Trabalhando com Javabeans

Nossa classe usuário será agora convertida em um JavaBean.

```
package estacio;

import java.io.Serializable;

public class Usuario implements Serializable{

	private String nome;
	private String login;
	private String senha;
	private String telefone;

	public Usuario() {

	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getLogin() {
		return login;
	}

	public void setLogin(String login) {
		this.login = login;
	}

	public String getSenha() {
		return senha;
	}

	public void setSenha(String senha) {
		this.senha = senha;
	}

	public String getTelefone() {
		return telefone;
	}

	public void setTelefone(String telefone) {
		this.telefone = telefone;
	}
}
```

Escopos dos JavaBeans

O escopo de um JavaBean diz respeito ao tempo em que esse permanecerá na memória depois de sido instanciado. Existem quatro possíveis escopos para um JavaBean: page, request, session e application.

Escopo page
Esse é o escopo padrão para os beans. O objeto é salvo no pageContext e dura o tempo em que a página é processada, isso é o bean; estará acessível somente na página onde foi instanciado.

Escopo request
Nesse escopo, o objeto é armazenado no ServletRequest. Seu funcionamento é semelhante ao escopo de página, só que outras páginas, que forem incluídas na página que inseriu o objeto, também poderão ter acesso ao JavaBean criado. Caso a requisição seja encaminhada os JavaBeans de escopo request continuam acessíveis nas que tratem essas requisições.

Escopo session
JavaBeans criados no escopo de sessão são armazenados no objeto HttpSession, estando accessíveis a todas as páginas residentes na sessão do usuário. Uma vez que a sessão expira ou é invalidada, esses objetos são descartados.

Escopo application
JavaBean, declarado com esse escopo, é armazenado no objeto ServletContext ,sendo compartilhado por todas as páginas existente no contexto. Esse objeto só é destruído, quando o servidor (contêiner) é parado ou reinicializado.

Utilizando JavaBeans
Com o intuito de demonstrar como utilizar os nossos JavaBeans, iramos criar uma classe chamada Util, onde iremos inserir o método md5.

Util.java

```
package estacio;

import java.io.Serializable;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Util implements Serializable{

        public Util() {
        }

        public String md5(String senha){
      String sen = "";
      MessageDigest md = null;
      try {
         md = MessageDigest.getInstance("MD5");
      } catch (NoSuchAlgorithmException e) {
         e.printStackTrace();
      }
      BigInteger hash = new BigInteger(1, md.digest(senha.getBytes()));
      sen = hash.toString(16);
      return sen;
   }
}
```

Agora iremos alterar nossas páginas logar.jsp e cadastrar.jsp para que essas utilizem o JavaBean Util

cadastrar.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>

<%@page import="estacio.Usuario" %>
<%@page import="java.util.*" %>

<!-- Indicamos que iremos usar o bean -->
<jsp:useBean id="u" class="estacio.Util" scope="page" />
<jsp:useBean id="usuario" class="estacio.Usuario" scope="request" />



<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Cadastro de usuário</title>
</head>
<body>
<%

try{
//Aqui passamos a utilizar os métodos geters and serters no lugar de ter acesso direto
aos atributos
usuario.setNome(request.getParameter("nome"));
usuario.setLogin(request.getParameter("login"));
usuario.setSenha(request.getParameter("senha"));
usuario.setTelefone(request.getParameter("telefone"));

//Exibe no console a senha recebida
System.out.println("Senha recebida:" + usuario.getSenha() );

//Substituo a senha sem conversão pela senha já convertida
usuario.setSenha(u.md5(usuario.getSenha()));

System.out.println("Senha em MD5:" + usuario.getSenha());

//Utilizarei o objeto application para armazenar os usuários
application.getAttribute("usuarios");
HashMap<String, Usuario> usuarios = (HashMap<String, Usuario>)
application.getAttribute("usuarios");
if(usuarios==null){
        usuarios = new HashMap<String, Usuario>();
}
usuarios.put(usuario.getLogin() , usuario);
application.setAttribute("usuarios", usuarios);
out.println("<h3>Cadastro realizado com sucesso!.</h3>");
}catch (Exception e){
        out.println("<h3>Erro inesperado.</h3>");

