



Universidade Estácio
Campus Vargem Grande Paulista
Curso: Desenvolvimento Full Stack
Disciplina: Vamos Integrar Sistemas
Turma: 2024.3
3º semestre letivo
Marcia da Silva e Souza

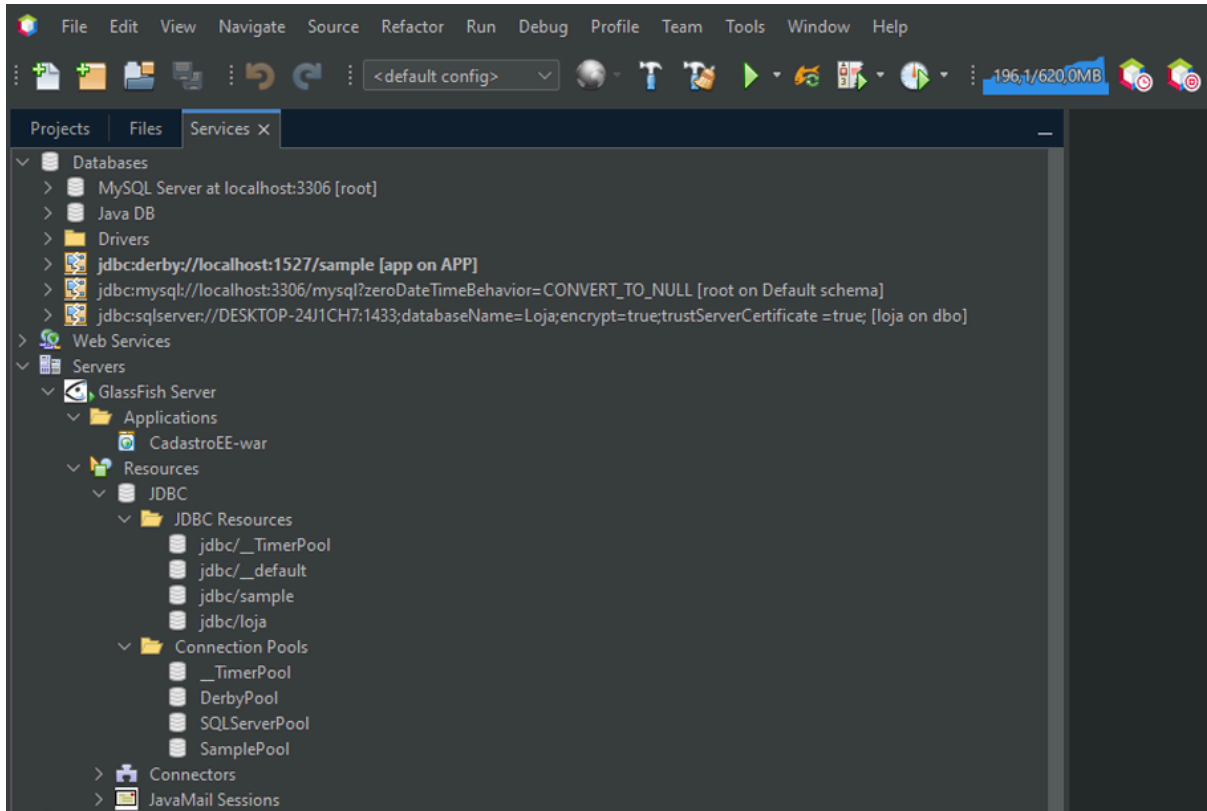
Título da Prática: Vamos Integrar Sistemas
Implementação de sistema cadastral com interface Web,
baseado nas tecnologias de
Servlets, JPA e JEE.

Objetivo 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.



Resultados da execução dos códigos:



Serviços configurados no NetBeans: JDBC, GlassFish, jdbc/Loja, SQLServerPool.



← → ↻ localhost:4848/common/index.jsf

Home About...

User: admin Domain: domain1 Server: localhost

Eclipse GlassFish

Tree

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - CadastroEE-war
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config

General Descriptor

Edit Application

Modify an existing application or module.

