

Lab 6 - Manipulando Sessões com Servlet e JSP.

Neste laboratório iremos continuar o Projeto **Banco-WebView**, realizaremos interações entre sessões e JSP.

Exercícios

Exercício 1: Refatorar as telas JSP, utilizando mais recursos do próprio JSP.

Exercício 2: Criar um fluxo de gerenciamento de logs na sessão.

Exercício 3: Refatorar o login.

Exercício 1 - Refatorando as telas JSP.

1. Crie uma nova pasta dentro de *WebContent* com o nome **templates**, dentro do diretório **templates** crie um arquivo **JSP** com o nome **cabeçalho.jsp**.

2. Adicione o código abaixo dentro do arquivo **cabeçalho.jsp**.

```
<%@page import="threeway.projeto.modelo.Log"%>
<%@page import="java.util.List"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.Date" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<div class="navbar navbar-default navbar-static-top" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse"
                data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span> <span
                    class="icon-bar"></span> <span class="icon-bar"></span> <span
                    class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="#">Projeto Banco 3Way NetWorks!</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav navbar-right">
                <li>
                    <a>
                        <form action="Logout" id="formIcons">
                            <span>Bem vindo, sr(a).</span>
                            <b>
                                <script type="text/javascript">
                                    document.write(getCookie());
                                </script>
                            </b>
                            <input type="submit" name="action" class="btn btn-xs btn-danger"
                                value="sair">
                            <button type="button" name="action"
                                class="btn btn-xs btn-default" value="info" data-toggle="modal"
                                data-target="#modalInfo">
                                
                            </button>
                        </form>
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>

<div class="modal fade" id="modalInfo" tabindex="-1" role="dialog" aria-hidden="true">
<div class="modal-dialog">
<div class="modal-content">
```

```
<div class="modal-header">
  <button type="button" class="close" data-dismiss="modal" aria-hidden="true">x</button>
  <h4 class="modal-title" id="myModalLabel">Gerenciamento de log</h4>
</div>
<div class="modal-body">
  <div class="table-responsive">
    <table class="table table-bordered table-hover">
      <thead>
        <tr>
          <th>Action</th>
          <th>Data</th>
        </tr>
      </thead>
      <tbody>
        <%
          List<Log> dados = (ArrayList<Log>) request.getSession().getAttribute("dadosLog");

          for (Log log : dados){

            <tr>
              <td><% out.print(log.getDescricao()); %></td>
              <td><% out.print(log.getDataFormatada()); %></td>
            </tr>
          <%>
        </tbody>
      </table>
    </div>
  </div>
  <div class="modal-footer">
    <button type="button" class="btn btn-default" data-dismiss="modal">Cancelar</button>
  </div>
</div>
</div>
```

Por enquanto vamos utilizar um laço **for** para percorrer a lista de logs e de acordo com cada item da lista, será criada uma nova `<tr>` com os dados do log. Repare que a lista de dados está sendo recuperada da sessão através do atributo `"dadosLog"`. Nos próximos tópicos criaremos os métodos para popular o objeto `"dadosLog"`.

2. Altere os arquivos **index.jsp**, **manterCliente.jsp** e **operacoesBancarias.jsp** removendo o código fonte que dentro do bloco abaixo.

```
<!-- Static navbar -->
...
<!-- /Static navbar -->
```

3. Após remover dos dados dentro do bloco referenciado acima no tópico 2, adicione o `include` abaixo no lugar do código removido, o arquivo contém o mesmo código removido e um popup, o diferencial é que esse código não será mais replicado nas demais páginas, agora teremos um "reaproveitamento desse código" usando o `include` do JSP.

```
<jsp:include page="/templates/cabecalho.jsp"/>
```

4. Crie um arquivo com o nome **menu.jsp** dentro do diretório **templates** e adicione o código abaixo.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<div class="panel-body" >
  <ul class="ds-btn">
    <li>
      <a class="btn btn-lg btn-primary" href="ManterCliente">
```

```

        <i class="glyphicon glyphicon-user pull-left"> </i>
        <fmt:message key="titulo.aba.cliente"/>
    </a>

</li>

<c:if test="${cliente != null}">
<li>
    <a class="btn btn-lg btn-danger" href="OperacoesBancarias">
        <i class="glyphicon glyphicon-tasks pull-left"> </i>
        <span>