}
%>
<a href="index.jsp">Clique aqui para se logar</a>
</body>
```

```
</html>
```

logar.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<%@page import="estacio.Usuario" %>
<%@page import="java.util.*" %>

<!-- Indicamos que iremos usar o bean -->
<jsp:useBean id="u" class="estacio.Util" scope="page" />
<jsp:useBean id="usuario" class="estacio.Usuario" scope="request" />

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login</title>
</head>
<body>
<%
//Pega os usuários adicionados ao objeto application
HashMap<String, Usuario> usuarios = (HashMap<String, Usuario>)
application.getAttribute("usuarios");

if(usuarios!=null){

//pega os parâmetros envias pelo formulários
String login = request.getParameter("login");
String senha = request.getParameter("senha");

//valida o usuário e senha

usuario = usuarios.get(login);

if(login!=null && senha !=null && usuarios.get(login)!=null &&
usuarios.get(login).getSenha()!=null &&
usuarios.get(login).getSenha().equals(u.md5(senha))){
        session.setAttribute("login", login);
        response.sendRedirect("principal.jsp");

}else{
        %>
        <h2 style="color: red">Usuário ou senha inválido</h2>
        <a href="index.jsp">Clique aqui para voltar</a>
        <%
}
}else{
        %>
        <h2 style="color: red">Não existem usuários cadastrados</h2>
        <a href="formulario.jsp">Clique aqui para se cadastrar</a>
        <%
}
%>
</body>
</html>
```

Ação jsp:setPropety
A ação setProperty pode ser utilizado para inserir valores passados nas requisições, automaticamente, para os JavaBeans criados.

```
<jsp:useBean id="usuario" class="estacio.Usuario" scope="request" />
<jsp:setProperty name="usuario" property="login"/>
```

Nesse exemplo acima, a propriedade nome indica o id no bean a ser referenciado; a propriedade property indica em qual atributo do bean iremos setar uma valor, caso a requisição possua também uma atributo com o mesmo nome que property, esse valor recebido será passado automaticamente para o bean. No exemplo acima, na página index.jsp, existe uma imputText com nome login, sendo assim o container pegará esse valor e irá inseri-lo na instancia do JavaBean Usuario instanciado através do jsp:useBean.

Além da forma acima exemplificada, indicar um valor especifico como a seguir:
`<jsp:setProperty name="usuario" property="nome" value="João Paulo"/>`
Nesse caso, após criado o bean esse tria setado a seu atributo nome a String *"João Paulo"*

Uma última forma possível seria usar o valor '*' no atributo property, dessa forma todo valor passado na requisição que tenha no bean um atributo com o mesmo nome; esses receberão automaticamente os valores daqueles.

Abaixo, alteraremos as páginas logar.jsp e cadastrar.jsp, para que essas utilizem a ação setProperty.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>

<%@page import="estacio.Usuario" %>
<%@page import="java.util.*" %>

<!-- Indicamos que iremos usar o bean -->
<jsp:useBean id="u" class="estacio.Util" scope="page" />
<jsp:useBean id="usuario" class="estacio.Usuario" scope="request" />

<jsp:setProperty name="usuario" property="*"/>


<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Cadastro de usuário</title>
</head>
<body>
<%

try{

//Exibe no console a senha recebida
System.out.println("Senha recebida:" + usuario.getSenha() );

//Substituo a senha sem conversão pela senha já convertida
```

```
usuario.setSenha(u.md5(usuario.getSenha()));

System.out.println("Senha em MD5:" + usuario.getSenha());

//Utilizarei o objeto application para armazenar os usuários
application.getAttribute("usuarios");
HashMap<String, Usuario> usuarios = (HashMap<String, Usuario>)
application.getAttribute("usuarios");
if(usuarios==null){
        usuarios = new HashMap<String, Usuario>();
}
usuarios.put(usuario.getLogin() , usuario);
application.setAttribute("usuarios", usuarios);
out.println("<h3>Cadastro realizado com sucesso!.</h3>");
}catch (Exception e){
        out.println("<h3>Erro inesperado.</h3>");

}
%>
<a href="index.jsp">Clique aqui para se logar</a>
</body>
</html>
```

Ação jsp:include

Inclui o recurso em tempo de requisição, difere da diretiva include, pois a primeira insere o conteúdo para posteriormente criar o Servlet equivalente, enquanto na ação include a inserção da página ocorre em tempo de execução.

```
<jsp:include page="relative URL" flush="true" />
```

Ação jsp:forward

Permite passar a requisição para outro recurso.

```
<jsp:forward page="relativeURL" />
```