



POLO SETOR O – CEILÂNDIA – DF

DESENVOLVIMENTO FULL STACK

VAMOS INTEGRAR SISTEMAS

2024.2

MAYCON MOURA

Título da Prática

Interface Cadastral com Servlet e JSPs

Objetivo da Prática

O objetivo desta prática é entender e aplicar o padrão Front Controller na arquitetura MVC (Model-View-Controller) em uma aplicação Web Java. Serão abordadas as diferenças e semelhanças entre Servlets e JSPs, e as formas de redirecionamento e despacho de requisições em um ambiente web, além do uso de parâmetros e atributos nos objetos HttpRequest.

Códigos

ProdutoLista.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Lista de Produtos</title>
  </head>
  <body>
    <h1>Lista de Produtos</h1>

    <table border="1">
      <thead>
        <tr>
          <th>ID</th>
          <th>Nome</th>
        </tr>
      </thead>
    </table>
  </body>
</html>
```

```

        <th>Preço</th>
        <th>Ações</th>
    </tr>
</thead>
<tbody>
    <c:forEach var="produto" items="${produtos}">
        <tr>
            <td>${produto.id}</td>
            <td>${produto.nome}</td>
            <td>${produto.preco}</td>
            <td>
                <a
href="ServletProdutoFC?acao=formAlterar&id=${produto.id}">Editar</a> |
                <a href="ServletProdutoFC?acao=excluir&id=${produto.id}"
onclick="return confirm('Tem certeza que deseja excluir este produto?');">Excluir</a>
            </td>
        </tr>
    </c:forEach>
</tbody>
</table>

<br>
<a href="ServletProdutoFC?acao=formIncluir">Incluir Novo Produto</a>
</body>
</html>

```

ProdutoDados.jsp

```

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Cadastro de Produto</title>
</head>
<body>
    <h1>${produto == null ? "Incluir Novo Produto" : "Alterar Produto"}</h1>

    <form action="ServletProdutoFC" method="post">
        <%
            // Recupera a entidade 'produto' do request
            cadastroee.model.Produto produto = (cadastroee.model.Produto)
request.getAttribute("produto");
            String acao = (produto == null) ? "incluir" : "alterar";
        %>

        <!-- Campo hidden para a ação -->
        <input type="hidden" name="acao" value="<%= acao %>">

```

```

<!-- Campo hidden para o ID (apenas se acao for alterar) -->
<c:if test="${produto != null}">
    <input type="hidden" name="id" value="${produto.id}">
</c:if>

<!-- Campo para o nome do produto -->
<div>
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome" value="${produto != null ?
produto.nome : ''}" required>
</div>

<!-- Campo para a quantidade do produto -->
<div>
    <label for="quantidade">Quantidade:</label>
    <input type="number" id="quantidade" name="quantidade" value="${produto
!= null ? produto.quantidade : ''}" required>
</div>

<!-- Campo para o preço de venda do produto -->
<div>
    <label for="preco">Preço de Venda:</label>
    <input type="text" id="preco" name="preco" value="${produto != null ?
produto.precoVenda : ''}" required>
</div>

<!-- Botão de envio -->
<div>
    <button type="submit">${produto == null ? "Incluir" : "Alterar"}</button>
</div>
</form>
</body>
</html>

```

ServletProdutoFC.java

```

package cadastroee.servlets;

import cadastroee.controller.ProdutoFacadeLocal;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.RequestDispatcher;
import jakarta.ejb.EJB;
import cadastroee.model.Produto;
import java.io.IOException;

```

```

import java.util.List;

@WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
public class ServletProdutoFC extends HttpServlet {

    @EJB
    private ProdutoFacadeLocal facade;

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

        String acao = request.getParameter("acao");
        String destino = "ProdutoLista.jsp";

        if (acao == null) {
            acao = "listar";
        }

        switch (acao) {
            case "listar":
                List<Produto> produtos = facade.findAll();
                request.setAttribute("produtos", produtos);
                destino = "ProdutoLista.jsp";
                break;

            case "formIncluir":
                destino = "ProdutoDados.jsp";
                break;

            case "formAlterar":
                int idAlterar = Integer.parseInt(request.getParameter("id"));
                Produto produtoAlterar = facade.find(idAlterar);
                request.setAttribute("produto", produtoAlterar);
                destino = "ProdutoDados.jsp";
                break;

            case "incluir":
                Produto produtoIncluir = new Produto();
                produtoIncluir.setNome(request.getParameter("nome"));

                produtoIncluir.setPrecoVenda(Float.parseFloat(request.getParameter("preco")));

                facade.find(produtoIncluir);
                request.setAttribute("produtos", facade.findAll());
                destino = "ProdutoLista.jsp";
                break;
        }
    }
}

```

```

        case "excluir":
            int idExcluir = Integer.parseInt(request.getParameter("id"));
            facade.remove(idExcluir);
            request.setAttribute("produtos", facade.findAll());
            destino = "ProdutoLista.jsp";
            break;

        default:
            destino = "ProdutoLista.jsp";
            break;
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/views/" + destino);
    dispatcher.forward(request, response);
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