Name: CadastroEE-war

Status: ☒

Implicit CDI ☒ Implicit discovery of CDI beans

Location: file:/D:/NetBeansProjects/CadastroEE/CadastroEE-war/build/web/

Deployment Order: 100
A number that determines the loading order of the application at server startup. Lower numbers are

Libraries:

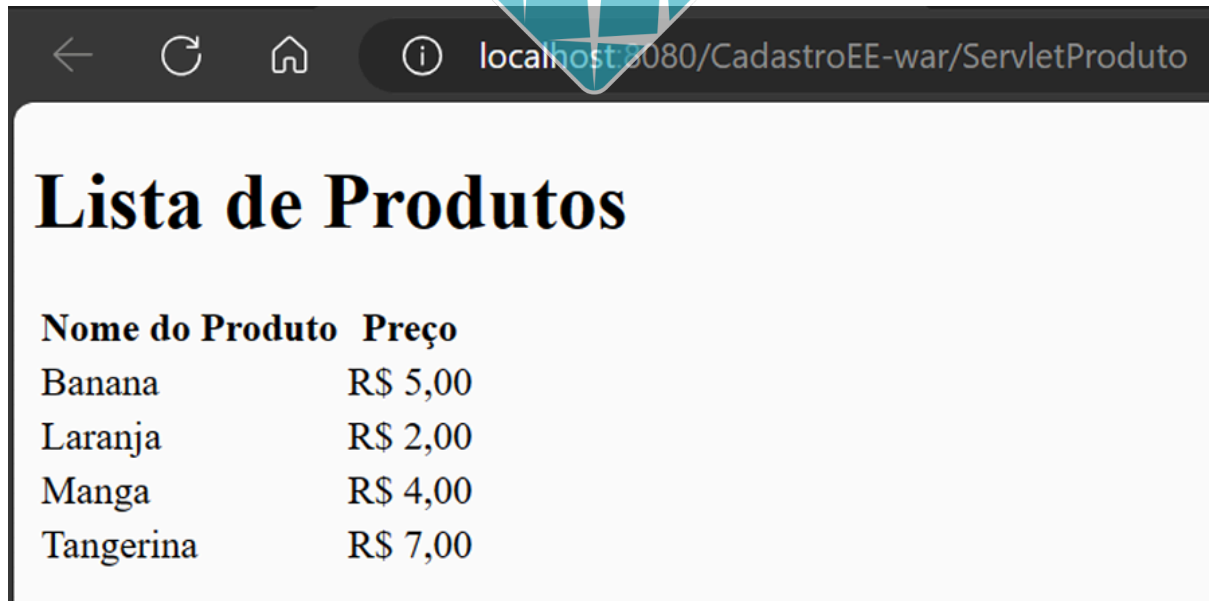
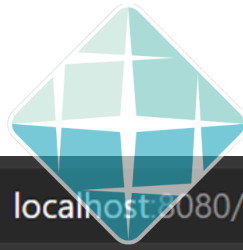
Description:

Modules and Components (11)			
Module Name	Engines	Component Name	Type
CadastroEE-war	[ejb, jpa, web, weld]	-----	-----
CadastroEE-war		PessoaFacade	StatelessSe
CadastroEE-war		ProdutoFacade	StatelessSe

