Estácio - Unidade São pedro
RJ 140 Km 2, 512 loja 1, São Pedro da Aldeia - RJ, 28941-182

Desenvolvimento Full Stack
Classe: Missão Prática | Nível 4 | Mundo 3
3º Semestre
Marvin de Almeida Costa

Título da Prática: 2º Procedimento | Interface Cadastral com Servlet e JSPs

Objetivos da prática:

- Implementar persistência com base em JPA.
- Implementar regras de negócio na plataforma JEE, através de EJBs.
- Implementar sistema cadastral Web com base em Servlets e JSPs.
- Utilizar a biblioteca Bootstrap para melhoria do design.
- No final do exercício, o aluno terá criado todos os elementos necessários para
- exibição e entrada de dados na plataforma Java Web, tornando-se capacitado para
- lidar com contextos reais de aplicação.

Todos os códigos solicitados neste roteiro de aula:

web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
>

```
 <servlet>

  <servlet-name>ServletProduto</servlet-name>

  <servlet-class>cadastroee.servlets.ServletProduto</servlet-class>

 </servlet>

 <servlet>

  <servlet-name>ServletProdutoFC</servlet-name>

  <servlet-class>cadastroee.servlets.ServletProdutoFC

  </servlet-class>

 </servlet>

 <session-config>

  <session-timeout>30</session-timeout>

 </session-config>

<servlet-mapping>
   <servlet-name>ServletProduto</servlet-name>
   <url-pattern>/ServletProduto</url-pattern>
</servlet-mapping>

<servlet-mapping>
   <servlet-name>ServletProdutoFC</servlet-name>
   <url-pattern>/ServletProdutoFC</url-pattern>
</servlet-mapping>

</web-app>
```

ProdutoDados.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>
```

```
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Cadastro de Produto</title>
    <style>
        label {
            display: block;
            margin-bottom: 10px;
        }
        input[type="text"], input[type="number"] {
            width: 100%;
            height: 30px;
            margin-bottom: 20px;
            padding: 10px;
            box-sizing: border-box;
        }
        .btn {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            cursor: pointer;
        }
        .btn:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Cadastro de Produto</h1>

    <form action="${pageContext.request.contextPath}/ServletProdutoFC" method="post">
        <input type="hidden" name="acao" value="${acao}">
        <input type="hidden" name="redirigir" value="true">
        <c:if test="${acao == 'alterar'}">
            <input type="hidden" name="id" value="${produto.idProduto}">
        </c:if>
        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome" value="${nomeproduto}">

        <label for="quantidade">Quantidade:</label>
        <input type="number" id="quantidade" name="quantidade" value="${produto.quantidade}">

        <label for="precoVenda">Preço de Venda:</label>
```

```
        <input type="number" id="precoVenda" name="precoVenda"
value="${produto.precoVenda}">

        <c:if test="${acao == 'incluir'}">
            <input type="submit" value="Cadastrar" class="btn">
        </c:if>
        <c:if test="${acao == 'alterar'}">
            <input type="submit" value="Alterar" class="btn">
        </c:if>
    </form>
</body>
</html>
```

ProdutoLista.jsp

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <title>Listagem de Produtos</title>
    <style>
        table {
            border-collapse: collapse;
            width: 100%;
        }
        th, td {
            border: 1px solid black;
            padding: 8px;
            text-align: center;
        }
        .btn {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            text-decoration: none;
            cursor: pointer;
        }
        .btn:hover {
            background-color: #45a049;
        }
    </style>
```

```
</head>
<body>
   <h1>Listagem de Produtos</h1>

   <a href="${pageContext.request.contextPath}/ServletProdutoFC?acao=incluir"
     class="btn">Novo Produto</a>

   <table>
      <tr>
         <th>ID</th>
         <th>Nome</th>
         <th>Quantidade</th>
         <th>Preço de Venda</th>
         <th>Opções</th>
      </tr>

      <c:forEach var="produto" items="${produtos}">
         <tr>
            <td>${produto.idProduto}</td>
            <td>${produto.nome}</td>
            <td>${produto.quantidade}</td>
            <td>R$ ${produto.precoVenda}</td>
            <td>
               <a
href="${pageContext.request.contextPath}/ServletProdutoFC?acao=alterar&id=${produto.idPro
duto}">Alterar</a>
                  |
                  <a
href="${pageContext.request.contextPath}/ServletProdutoFC?acao=excluir&id=${produto.idPro
duto}"
                     onclick="return confirm('Tem certeza que deseja excluir esse
produto?')">Excluir</a>
            </td>
         </tr>
      </c:forEach>
   </table>
</body>
</html>
```

