**Autenticação com JSF**

**Maneiras de se fazer**

* **JSF** => Controlar manualmente através de **ManagedBeans** de Sessão e Filtros.
* **JAAS** => **Java Authentication and Authorization Service API** (Padrão Java EE para segurança de sistemas)
* **Frameworks** => Existem frameworks específicos para este trabalho, dentre os principais podemos citar:
* **Spring Security**: <http://projects.spring.io/spring-security/>
* **Apache Shiro**: <http://shiro.apache.org>

Para fins de estudos abordaremos apenas a primeira maneira (JSF), e para implementar o módulo de **login** será necessário seguir os seguintes passos:

Criar método para buscar usuário a partir de **email** e **senha** (EJB)

//simples busca de cliente no banco

public Cliente procurarClientePorEmailSenha(String email, String senha) {

TypedQuery query = em .createQuery( "SELECT c FROM Cliente c WHERE c.email = :email AND c.senha = :senha", Cliente.class);

query.setParameter("email", email);

query.setParameter("senha", convertStringToMd5(senha));

List<Cliente> clientes = query.getResultList();

if (clientes.size() == 1)

return clientes.get(0);

return null;

}

* Conversão da senha para MD5

//método para criptografia da senha

private String convertStringToMd5(String valor) {

MessageDigest mDigest;

try {

// Instanciamos o nosso HASH MD5, poderíamos usar outro como SHA, por exemplo.

mDigest = MessageDigest.getInstance("MD5");

// Convert a String valor para um array de bytes em MD5

byte[] valorMD5 = mDigest.digest(valor.getBytes("UTF-8"));

// Convertemos os bytes para hexadecimal, assim podemos salvar

// no banco para posterior comparação se senhas

StringBuffer sb = new StringBuffer();

for (byte b : valorMD5) {

sb.append(Integer.toHexString((b & 0xFF) | 0x100).substring(1,3));

}

return sb.toString();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

return null;

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

return null;

}

}

* Criar métodos para obter usuário ‘logado’, efetuar **login** e **logout** (ManagedBean).

// verifica o email e senha, se válidos registra o atributo "**usuarioLogado**" na sessão do navegador com o valor do objeto cliente buscado no banco

public void fazerLogin() {

FacesContext facesContext = FacesContext.getCurrentInstance();

if (getUsuarioLogado() == null) {

Cliente cliente = clienteEJB.procurarClientePorEmailSenha(email, senha);

if (cliente != null) {

Map session = facesContext.getExternalContext() .getSessionMap(); session.put("usuarioLogado", cliente);

try {

facesContext.getExternalContext().redirect("index.xhtml");

} catch (IOException e) {

e.printStackTrace();

}

} else {

facesContext.addMessage(null, new FacesMessage( FacesMessage.SEVERITY\_ERROR, "Erro de Login", "Usuario e/ou Senha incorretas"));

}

} }

// remove o atributo "usuarioLogado" registrado na sessão no momento do login

// e invalida a sessão do usuário no navegador

public void fazerLogout() {

Map<String, Object> session = FacesContext.getCurrentInstance()

.getExternalContext().getSessionMap();

session.remove("usuarioLogado");

HttpServletRequest request = (HttpServletRequest) FacesContext

.getCurrentInstance().getExternalContext().getRequest();

// invalida sessão do usuario

request.getSession().invalidate();

try {

FacesContext.getCurrentInstance().getExternalContext()

.redirect("login.xhtml");

} catch (IOException e) {

e.printStackTrace();

}}

// obtem usuário logado

public Cliente getUsuarioLogado() {

Map<String, Object> session = FacesContext.getCurrentInstance()

.getExternalContext().getSessionMap();

return (Cliente) session.get("usuarioLogado");

}

* Criar pagina de login **(.xhtml)**, apontando para o **bean** anterior

<p:fieldset legend="Login">

<p:growl showDetail="true" />

<h:panelGrid columns="2">

<p:outputLabel value="Email:" for="email" />

<p:inputText id="email" value="#{usuarioBean.email}" required="true"/>

<p:outputLabel value="Senha:" for="senha" />

<p:password id="senha" value="#{usuarioBean.senha}" required="true" />

</h:panelGrid>

<p:commandButton icon="ui-icon-user" value="Entrar"

action="#{usuarioBean.fazerLogin}" update="@form :messages" />

</p:fieldset>

* Definir filtro para interceptar requisições e evitar que o usuário acesse as páginas sem está “**logado**”.

@Override

public void destroy() {

// sem necessidade de implementação

}

@Override

public void destroy() {

// sem necessidade de implementação

}

@Override

public void doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) throws IOException, ServletException {

HttpServletRequest httpRequest = (HttpServletRequest) request;

Cliente cliente = (Cliente) httpRequest.getSession().getAttribute(

"usuarioLogado");

String paginaAcessada = httpRequest.getRequestURI();

boolean requestDoLogin = paginaAcessada.contains("login.xhtml");

if (cliente != null) {

if (requestDoLogin) {

HttpServletResponse httpResponse = (HttpServletResponse) response;

httpResponse.sendRedirect("index.xhtml");

} else {

chain.doFilter(request, response);

}

} else {

if (!requestDoLogin

&& !paginaAcessada.contains("javax.faces.resource")) {

httpRequest.getRequestDispatcher("/login.xhtml").forward(

request, response);

} else {

chain.doFilter(request, response);

}

}

}

**Configurar tempo de inatividade**

Podemos configurar o tempo de inatividade no **web.xml**. Após esse tempo configurado, a sessão automaticamente será invalidada e o usuário perderá seu login. Para isso basta adicionar o trecho de código abaixo no arquivo web.xml.

<session-config> <session-timeout>5</session-timeout> </session-config>

**Obs.:** O valor do parâmetro é dado em minutos. No exemplo apresentado, o usuário perderá a sessão automaticamente se ficar 5 minutos sem executar qualquer requisição a aplicação.

**Referências**

**Livro de apoio**: <http://alga.works/livro-javaee/>

**Criando um módulo de login:** <http://www.devmedia.com.br/jsf-session-criando-um-modulo-de-login/30975>

**Login utilizando JAAS**: <http://www.javacodegeeks.com/2012/06/full-webapplication-jsf-ejb-jpa-jaas.html>

**JAAS e JSF:** <http://uaihebert.com/user-login-validation-with-jaas-and-jsf/>

**Configurações de segurança com o JSF**: <https://blogs.oracle.com/enterprisetechtips/entry/improving_jsf_security_configuration_with>

**Segurança com JSF + JavaEE + JBoss:** <http://www.javacodegeeks.com/2014/01/securing-a-jsf-application-with-java-ee-security-and-jboss-as-7-x.html>