Disciplina: Nível 4: Vamos Integrar Sistemas
Semestre: 1º - 2024
Aluno: DAVIDSON PEREIRA DE OLIVEIRA
Matrícula: 2022.11.54585-5
Repositório no GIT: dev-davidson/RPG0017---Vamos-integrar-sistemas (github.com)

Relatório discente de acompanhamento

👉 2º Procedimento | Interface Cadastral com Servlet e JSPs

Códigos:

Servlets/ServletProdutoFC

```
package cadastroee.servlets;

import java.io.IOException;
import java.util.List;

import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import cadastroee.model.Produto;
import cadastroee.controller.ProdutoFacadeLocal;

public class ServletProdutoFC extends HttpServlet {
    @EJB
    private ProdutoFacadeLocal facade;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String acao = request.getParameter("acao");
        if (acao == null) {
            acao = "listar";
        }

        Integer idProduto;
        Produto produto;
        String destino = "ProdutoLista.jsp";
```

```
        switch (acao) {
            case "listar":
                List<Produto> listaProdutos = facade.findAll();
                request.setAttribute("listaProdutos", listaProdutos);
                break;
            case "excluir":
                idProduto = Integer.parseInt(request.getParameter("idProduto"));
                produto = facade.find(idProduto);
                facade.remove(produto);
                listaProdutos = facade.findAll();
                request.setAttribute("listaProdutos", listaProdutos);
                break;
            case "alterar":
                idProduto = Integer.parseInt(request.getParameter("idProduto"));
                produto = facade.find(idProduto);
                produto.setNome(request.getParameter("nome"));
                produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
                produto.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
                facade.edit(produto);
                listaProdutos = facade.findAll();
                request.setAttribute("listaProdutos", listaProdutos);
                break;
            case "incluir":
                produto = new Produto();
                produto.setNome(request.getParameter("nome"));
                produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
                produto.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
```

```
                facade.create(produto);
                listaProdutos = facade.findAll();
                request.setAttribute("listaProdutos", listaProdutos);
                break;
            case "formIncluir":
                destino = "ProdutoDados.jsp";
                break;
            case "formAlterar":
                idProduto = Integer.parseInt(request.getParameter("idProduto"));
                produto = facade.find(idProduto);
                request.setAttribute("produto", produto);
                destino = "ProdutoDados.jsp";
                break;

        }
        request.getRequestDispatcher(destino).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 "Short description";
    }
}
```

Servlets/ServletProduto

```
package cadastroee.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

import jakarta.ejb.EJB;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;

public class ServletProduto extends HttpServlet {
    @EJB
    private ProdutoFacadeLocal facade;

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet ServletProduto</title>");
            out.println("</head>");
```

```
            out.println("<body>");
            out.println("<h1>Servlet ServletProduto at " + request.getContextPath() + "</h1>");
            out.println("</body>");
            out.println("</html>");
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        List<Produto> produtos = facade.findAll();

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Lista de Produtos</h1>");
        out.println("<ul>");

        for (Produto produto : produtos) {
            out.println("<li>" + produto.getNome() + "</li>");
        }

        out.println("</ul>");
        out.println("</body></html>");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
```

```
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }
}
```

web/ProdutoLista

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Listagem de Produtos</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
</head>
<body class="container">
    <h1>Listagem de Produtos</h1>

    <a href="ServletProdutoFC?acao=formIncluir" class="btn btn-primary m-2">Novo Produto</a>

    <table class="table table-striped">
        <thead class="table-dark">
            <tr>
                <th>#</th>
                <th>Nome</th>
                <th>Quantidade</th>
                <th>Preço de Venda</th>
                <th>Opções</th>
            </tr>
        </thead>

        <tbody>
            <c:forEach items="${listaProdutos}" var="produto">
                <tr>
                    <td>${produto.idProduto}</td>
                    <td>${produto.nome}</td>
                    <td>${produto.quantidade}</td>
                    <td>${produto.precoVenda}</td>
```