ServletProdutoFC.java

```
/*
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this
template
 */
package cadastroee.servlets;


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

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 = "";

        switch (acao) {
            case "listar":
                destino = "ProdutoLista.jsp";
                break;
            case "incluir":
            case "alterar":
                destino = "ProdutoDados.jsp";
                break;
            default:
                // Caso não seja uma ação válida, redireciona para listar
                destino = "ProdutoLista.jsp";
        }

        try {
            switch (acao) {
```

```
            case "listar":
                listarResultados(request);
                break;
            case "incluir":
                incluirProduto(request, acao);
                if (request.getParameter("redirigir") != null) {
                    destino = "ServletProdutoFC?acao=listar";
                }
                break;
            case "excluir":
                excluirProduto(request);
                break;
            case "alterar":
                alterarProduto(request, acao);
                if (request.getParameter("redirigir") != null) {
                    destino = "ServletProdutoFC?acao=listar";
                }
                break;
        }
    } catch (Exception e) {
        request.setAttribute("mensagemErro", "Ocorreu um erro: " + e.getMessage());
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher(destino);
    dispatcher.forward(request, response);
}

private void listarResultados(HttpServletRequest request) {
    List<Produto> produtos = facade.findAll();
    System.out.println("N produtos: " + produtos.size());
    request.setAttribute("produtos", produtos);
}

private void incluirProduto(HttpServletRequest request, String acao) {
    Produto produto = new Produto();
    request.setAttribute("acao", acao);
    produto.setNome(request.getParameter("nome"));
    produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
    produto.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
    facade.create(produto);
}

private void alterarProduto(HttpServletRequest request, String acao) {
    int id = Integer.parseInt(request.getParameter("id"));
```

```
        request.setAttribute("acao", acao);
        Produto produto = facade.find(id);
        request.setAttribute("nomeproduto", produto.nome);
        request.setAttribute("produto", produto);
        produto.setNome(request.getParameter("nome"));
        produto.setQuantidade(Integer.parseInt(request.getParameter("quantidade")));
        produto.setPrecoVenda(Float.parseFloat(request.getParameter("precoVenda")));
        facade.edit(produto);
    }

    private void excluirProduto(HttpServletRequest request) {
        int id = Integer.parseInt(request.getParameter("id"));
        Produto produto = facade.find(id);
        facade.remove(produto);
        listarResultados(request);
    }

    @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);
    }
}
```

ServletProduto.java

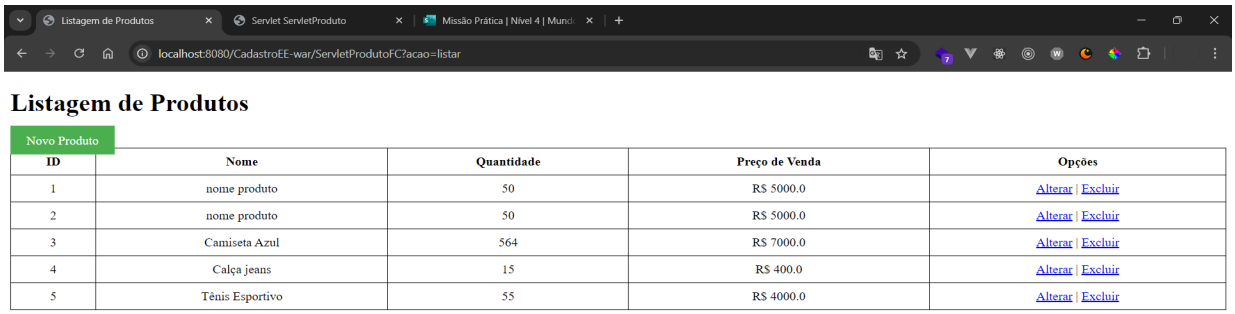
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this
template
 */
package cadastroee.servlets;

