Unidade 12: Uso de Sessão e DAO e Servlets

Servlets de Comportamento Variável

Prof. Daniel Caetano

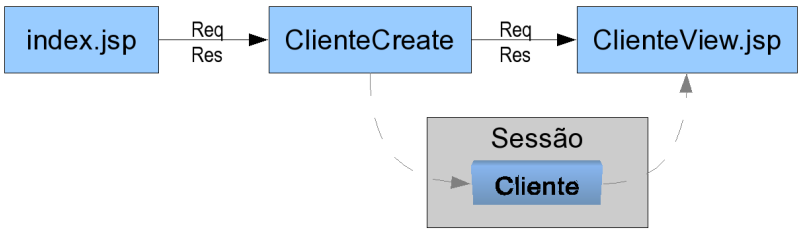
Objetivo: Implementar servlets de comportamento variável de acordo com o estado de atributos de requisição.

Bibliografia: QIAN, 2007; DEITEL, 2005.

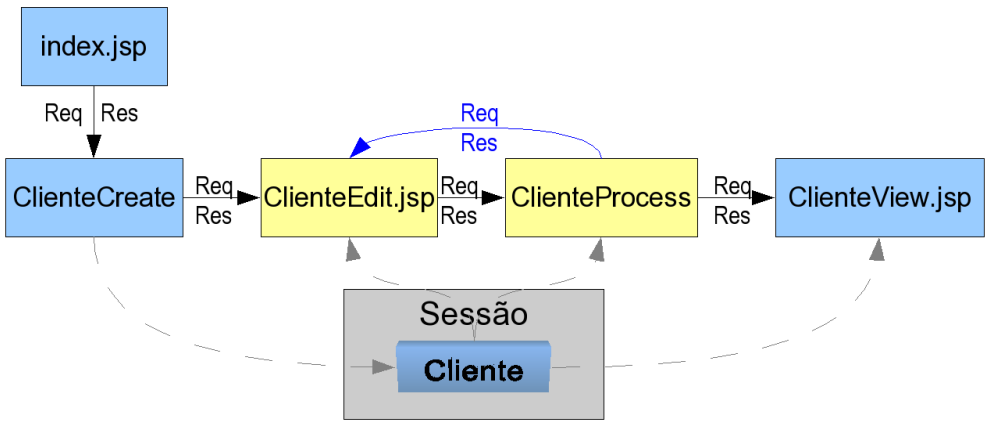
INTRODUÇÃO

Nas aulas anteriores estudamos todos os elementos para construir uma Aplicação Web elaborada de forma desconexa, isto é, vimos cada uma das partes de maneira isolada.

Nesta aula iremos construir um sistema básico muito simples, como o indicado abaixo, usando alguns dos elementos que já construímos anteriormente (em especial, o Cliente e o ClienteDAO):



E, mais ao fim da aula, modificaremos o mesom para um novo estágio, representado a seguir, permitindo que o cliente criado pelo **ClienteCreate** possa ser editado pela dupla de edição (a JSP de apresentação **ClienteEdit.jsp** e o servlet de processamento **ClienteProcess**) antes de ser exibido na tela.



1. CRIANDO O APLICATIVO WEB BÁSICO

O Servlet base é o servlet inicial de nossa aplicação, normalmente fornecendo um menu para acesso às principais funções do aplicativo. Para criá-lo, façamos o seguinte:

PASSO 1. Clique em **Criar Projeto** e selecione "**Java Web**" e "**Aplicação Web**" e clique em **Próximo.** Dê o nome ao projeto de **WProjeto9** e clique em **Próximo**, verifique se o servidor de aplicações selecionado é o **GlassFish** e clique em **Finalizar**.

PASSO 2. Primeiramente, vamos "migrar" as informações úteis de projetos anteriores. Abra o projeto **WProjeto6**, da aula 10, clique com o botão direito no pacote "**entidades**" e selecione a opção "**copiar**". Clique no "**Pacotes de Código Fonte**" do WProjeto9 que acabamos de criar e selecione "**colar**". Feche o WProjeto6. Consideramos aqui que o banco de dados necessário já está criado, por conta do antigo projeto!