            <fmt:message key="titulo.aba.operacoesBancarias" />

        </span>
    </a>
</li>
</c:if>
</ul>
</div>

```

5. Remova o código abaixo das paginas **index.jsp**, **manterCliente.jsp** e **operacoesBancarias.jsp** e adicione o **include** da pagina **menu.jsp** criada no passo anterior.

(Antes)

```

<!-- Panel referente ao menu -->
<div class="panel-body">
    <button type="button" class="btn btn-default"
        onclick="window.location='./manterCliente.jsp'">
        Manter Clientes
    </button>
    <button type="button" class="btn btn-default"
        onclick="window.location='./operacoesBancarias.jsp'">
        Operações Bancarias
    </button>
</div>

<!-- Fim Panel menu -->

```

(Depois)

```

<!-- Panel referente ao menu -->
<jsp:include page="/templates/menu.jsp"/>
<!-- Fim panel menu -->

```

6. Refatore o arquivo **login.jsp**, adicionando o código no abaixo da tag **<div class="container">**, código com o objetivo de recuperar e mostrar para o usuário o objeto `{msg}` vinculado a sessão caso seja diferente de nulo, o sistema exibira uma mensagem ao usuário.

```

<c:if test="${msg != null}">
    <div class="alert alert-danger fade in" id="btnIndividual">
        <button type="button" class="close" data-dismiss="alert" aria-hidden="true">x</button>
        <strong>${msg}</strong>
    </div>
</c:if>

```

Agora adicione a seguinte tagLib no topo da página:

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

Essa tagLib é referente a biblioteca de tags JSTL (tags de internacionalização, formatação, xml e outras) que será explicada nos próximos laboratórios.

Exercício 2 - Criando o fluxo de gerenciamento de logs na sessão.

1. Vamos ajustar o servlet Login para realizar a autenticação correta e também implementaremos a funcionalidade de gerar log das ações realizadas no sistema, primeiro crie uma classe JAVA com o nome **AppController** dentro do pacote servlets.

```
package threeway.projeto.servlets;
```

```
public class AppController extends HttpServlet
```

2. Note que essa classe JAVA herda de HttpServlet. Agora altere todos os servlets da aplicação mudando a classe pai (**extends**).

(Antes)

```
public class Login extends HttpServlet
```

(Depois)

```
public class Login extends AppController
```

A classe **AppController** terá ações gerais que servirão para todos os servlets que herdarem de **AppController** e também terá todos os métodos necessários para um **Servlet** pois **AppController** estende de **HttpServlet**.

3. Crie uma classe JAVA como nome **Log** dentro do pacote **threeway.projeto.modelo** do projeto **Banco-Modelo**.

```
public class Log {
```

```
private Long id;
```

```
private String descricao;
```

```
private Date data;
```

```
//Crie os getters e setters
```

```
}
```

4. Dentro da classe **AppController** adicione os seguintes métodos e atributos, lembre de gerar os métodos getters e setters dos atributos.

```
//Criando um atributo referente a interface HttpServletRequest
```

```
private HttpServletRequest request;
```

```
//Criando um atributo referente a interface HttpServletResponse
```

```
private HttpServletResponse response;
```

```
//Criando um atributo referente a classe de modelo Log
```

```
private Log log;
```

```
@Override
```

```
protected void service(HttpServletRequest req, HttpServletResponse resp)  
throws ServletException, IOException {
```

```
    //atribui o objeto request ao atributo request que criamos acima  
    this.request = request;
```

```
    //atribui o objeto response ao atributo response que criamos acima  
    this.response = response;
```

```
    //chama metodo service da classe pai  
    super.service(request, response);
```

```
}
```

5. Agora vamos criar o método de gerar log dentro de **AppController**.

```
public void gerarLog(final String descricao){
```

```
//obtem a lista de logs já existentes
List<Log> logsAnteriores = (List<Log>) this.getRequest().getSession().getAttribute("dadosLog");

// verifica se essa lista de logs está vazia, se estiver ela é e inicializada com
// a instância de um ArrayList
if (logsAnteriores == null || logsAnteriores.isEmpty()){

    logsAnteriores = new ArrayList<Log>();

}

//Monta a descrição do log
this.getLog().setDescricao(descricao);

//Monta a data que o log foi gerado
this.getLog().setData(new Date());

//Adiciona o novo log a lista de logs existentes
logsAnteriores.add(this.getLog());

//coloca a lista de logs anteriores como o novo log gerado na sessão novamente.
this.getRequest().getSession().setAttribute("dadosLog", logsAnteriores);
}
```

6. Ainda na classe **AppController** altere o método **getter** do atributo **log** para que fique com a forma abaixo.