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

/**
 *
 * @author marvin
 */
public class ServletProduto extends HttpServlet {

    @EJB
    ProdutoFacadeLocal facade;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    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>Lista de Produtos</h1>");
            out.println("<ul>");

            // Utilizar o facade para recuperar os dados
            List<Produto> produtos = facade.findAll();

            // Apresentar os dados em uma lista HTML
            for (Produto produto : produtos) {
```

```
            out.println("<li>" + produto.getNome() + "</li>");
        }

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

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
```

```
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

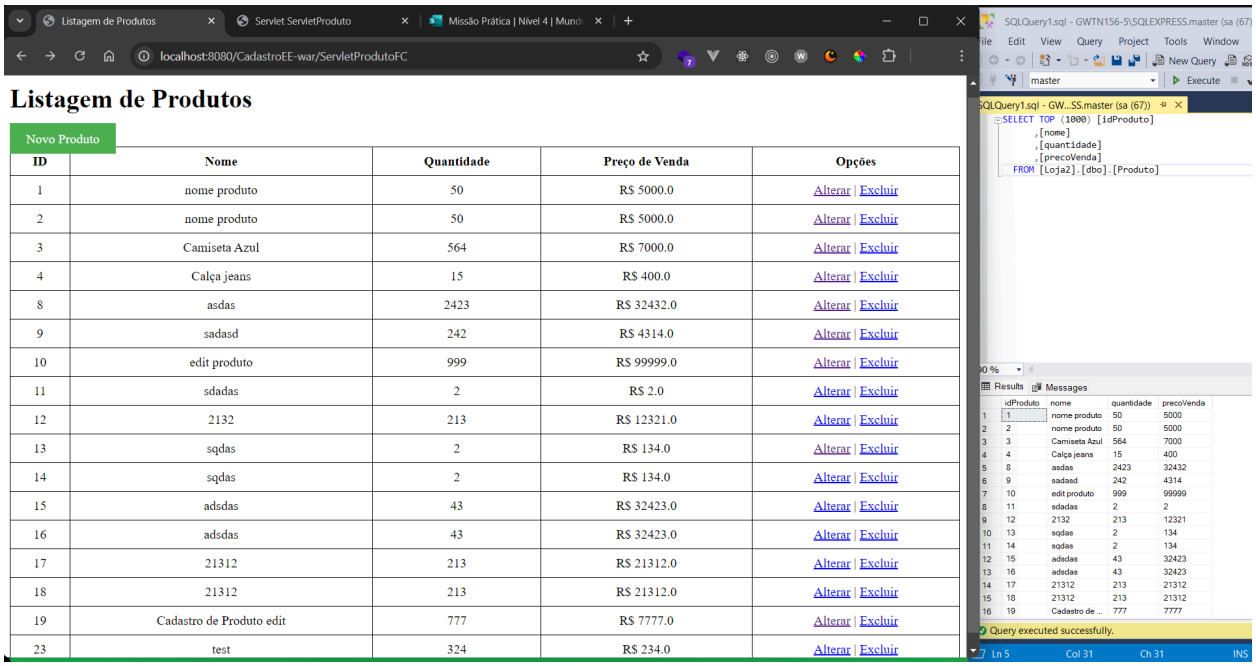
Os resultados da execução dos códigos também devem ser apresentados;



Listagem de Produtos

Novo Produto

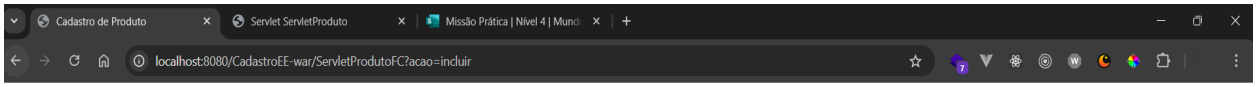
ID	Nome	Quantidade	Preço de Venda	Opções
1	nome produto	50	R$ 5000.0	Alterar | Excluir
2	nome produto	50	R$ 5000.0	Alterar | Excluir
3	Camiseta Azul	564	R$ 7000.0	Alterar | Excluir
4	Calça jeans	15	R$ 400.0	Alterar | Excluir
5	Tênis Esportivo	55	R$ 4000.0	Alterar | Excluir