PASSO 3. Vamos agora ajustar o index.jsp para apontar para nosso servlet inicial que, pelo projeto, irá se chamar ClienteCreate:

index.jsp

```
<%--
    Document   : index
    Created on : 04/03/2011, 10:13:15
    Author     : djcaetano
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Acessa o Sistema</title>
    </head>
    <body>
        <p><a href='ClienteCreate'>Criar Cliente!</a></p>
   </body>
</html>
```

PASSO 4. Agora que a aplicação está confgiruada, vamos criar o nosso servlet base. Clique com o botão direito em "**Pacotes de Código Fonte**" e selecione **Novo > Pacote Java**. Dê o nome de **cadcli** ao pacote, clicando depois em **Finalizar**.

PASSO 5. Clique com botão direito no pacote **cadcli** e selecione **Novo > Servlet**. Dê o nome **ClienteCreate** para o servlet e clique em **Próximo**. Marque a opção "**Adicionar informação ao descritor de implementação (web.xml)**" e clique em **Ok/Finalizar**.

PASSO 6. Já com o arquvo ClienteCreate.java aberto, remova tudo que é desnecessário, considerando que este é um servlet de processamento. A função deste Servlet é

criar um objeto Cliente e inseri-lo na sessão. Observe como isso pode ser feito no código a seguir.

ClienteCreate.java (método processRequest)

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

         try {

             // Cria um Cliente de Teste
             Cliente c = new Cliente();
             c.setNome("Fulano da Silva");
             c.setCpf("12345678901");

             // Requisita Objeto de Sessão
             HttpSession sessao = request.getSession();
             // Insere cliente na sessão
             sessao.setAttribute("cliente", c);

             // Solicita o despachante de requisições
             RequestDispatcher rd;
             rd = request.getRequestDispatcher("/ClienteView.jsp");
             // Encaminha requisição
             rd.forward(request,response);
             return;

         } finally {
         }
     }
```

PASSO 7. Vamos, finalmente, criar o ClienteView.jsp. Clique com o botão direito em "**Páginas Web**" e selecione **Novo > JSP...**. Dê o nome de **ClienteView** ao JSP e modifique o código dele como se segue:

ClienteView.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8" import="entidades.*" %>

<%
      String texto = "Nenhum cliente para visualizar.";
      Cliente c = (Cliente)session.getAttribute("cliente");
      if (c != null) texto = c.toString();
%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente View</title>
    </head>
    <body>

        <%= texto %>

    </body>
</html>
```

Experimente a aplicação! Veja se funciona!

2. ADICIONANDO A EDIÇÃO À APLICAÇÃO

PASSO 8. Primeiramente, vamos criar a apresentação da edição, ou seja, o ClienteEdit.jsp. Para tanto, clique com o botão direito em "**Páginas Web**" e selecione **Novo > JSP...**. Dê o nome de **ClienteEdit** ao JSP e modifique o código dele como se segue:

ClienteEdit.jsp

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

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente Edit</title>
    </head>
    <body>

      <form action="ClienteProcess" method="post">
            <p>CPF: <input type="text" name="cpf"></p>
            <p>Nome: <input type="text" name="nome"></p>
            <input type="submit" value="Gravar">
      </form>