aplicação CadastroEE-war em execução no GlassFish

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence xmlns="http://java.sun.com/xml/ns/persistence"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
5   <persistence-unit name="CadastroEE-ejbPU" transaction-type="JTA">
6     <jta-data-source>jdbc/loja</jta-data-source>
7     <exclude-unlisted-classes>false</exclude-unlisted-classes>
8     <properties/>
9   </persistence-unit>
10 </persistence>
```

Arquivo de configuração persistence.xml



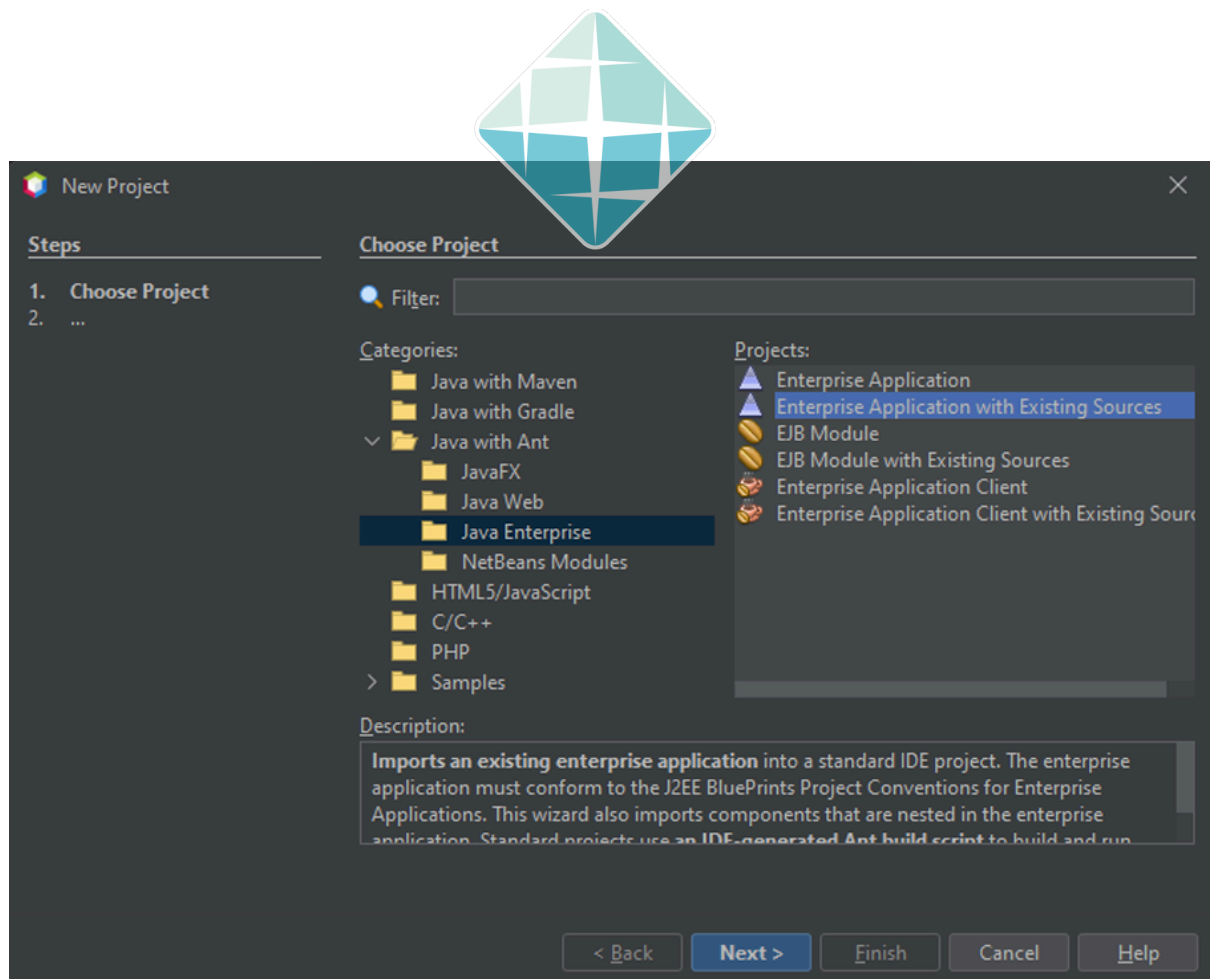
Nome do Produto	Preço
Banana	R\$ 5,00
Laranja	R\$ 2,00
Manga	R\$ 4,00
Tangerina	R\$ 7,00

ServletProduto em execução

Análise e Conclusão:

Como é organizado um projeto corporativo no NetBeans?

Um projeto corporativo (ou Enterprise) no NetBeans segue uma estrutura modular hierárquica, claramente visível ao se criar um novo projeto: escolher uma categoria de gerenciador de pacotes, dentre os mais utilizados na atualidade: Maven, Gradle; em seguida, escolher o tipo de projeto corporativo mais adequado, conforme mostra a figura abaixo:



Criação de um novo projeto corporativo no NetBeans

Projetos corporativos web no NetBeans, mesmo que sejam de diferentes categorias e tipos, seguem um padrão comum de pastas: código-fonte (source), bibliotecas ou dependências (libraries), configurações (configuration ou WEB-INF), páginas web ou servlets (web pages, war). Nesta Missão Prática, foi escolhida a categoria “Java with Ant”, seguida da sub-categoria “Java Enterprise”, com tipo de projeto “Enterprise Application with Existing Sources”, o qual é gerado a partir de um banco de dados existente, como resultado: além da pasta principal Enterprise Application, são criados 2 módulos com diferentes funções: um módulo EJB (Cadastro-ejb) e um módulo de aplicação web (Cadastro-war), gerenciados separadamente e integrados para apresentar a total funcionalidade do sistema. O NetBeans facilita a organização desses módulos, com o uso de