```
public Log getLog() {
    //verifica se o objeto log está nulo, caso sim cria uma instancia do objeto Log
    if (log == null){
        log = new Log();
    }
    return log;
}
```

Caso ainda tenha dúvidas sobre **servlet** e os métodos da **interface HttpServlet**, aproveite o tempo para extrair esse conhecimento do seu instrutor, nesse lab usaremos um dos métodos que é o **service** para obter o objeto **request** nas requisições.

Exercício 3: Refatorar o login.

1. Altere o método **doGet()** do **servlet Login** para que o mesmo fique conforme abaixo.

```
// recupera do form através do request o valor do input com o name login
final String login = request.getParameter("login");

// recupera do form através do request o valor do input com o name senha
final String senha = request.getParameter("senha");

//Verifica se login e senha conferem, utilizaremos o login: admin e senha: admin como valores fixos.
if ( "admin".equals(login) && "admin".equals(senha) ){

    final String cookieName = "usuario";

    final String cookieValue = login;

    //criando o objeto cookie
    Cookie cookie = new Cookie(cookieName, cookieValue);

    //duração de um dia
    cookie.setMaxAge(60*60*24);

    //adicionando o cookie ao navegador
    response.addCookie(cookie);

    this.gerarLog("Realizou o Login");

    //limpando o campo mensagem da sessão
    this.getRequest().getSession().setAttribute("msg", null);

    //adicionando o login do usuario na sessão
    this.getRequest().getSession().setAttribute("usuario", login);
}
```

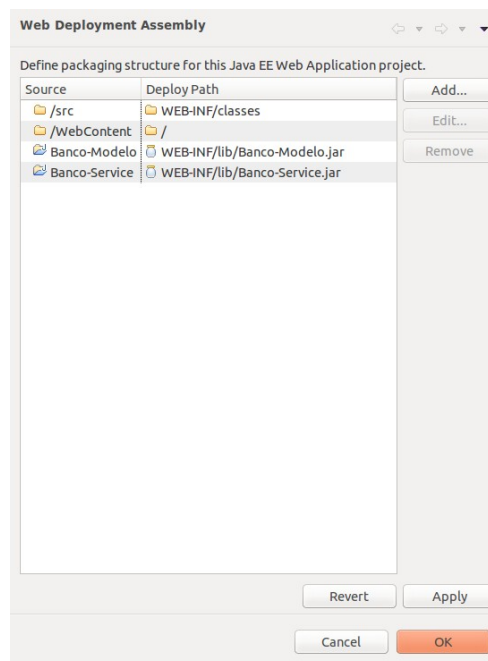
```
//adicionando os campos agencia e banco na sessão
this.getRequest().getSession().setAttribute("agencia", AgenciaService.agenciaSistema());

//esta funcionalidade será ensinada nas próximas aulas
response.sendRedirect("index.jsp");


} eles {
    //caso digite o login e senha invalido
    this.getRequest().getSession().setAttribute("msg", "Dados do login inválido");

    //Esta funcionalidade será ensinada nas proximas aulas
    response.sendRedirect("login.jsp");
}
```

2. Como estamos trabalhando uma arquitetura de projeto diferenciada e adicionamos mais uma classe no projeto Banco-Modelo temos que gerar o **jar** dos projetos *modelo* e *servico*, siga os passos.
 1. Clique com o botão direito no projeto **Banco-WebView** > **Properties** > **Deployment Assembly** > **Add...** > **Project** > **selecione os projeto Banco-Modelo e Banco-Servico**.
 2. Isso fará com que toda vez que uma build do projeto *Banco-WebView* for gerada, serão gerados os *.jars* de *modelo* e *servico* em *webview*.



3. Reinicie o servidor, acesse no navegador a URL <http://localhost:8080/Banco-WebView/>.

3. Após realizar o login com os dados (**usuário:admin senha:admin**), clique na imagem  e verifique que o primeiro log na sessão foi gerado.