    </body>
</html>
```

PASSO 9: Precisamos, agora, criar o servlet de processamento ClienteProcess, que irá receber estes dados e armazená-lo no objeto Cliente. Clique com botão direito no pacote **cadcli** e selecione **Novo > Servlet**. Dê o nome **ClienteProcess** para o servlet e clique em **Próximo**. Marque a opção "**Adicionar informação ao descritor de implementação (web.xml)**" e clique em **Ok/Finalizar**.

PASSO 10. Já com o arquvo ClienteProcess.java aberto, remova tudo que é desnecessário, considerando que este é um servlet de processamento. A função deste Servlet é coletar os dados do formulário, modificar o objeto cliente que está na sessão e, se tudo correr bem, persistí-lo. Observe como isso pode ser feito no código a seguir.

ClienteProcess.java (método processRequest)

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            // Recupera cliente da sessão
            HttpSession sessao = request.getSession();
            Cliente c = (Cliente)sessao.getAttribute("cliente");
            // Se não há um cliente... erro fatal. Redireciona.
            if (c == null) response.sendRedirect("index.jsp");

            // Recupera dados do formulário
            String nome = request.getParameter("nome");
            String cpf = request.getParameter("cpf");

            // Insere dados no cliente
            boolean res = true
            if (c.setNome(nome) == false) {
                res = false;
            }
            else if (c.setCpf(cpf) == false) {
                res = false;
            }
```

```
            // Se tudo correu bem...
            if (res == true) {
                // Persiste objeto cliente
                ClienteDAO.adiciona(cliente);
                // Envia processamento para Mostra Cliente
                RequestDispatcher rd;
                rd = request.getRequestDispatcher("/ClienteView.jsp");
                rd.forward(request,response);
                return;
            }
            // Caso contrário...
            else {
                // Envia processamento de volta para tela de edição
               RequestDispatcher rd;
                rd = request.getRequestDispatcher("/ClienteEdit.jsp");
                rd.forward(request,response);
                return;
            }

        } finally {
        }
    }
```

PASSO 11: O toque final é modificar o fluxo da aplicação: o nosso ClienteCreate redirecionava a execução diretamente para o ClienteView.jsp. Vamos modificá-lo para redirecionar par o ClienteEdit.jsp:

ClienteCreate.java (método processRequest)

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

        try {

            // Cria um Cliente de Teste
            Cliente c = new Cliente();
            c.setNome("Fulano da Silva");
            c.setCpf("12345678901");

            // Requisita Objeto de Sessão
            HttpSession sessao = request.getSession();
            // Insere cliente na sessão
            sessao.setAttribute("cliente", c);

            // Solicita o despachante de requisições
            RequestDispatcher rd;
            rd = request.getRequestDispatcher("/ClienteEdit.jsp");
            // Encaminha requisição
            rd.forward(request,response);
            return;

        } finally {
        }
     }
```

PASSO 12: Experimente! Veja o que acontece quando você tenta editar um cliente e usar um CPF inválido!

3. COMPLEMENTANDO COM RECURSOS ADICIONAIS

Nosso sistema de edição tem dois problemas graves ainda:

1) Quando vou editar um cliente que já está na sessão, seus dados não são mostrados no formulário;
2) Quando erramos algo no formulário de edição, ele reaparece sem nenhuma indicação do erro.

Vaoms corrigir essas duas pendências.

PASSO 13: Primeiramente vamos modificar o ClienteEdit.jsp para que mostre os dados do cliente em seus campos - se o cliente existir, é claro...

ClienteEdit.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8" import="entidades.*" %>

<%
      String nome = "";
      String cpf = "";
      Cliente c = (Cliente)session.getCliente();
      if (c != null) {
             nome = c.getNome();
             cpf = c.getCpf();
      }
%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Cliente Edit</title>
    </head>
    <body>

      <form action="ClienteProcess" method="post">
             <p>CPF: <input type="text" name="cpf" value="<%= cpf %>"></p>
             <p>Nome: <input type="text" name="nome" value="<%= nome %>"></p>
             <input type="submit" value="Gravar">
      </form>

    </body>
</html>
```

PASSO 14: O segundo problema exige um pouco mais de modificações. Primeiramente, vamos fazer com que o servlet ClienteProcess armazene mensagens de erro na requisição, com a etiqueta **erros**. Como os erros podem ser muitos, vamos usar uma **ArrayList** para isso.