Listagem de Produtos

Novo Produto

ID	Nome	Quantidade	Preço de Venda	Opções
1	nome produto	50	R$ 5000.0	Alterar | Excluir
2	nome produto	50	R$ 5000.0	Alterar | Excluir
3	Camiseta Azul	564	R$ 7000.0	Alterar | Excluir
4	Calça jeans	15	R$ 400.0	Alterar | Excluir
8	asdas	2423	R$ 32432.0	Alterar | Excluir
9	sadasd	242	R$ 4314.0	Alterar | Excluir
10	edit produto	999	R$ 99999.0	Alterar | Excluir
11	sdadas	2	R$ 2.0	Alterar | Excluir
12	2132	213	R$ 12321.0	Alterar | Excluir
13	sqdas	2	R$ 134.0	Alterar | Excluir
14	sqdas	2	R$ 134.0	Alterar | Excluir
15	adsdas	43	R$ 32423.0	Alterar | Excluir
16	adsdas	43	R$ 32423.0	Alterar | Excluir
17	21312	213	R$ 21312.0	Alterar | Excluir
18	21312	213	R$ 21312.0	Alterar | Excluir
19	Cadastro de Produto edit	777	R$ 7777.0	Alterar | Excluir
23	test	324	R$ 234.0	Alterar | Excluir



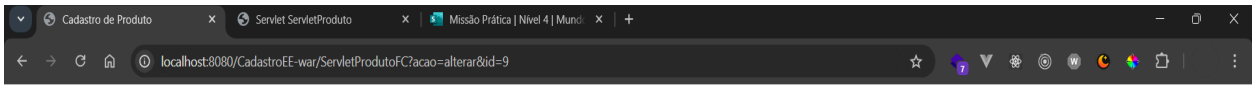
Cadastro de Produto

Nome:

Quantidade:

Preço de Venda:

Cadastrar



Cadastro de Produto

Nome:

sadasd

Quantidade:

242

Preço de Venda:

4314,0

Alterar

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 uma abordagem importante na arquitetura MVC (Model-View-Controller) para gerenciar requisições HTTP em aplicações web. Ele atua como um intermediário entre o cliente e os controladores, fornecendo uma camada de apresentação unificada para toda a aplicação. Vamos explorar como funciona o Front Controller e como ele é implementado em um aplicativo web Java usando a arquitetura MVC.

Funcionamento do Front Controller
O Front Controller opera da seguinte maneira:

1. Captura todas as requisições HTTP antes que elas cheguem aos controladores individuais.
2. Processa essas requisições, possivelmente realizando tarefas comuns como autenticação, logging ou manipulação de sessões.
3. Distribui as requisições processadas para os controladores apropriados.
4. Gerencia a resposta final, combinando a lógica de negócios com a apresentação.

Quais as diferenças e semelhanças entre Servlets e JSPs?

Servlets e JSPs (JavaServer Pages) são tecnologias importantes no desenvolvimento de aplicações web com Java. Embora ambos sejam usados para criar páginas dinâmicas em um aplicativo web, existem algumas diferenças significativas entre eles. Vamos explorar as principais diferenças e semelhanças:

Diferenças

1. Estrutura e sintaxe:

- Servlets são classes Java que implementam interfaces específicas do Java Servlet API.
- JSPs são arquivos HTML que contêm código Java embutido.

2. Funcionalidade principal:

- Servlets são focados na lógica de negócios e processamento de requisições HTTP.
- JSPs são mais voltados para a apresentação da informação e renderização de páginas.

3. Ciclo de vida:

- Um servlet tem um ciclo de vida controlado pelo contêiner do servlet, sendo criado quando o servidor é iniciado e destruído quando ele é desligado.
- Uma página JSP pode ser compilada e recompilada conforme necessário durante a execução do aplicativo.