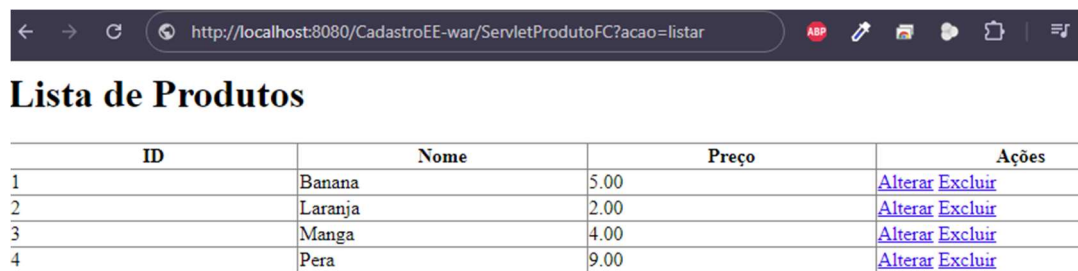
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    processRequest(request, response);
}

@Override
public String getServletInfo() {
    return "Front Controller Servlet for Product Management";
}
}

```

Resultados

Lista de Produtos



ID	Nome	Preço	Ações
1	Banana	5.00	Alterar Excluir
2	Laranja	2.00	Alterar Excluir
3	Manga	4.00	Alterar Excluir
4	Pera	9.00	Alterar Excluir

Cadastro Produto.



Dados do Produto

Nome:

Quantidade:

Preço de Venda:

Análise e Conclusão

Como funciona o padrão Front Controller, e como ele é implementado em um aplicativo Web Java, na arquitetura MVC?

O padrão Front Controller é um padrão de design que centraliza o controle das requisições para uma aplicação Web em um único ponto. Em vez de ter múltiplos pontos de entrada (Servlets) para processar diferentes tipos de requisições, o Front Controller lida com todas as requisições e as encaminha para o componente apropriado.

Na arquitetura MVC, o Front Controller atua como o controlador central (Controller). A implementação do padrão Front Controller em uma aplicação Web Java geralmente é feita com um Servlet que intercepta todas as requisições para a aplicação. Esse Servlet, chamado de Front Controller, é configurado para receber todas as requisições (ou um subconjunto delas) e encaminhar para outros componentes (como Model e View) com base na lógica de negócio.

Exemplo de implementação:

1. **Definir o Front Controller (Servlet):** Um Servlet, por exemplo, ServletProdutoFC, é configurado para capturar todas as requisições para a aplicação.
2. **Processar a Requisição:** O Servlet Front Controller analisa os parâmetros da requisição e decide qual ação tomar (listar produtos, incluir, alterar, excluir, etc.).
3. **Encaminhar para a Visão:** Dependendo da ação, o Front Controller pode encaminhar a requisição para uma página JSP específica para renderizar a resposta.

Benefícios:

- Centralização do controle das requisições.
- Facilidade de manutenção e extensibilidade.
- Separação clara entre a lógica de controle e a lógica de apresentação.

Quais as diferenças e semelhanças entre Servlets e JSPs?

Semelhanças:

- Ambos são componentes da tecnologia Java EE para desenvolvimento de aplicações Web.
- Servlets e JSPs são executados no lado do servidor e são usados para gerar conteúdo dinâmico.

Diferenças:

- **Servlets:**
 - Servlets são classes Java que processam requisições HTTP e geram respostas HTTP.
 - Servlets são usados principalmente para lógica de controle e processamento de dados.
 - Servlets podem gerar conteúdo HTML, mas o código HTML é misturado com o código Java, o que pode dificultar a manutenção.
- **JSPs (JavaServer Pages):**
 - JSPs são arquivos que contêm HTML e código Java, onde o código Java é escrito entre tags específicas.
 - JSPs são usados principalmente para a apresentação da interface do usuário e são mais fáceis de manter, pois separam a lógica de apresentação da lógica de controle.
 - O código Java em uma JSP é convertido em um Servlet pelo servidor durante o processo de compilação.

Qual a diferença entre um redirecionamento simples e o uso do método forward, a partir do RequestDispatcher? Para que servem parâmetros e atributos nos objetos HttpRequest?

Redirecionamento (Redirect):

- O redirecionamento é feito utilizando o método `response.sendRedirect()`.
- Quando o redirecionamento é executado, o servidor instrui o navegador do cliente a fazer uma nova requisição para uma URL diferente.
- O redirecionamento é uma forma de navegar para uma nova URL e pode resultar em uma nova requisição HTTP.

Forward (Despacho):

- O despacho é feito utilizando o método `RequestDispatcher.forward()`.
- Quando o forward é executado, o servidor encaminha a requisição para outro recurso (como um Servlet ou JSP) dentro da mesma aplicação sem que o cliente perceba.
- O forward mantém a mesma requisição e resposta, permitindo que o recurso de destino acesse os dados da requisição original.

Parâmetros e Atributos no HttpRequest:

- **Parâmetros:** São informações enviadas com a requisição (por exemplo, através de URL ou formulário). Eles são acessados usando `request.getParameter()`.
- **Atributos:** São informações armazenadas na requisição para serem usadas por outros componentes, como Servlets ou JSPs. São definidos usando `request.setAttribute()` e acessados usando `request.getAttribute()`.