```
                    <!-- Links para alteração e exclusão -->
                    <td>
                        <a class="btn btn-primary btn-sm" href="ServletProdutoFC?
acao=formAlterar&idProduto=${produto.idProduto}">Alterar</a> |
                        <a class="btn btn-danger btn-sm" href="ServletProdutoFC?
acao=excluir&idProduto=${produto.idProduto}">Excluir</a>
                    </td>
                </tr>
            </c:forEach>
            <c:if test="${empty listaProdutos}">
                <tr>
                    <td colspan="5">Nenhum produto encontrado.</td>
                </tr>
            </c:if>
        </tbody>
    </table>
</body>
</html>
```

web/ProdutoDados

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Dados do Produto</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-
HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
</head>
<body class="container">
    <h1>Dados do Produto</h1>
    <form action="ServletProdutoFC" method="post" class="form">
        <input type="hidden" name="acao" value="${empty produto ? 'incluir' : 'alterar'}">
        <c:if test="${not empty produto}">
            <input type="hidden" name="idProduto" value="${produto.idProduto}">
        </c:if>
        <div class="mb-3">
            <label for="nome" class="form-label">Nome:</label>
            <input class="form-control" type="text" name="nome" id="nome" value="${not empty produto ? produto.nome :
''}">
        </div>
        <div class="mb-3">
            <label for="quantidade" class="form-label">Quantidade:</label>
            <input class="form-control"  type="number" name="quantidade" id="quantidade" value="${not empty produto ?
produto.quantidade : ''}">
        </div>
        <div class="mb-3">
            <label for="precoVenda" class="form-label">Preço de Venda:</label>
            <input class="form-control"  type="number" name="precoVenda" id="precoVenda" min="0" step="0.01"
value="${not empty produto ? produto.precoVenda : ''}">
        </div>
        <button type="submit" class="btn btn-primary">${empty produto ? 'Adicionar' : 'Alterar'} Produto</button>
    </form>
</body>
</html>
```

Análise e Conclusão

a) 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 projeto usado em desenvolvimento web para centralizar o processamento de todas as requisições que chegam a uma aplicação. Na arquitetura MVC (Model-View-Controller), o Front Controller é o componente responsável por receber todas as requisições do cliente, determinar qual ação deve ser executada com base nas informações da requisição e despachar essa requisição para o controlador apropriado.

Em um aplicativo web Java, o Front Controller é geralmente implementado usando Servlets. Um Servlet é uma classe Java que estende a classe javax.servlet.http.HttpServlet e é mapeada para uma URL específica no arquivo de configuração do aplicativo web (como o arquivo web.xml ou por meio de anotações @WebServlet no Java EE 6 e versões posteriores).

O Servlet atua como o ponto de entrada para todas as requisições HTTP recebidas pelo aplicativo web. Ele processa essas requisições, executa a lógica de negócios necessária (ou delega essa responsabilidade para outros componentes, como controladores), e então envia uma resposta de volta ao cliente.

b) Quais as diferenças e semelhanças entre Servlets e JSPs?

- Diferenças:

Servlets são classes Java que processam requisições HTTP e geram respostas dinamicamente, enquanto JSPs (JavaServer Pages) são arquivos que contêm mistura de código HTML e Java, permitindo a geração de conteúdo dinâmico.

Servlets são orientados a controle, enquanto JSPs são orientados a visualização.

Em um Servlet, a lógica de negócios e a lógica de apresentação são geralmente separadas, enquanto em um JSP, a lógica de apresentação e o código Java podem estar misturados.

- Semelhanças:

Ambos os Servlets e JSPs são componentes de aplicativos web Java que podem ser usados para criar aplicativos dinâmicos.

Ambos podem acessar objetos de solicitação e resposta HTTP, bem como interagir com outras partes do aplicativo, como sessões e escopos de aplicativo.

Tanto Servlets quanto JSPs podem ser usados em conjunto para criar aplicativos web Java robustos e escaláveis.

c) 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 simples (sendRedirect): O redirecionamento simples envia um cabeçalho de resposta HTTP para o navegador, instruindo-o a fazer uma nova solicitação para uma URL diferente. Isso faz com que o navegador envie uma nova solicitação ao servidor para a nova URL, resultando em duas solicitações separadas. O redirecionamento é realizado usando o método sendRedirect() do objeto HttpServletResponse.

Forward (encaminhamento): O encaminhamento (forward) é feito no lado do servidor e é transparente para o navegador. O servidor despacha internamente a requisição do cliente para outra parte do aplicativo web antes de enviar a resposta de volta ao cliente. Isso é feito usando o método forward() do objeto RequestDispatcher. O encaminhamento é mais eficiente que o redirecionamento, pois não envolve uma nova solicitação HTTP do cliente.

Os parâmetros e atributos nos objetos HttpRequest são usados para transportar dados entre diferentes partes da aplicação web. Os parâmetros são informações específicas da requisição HTTP, como dados de formulário enviados pelo cliente, enquanto os atributos são objetos Java que podem ser armazenados na requisição, na sessão ou no escopo de aplicativo e acessados em diferentes partes da aplicação durante o ciclo de vida da requisição. Os atributos podem ser usados para compartilhar dados entre Servlets, JSPs e outros componentes do aplicativo web.