4. Flexibilidade:

- Servlets oferecem maior flexibilidade para implementar lógica complexa e manipular dados.
- JSPs são mais simples e diretos, permitindo uma separação clara entre a lógica e a apresentação.

5. Desenvolvimento:

- O desenvolvimento de servlets requer conhecimento avançado de programação Java.
- A criação de JSPs é mais acessível, mesmo para desenvolvedores menos experientes.

6. Performance:

- Em geral, os servlets podem ser mais eficientes em termos de performance, especialmente para operações intensivas de processamento.
- As JSPs podem ter um overhead menor para tarefas simples de renderização de páginas.

Semelhanças

1. Ambiente de execução:

- Ambos são executados dentro do ambiente de um servidor web Java (como Tomcat, Jetty ou GlassFish).

2. Uso de expressões EL (Expression Language):

- Ambos suportam expressões EL para realizar operações simples dentro do contexto da página.

3. Integração com frameworks:

- Pode-se integrar tanto servlets quanto JSPs com frameworks populares como Spring MVC ou Struts.

4. Suporte a tags personalizadas:

- É possível criar tags personalizadas para uso em ambas as tecnologias.

5. Gerenciamento de sessão:

- Ambos podem participar do gerenciamento de sessão do usuário através do objeto HttpSession.

6. Tratamento de exceções:

- Tanto servlets quanto JSPs podem lidar com exceções usando blocos try-catch ou tags de tratamento de erros.

7. Compartilhamento de recursos:

- Podem compartilhar objetos e variáveis entre si, facilitando a comunicação entre a lógica e a apresentação.

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?

Para responder à sua pergunta sobre a diferença entre redirecionamentos simples e o uso do método forward através do RequestDispatcher, bem como os usos de parâmetros e atributos em objetos HttpRequest, vamos dividir a resposta em diferentes seções:

Diferença entre redirecionamento simples e forward

O redirecionamento simples e o método forward são dois mecanismos diferentes para transferir o controle de uma requisição em Java Servlets, mas eles têm propósitos e comportamentos distintos:

1. Redirecionamento Simples:

- Realiza uma mudança temporária da URL exibida no navegador.
- Cria uma nova requisição HTTP.
- O cliente (navegador) é redirecionado para uma nova página.
- É útil quando você quer que o usuário seja redirecionado para outra página após realizar alguma ação.

2. Forward (Método do RequestDispatcher):

- Mantém a mesma URL na barra de endereços do navegador.
- Não cria uma nova requisição HTTP; sim, passa o controle para outro componente.
- O conteúdo é enviado diretamente ao cliente sem alterar o URL exibido.
- É útil quando você quer transferir o controle para outro componente dentro da mesma aplicação sem alterar o endereço na barra de endereços.

Uso de parâmetros e atributos em objetos HttpRequest

Parâmetros e atributos são elementos importantes nos objetos HttpRequest que permitem transmitir informações entre diferentes partes da aplicação ou entre a aplicação e o cliente.

Parâmetros

Os parâmetros são dados enviados pelo cliente (navegador) junto com a requisição HTTP. Eles podem ser acessados através do objeto ServletRequest.

Uso comum:

- Para obter parâmetros GET: request.getParameter("nomeParametro")
- Para obter parâmetros POST: request.getParameter("nomeParametro")

Atributos

Os atributos são dados adicionais que podem ser armazenados no objeto ServletRequest durante a execução da requisição. Eles são úteis para passar informações entre diferentes componentes da aplicação.

Uso comum:

- Adicionar um atributo: request.setAttribute("nomeAtributo", valor)
- Obter um atributo: Object valor = request.getAttribute("nomeAtributo")

Quando usar cada uma dessas abordagens

- Use redirecionamento simples quando você quer que o usuário seja redirecionado para outra página após realizar alguma ação.
- Use forward quando você precisa transferir o controle para outro componente dentro da mesma aplicação sem alterar o endereço na barra de endereços.
- Use parâmetros para transmitir dados enviados pelo cliente junto com a requisição.

- Use atributos para passar informações entre diferentes componentes da aplicação durante a execução da requisição.