ferramentas e assistentes de configuração, construção e implantação de maneira eficiente.

Qual o papel das tecnologias JPA e EJB na construção de um aplicativo para a plataforma Web no ambiente Java?

Essencialmente, JPA e EJB são ferramentas poderosas para construir aplicações Java robustas e escaláveis.

JPA (Java Persistence API): Simplifica o mapeamento entre objetos Java e bancos de dados relacionais. Com ele, você pode salvar, atualizar e deletar dados de forma direta e eficiente. Pense nele como um tradutor entre o mundo dos objetos e o mundo dos dados relacionais.

EJB (Enterprise JavaBeans): Oferece componentes modulares para desenvolver aplicações empresariais. Eles gerenciam aspectos como transações, segurança e persistência. Isso libera os desenvolvedores para focar na lógica de negócios, enquanto o EJB cuida dos detalhes mais complexos.

Como o NetBeans viabiliza a melhoria de produtividade ao lidar com as tecnologias JPA e EJB?

NetBeans é como ter um superpoder para desenvolvedores que usam JPA e EJB. Ele oferece uma série de recursos integrados que simplificam e aceleram o desenvolvimento, como:

Assistentes de Código: Autocompletar para mapeamento JPA e configurações EJB, reduzindo erros e economizando tempo.



Designers Visuais: Ferramentas gráficas para configurar e mapear entidades JPA e EJB, tornando o processo mais intuitivo.

Gerenciamento de Bancos de Dados: Integração direta com bancos de dados, permitindo testes e ajustes rápidos de consultas e persistência.

Ferramentas de Debugging: Facilita a identificação e resolução de problemas em tempo real, mantendo o fluxo de trabalho contínuo.

Suporte a Testes: Integração com frameworks de teste, permitindo a execução de testes unitários e de integração de forma eficiente.

O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?