ClienteProcess.java (método processRequest)

```
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        try {
            // Recupera cliente da sessão
            HttpSession sessao = request.getSession();
            Cliente c = (Cliente)sessao.getAttribute("cliente");
            // Se não há um cliente... erro fatal. Redireciona.
            if (c == null) response.sendRedirect("index.jsp");
```

```
            // Recupera dados do formulário
            String nome = request.getParameter("nome");
            String cpf = request.getParameter("cpf");

            // Cria lista de erros
            ArrayList<String> erros = new ArrayList<String>();

            // Insere dados no cliente
            boolean res = true
            if (c.setNome(nome) == false) {
                res = false;
                erros.add("Nome inválido");
            }
            else if (c.setCpf(cpf) == false) {
                res = false;
                erros.add("CPF inválido");
            }

            // Se tudo correu bem...
            if (res == true) {
                // Persiste objeto cliente
                ClienteDAO.adiciona(cliente);
                // Envia processamento para Mostra Cliente
                RequestDispatcher rd;
                rd = request.getRequestDispatcher("/ClienteView.jsp");
                rd.forward(request,response);
                return;
            }
            // Caso contrário...
            else {
                // Adiciona mensagens de erro à requisição
                request.setAttribute("erros", erros);
                // Envia processamento de volta para tela de edição
                RequestDispatcher rd;
                rd = request.getRequestDispatcher("/ClienteEdit.jsp");
                rd.forward(request,response);
                return;
            }

        } finally {
        }
    }
```

PASSO 15: Agora vamos modificar o JSP ClienteEdit.jsp para que ele **mostre** os erros, caso eles existam!

ClienteEdit.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"
      import="entidades.*, java.util.*" %>

<%
      String nome = "";
      String cpf = "";
      Cliente c = (Cliente)session.getCliente();
      if (c != null) {
            nome = c.getNome();
            cpf = c.getCpf();
      }

      ArrayList<String> lista = (ArrayList<String>)request.getAttribute("erros");
%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
        <title>Cliente Edit</title>
    </head>
    <body>

<%
      if (lista != null && lista.size() > 0) {
            int i;
            out.println("<p>Por favor, corrija os erros indicados abaixo:</p>");
            for (i = 0; i < lista.size(); i++) {
                  out.println("<p>" + lista.get(i) + "</p>");
            }
      }
%>

      <form action="ClienteProcess" method="post">
            <p>CPF: <input type="text" name="cpf" value="<%= cpf %>"></p>
            <p>Nome: <input type="text" name="nome" value="<%= nome %>"></p>
            <input type="submit" value="Gravar">
      </form>

    </body>
</html>
```

NOTA: OBSERVE que as mensagens de erro não estão sendo colcoadas na sessão! Isso ocorre porque **não** queremos que a mensagem de erro fique registrada permanentemente.... ela é **temporária**. A sessão é usada apenas para informações **que devem ser mantidas ao longo da execução, permanentes**, o que não é o caso de uma mensagem de erro.

PASSO 16: Para arrematar, vamos voltar à classe ClienteCreate para remover as informações iniciais que estão sendo preenchdias automaticamente no cliente!

ClienteCreate.java (método processRequest)

```
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
     throws ServletException, IOException {

         try {

             // Cria um Cliente de Teste
             Cliente c = new Cliente();
             c.setNome("Fulano da Silva");
             c.setCpf("12345678901");

             // Requisita Objeto de Sessão
             HttpSession sessao = request.getSession();
             // Insere cliente na sessão
             sessao.setAttribute("cliente", c);

             // Solicita o despachante de requisições
             RequestDispatcher rd;
             rd = request.getRequestDispatcher("/ClienteView.jsp");
             // Encaminha requisição
             rd.forward(request,response);
             return;

         } finally {
         }
     }
```

4. ATIVIDADE

Agora que criamos um elemento mais completo de um sistema, você saberia protegê-lo com o sistema de logon criado na aula passada?

Experimente copiar o pacote "logon" do WProjeto8, feito na aula 11 para os pacotes de código fonte do projeto atual (WProjeto9). Não se esqueça de criar as entradas necessárias no **web.xml** pois, caso contrário, você não será capaz de usar os servlets /WLogon e /WLogoff.

Modifique o index.jsp para conter um formulário de login que redirecione para o servlet ClienteCreate em caso de sucesso e acrescente a linha de verificação de logon em todos os Servlets e JSPs do sistema!

Tente!

5. BIBLIOGRAFIA

QIAN, K; ALLEN, R; GAN, M; BROWN, R. **Desenvolvimento Web Java**. Rio de Janeiro: LTC, 2007.

DEITEL, H.M; DEITEL, P.J. **Java: como programar** - Sexta edição. São Paulo: Pearson-Prentice Hall, 2005.