal callerPrincipal) throws LoginException {
                    ...
            }
        }
```

1.5. Anexo 1: Configuração do JBoss 6 EAP para uso de certificado digital

Para viabilizar o uso de certificado digital, é necessário configurar uma conexão segura (SSL) entre a máquina do usuário e o servidor de aplicações JBoss 6 EAP. Para isso, deve ser adicionado um conector HTTPS no arquivo $JBoss/standalone/ configuration/standalone.xml no subsistema urn:jboss:domain:web, conforme o exemplo abaixo:

```
               <subsystem  xmlns="urn:jboss:domain:web:1.1"  default-virtual-server="default-
host" native="false">

            <connector name="http" protocol="HTTP/1.1" scheme="http" socket-binding="http"/>

                        <connector  name="https"  protocol="HTTP/1.1"  scheme="https"  socket-
binding="https" secure="true">

                  <ssl key-alias="tibco" password="123456" certificate-key-file=".../keystore-
tibco.jks"  cipher-suite="ALL"  protocol="TLS"  verify-client="true"  ca-certificate-file=".../
truststore.jks"/>
```

```
            </connector>

            <virtual-server name="default-host" enable-welcome-root="true">

                <alias name="localhost"/>

                <alias name="example.com"/>
            </virtual-server>

        </subsystem>
```

Em relação ao exemplo acima, seguem alguns conceitos:

- Atributo certificate-key-file: representa o certificado do servidor JBoss 6 EAP.
- Atributo password: representa a senha do certificado do servidor JBoss 6 EAP.
- Atributo ca-certificate-file: representa o repositório de certificados de todas as Autoridades Certificadoras reconhecidas pelo servidor JBoss 6 EAP.
- Atributo verify-client: define se a conexão segura vai solicitar o certificado digital do usuário.

Na definição do domínio de segurança utilizado pela aplicação, devem ser declaradas informações sobre a conexão segura, conforme o exemplo abaixo. Um domínio de segurança é configurado em $JBoss/standalone/configuration/standalone.xml, no subsistema urn:jboss:domain:security:

```
        <security-domain name="demoiselle-loginmodule-domain" cache-type="default">

<authentication>

    <login-module code="br.gov.frameworkdemoiselle.loginmodule.provider.ProviderLoginModule"
                  flag="sufficient" module="br.gov.frameworkdemoiselle.loginmodule.provider">

       ...

    </login-module>

</authentication>

<jsse    keystore-password="123456"    keystore-url="file:.../keystore-tibco.jks"    truststore-
password="changeit" truststore-url="file:.../truststore.jks"/>

                </security-domain>
```

Em relação ao exemplo acima, seguem alguns conceitos:

- Atributo keystore-url: representa o certificado do servidor JBoss 6 EAP.
- Atributo truststore-url: representa o repositório de certificados de todas as Autoridades Certificadoras reconhecidas pelo servidor JBoss 6 EAP.
- Atributo keystore-password: representa a senha do certificado do servidor JBoss 6 EAP.

- Atributo truststore-password: representa a senha do repositório de certificados de todas as Autoridades Certificadoras reconhecidas pelo servidor JBoss 6 EAP.

1.6. Anexo 2: Adição de certificados no browser do usuário

É necessário importar as cadeias de Autoridades Certificadoras que devem ser reconhecidas pelo browser. Considerando que o browser do usuário seja o Firefox, deve-se clicar em Editar → Preferências → Opção Avançado → Aba Criptografia → Clicar no botão Certificados → Aba Autoridades → Clicar no botão Importar → Selecionar o arquivo que contém a lista das Autoridades Certificadoras.

Também é necessário carregar o driver do token no browser do usuário. Considerando que o browser do usuário seja o Firefox, deve-se clicar em Editar → Preferências → Opção Avançado → Aba Criptografia → Clicar no botão Dispositivos de Segurança → Carregar → Selecionar o driver do token que contém o certificado do usuário (ex: libepsng_p11.so.1.2.2).