O que são Servlets, e como o NetBeans oferece suporte à construção desse tipo de componentes em um projeto Web?

Servlets são classes Java que estendem as capacidades de servidores para hospedar aplicações acessadas via modelo de requisição-resposta, geralmente por meio de HTTP.

Basicamente, são componentes do lado servidor que tratam requisições e geram respostas dinâmicas, frequentemente utilizados para criar aplicações web interativas e dinâmicas.

NetBeans facilita bastante o trabalho com Servlets:

Assistentes e Templates: Possui assistentes para criar novos Servlets com templates predefinidos que agilizam o processo de configuração.



Deploy Rápido: Integração com servidores como Tomcat e GlassFish, permitindo deploys rápidos diretamente do IDE.

Debugging e Monitoramento: Ferramentas de debugging avançadas para rastrear e resolver problemas em tempo real.

Design e Configuração Visual: Interface gráfica para configurar mapeamentos e parâmetros de Servlets sem mergulhar no XML.

Como é feita a comunicação entre os Servlets e os Session Beans do pool de EJBs?

A comunicação entre Servlets e Session Beans em um pool de EJBs é realizada através de um mecanismo chamado JNDI (Java Naming and Directory Interface). Basicamente, os Servlets utilizam JNDI para localizar e referenciar Session Beans disponíveis no pool; após a localização, os Servlets podem invocar métodos nos Session Beans como se realizassem chamadas a métodos em objetos locais. Essa abordagem permite que Servlets, que geralmente gerenciam solicitações de usuários via HTTP, interajam com lógicas de negócios complexas encapsuladas nos Session Beans, assim promovem uma eficiente separação entre camada de apresentação e lógica de negócios.

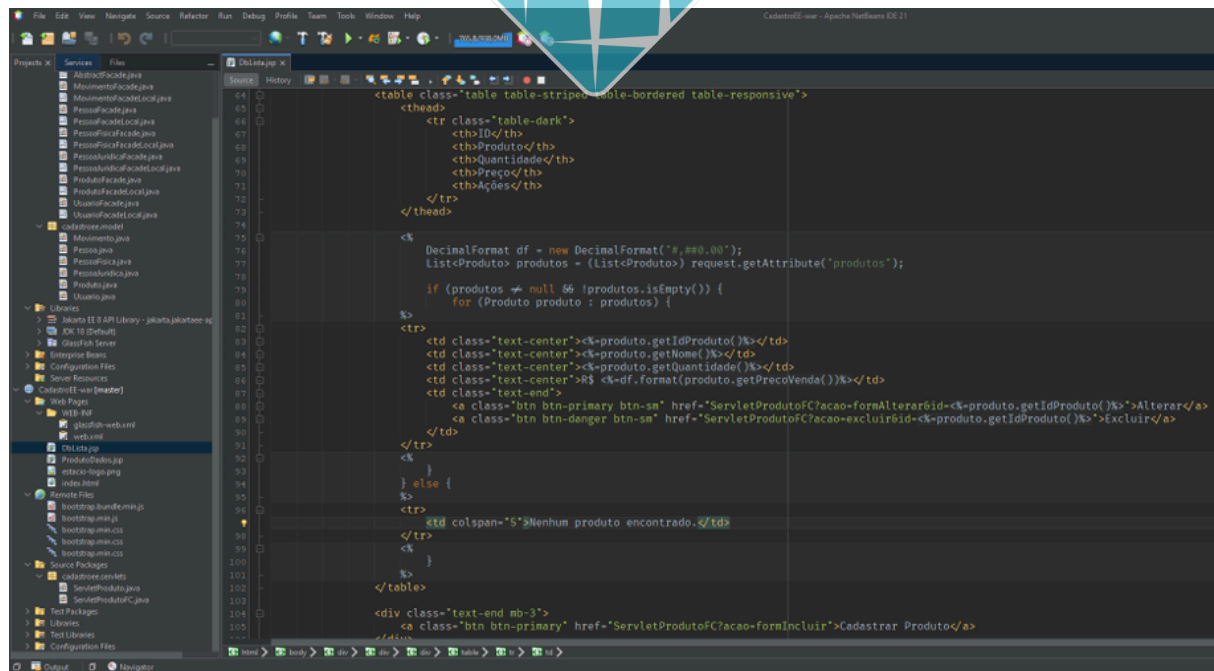


Título da Prática: “2º Procedimento | Interface Cadastral com Servlet e JSP”

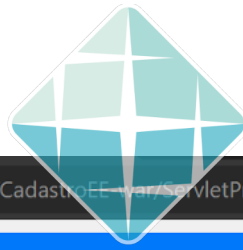
Resultados da execução:

```
16 import jakarta.servlet.http.HttpServletRequest;
17 import jakarta.servlet.http.HttpServletResponse;
18
19 @WebServlet(name = "ServletProdutoFC", urlPatterns = {"/ServletProdutoFC"})
20 public class ServletProdutoFC extends HttpServlet {
21
22     @EJB
23     private ProdutoFacadeLocal facade;
24
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         String acao = request.getParameter("acao");
29         String destino = handleGetAction(acao, request);
30         dispatchRequest(request, response, destino);
31     }
32
33     @Override
34     protected void doPost(HttpServletRequest request, HttpServletResponse response)
35         throws ServletException, IOException {
36         String acao = request.getParameter("acao");
37         acao = acao == null || acao.isEmpty() ? " " : acao;
38         String destino = handlePostAction(acao, request);
39         dispatchRequest(request, response, destino);
40     }
41
42     private String handleGetAction(String acao, HttpServletRequest request) {
43         switch (acao) {
44             case "formIncluir":
45                 return "ProdutoDados.jsp";
46             case "excluir":
47                 return handleExcluir(request);
48             case "formAlterar":
49                 return handleFormAlterar(request);
50             default:
51                 return handleListarProdutos(request);
52         }
53     }
54
55     private String handlePostAction(String acao, HttpServletRequest request) {
56         switch (acao) {
57             case "incluir":
58                 return handleIncluir(request);
59         }
60     }
61 }
```

Trecho do código do ServletProdutoFC.



Trecho do código JSP responsável pela exibição da lista de produtos.



ID	Produto	Quantidade	Preço	Ações	
1	Banana	100	R\$ 5,00	Alterar	Excluir
2	Laranja	500	R\$ 2,00	Alterar	Excluir
3	Manga	800	R\$ 4,00	Alterar	Excluir
4	Tangerina	600	R\$ 7,00	Alterar	Excluir

[Cadastrar Produto](#)

ServletProdutoFC em execução, sem uso de bibliotecas CSS, apenas CSS básico

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** centraliza o tratamento de todas as requisições para um único ponto de entrada, que depois distribui as requisições para os componentes adequados. Aqui está como ele funciona e como é implementado na arquitetura MVC:



1. **Requisição Inicial:** Toda requisição do cliente é direcionada para o Front Controller.
2. **Processamento da Requisição:** O Front Controller examina a requisição e determina qual ação tomar.
3. **Delegação:** Ele delega a lógica de negócios apropriada para um controlador específico ou handler.
4. **Retorno da Resposta:** Depois que o controlador específico processa a lógica, o Front Controller coordena a resposta de volta ao cliente.

Na arquitetura **MVC (Model-View-Controller)**, isso é implementado geralmente com um Servlet ou filtro:

- **Servlet Front Controller:** Captura todas as requisições e determina qual controlador deve lidar com a requisição.
- **Controlador:** Processa a lógica de negócios e interage com o modelo (dados).
- **View:** O controlador seleciona a vista adequada para renderizar a resposta.

Isso resulta em um design mais modular e fácil de manter, com todas as requisições centralizadas e a lógica de roteamento separada da lógica de negócios.

Quais as diferenças e semelhanças entre Servlets e JSPs?

Diferenças:

Natureza: Servlets são classes Java que controlam a lógica de requisições e respostas. JSPs (JavaServer Pages) são páginas HTML com inserções de código Java.



Objetivo: Servlets são mais voltados para a lógica do lado do servidor. JSPs são mais usados para a apresentação, facilitando a criação de conteúdo dinâmico.

Compilação: Servlets são compilados de imediato. JSPs são convertidos em Servlets pelo servidor web quando acessados pela primeira vez.

Semelhanças:

Parte da mesma API: Ambos fazem parte da especificação Java EE.

Interação: Ambos podem interagir e compartilhar dados através de métodos de encaminhamento e redirecionamento.

Finalidade: Ambos lidam com requisições HTTP e geram respostas dinâmicas.

Pense neles como duas faces da mesma moeda: um para a lógica e outro para a apresentação, trabalhando em conjunto para criar aplicações web dinâmicas.

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?

Os redirecionamentos e o uso do método forward têm funções diferentes:

Redirecionamento: Quando você usa `response.sendRedirect()`, o servidor envia uma resposta HTTP ao cliente indicando que ele deve fazer uma nova requisição para uma URL diferente. Isso resulta em uma nova URL no navegador do



usuário e uma nova requisição sendo gerada. Por isso, quaisquer dados enviados na requisição original são perdidos a menos que sejam armazenados de outra forma (como em sessões).

- Forward: O método `forward` do `RequestDispatcher` permite que você encaminhe a requisição e a resposta atuais para outro recurso (como um servlet, JSP ou HTML) no servidor. O cliente não sabe que esse redirecionamento aconteceu, pois a URL do navegador não muda. Os dados na requisição original são mantidos.

Parâmetros e atributos nos objetos `HttpRequest` têm papéis específicos:

- Parâmetros: São dados enviados pelo cliente como parte da URL (em requisições GET) ou no corpo da requisição (em requisições POST). Você pode acessá-los usando `request.getParameter()`.
- Atributos: São dados armazenados no objeto `HttpRequest` durante o processamento da requisição, usando `request.setAttribute()`. Eles são úteis para compartilhar dados entre diferentes componentes de uma aplicação web, como ao usar o método `forward`.

Ambos ajudam a manter a comunicação eficiente e organizada dentro da aplicação



3º Procedimento | Melhorando o Design da Interface

ID	Produto	Quantidade	Preço	Ações
1	Banana	100	R\$ 5,00	<button>Alterar</button> <button>Excluir</button>
2	Laranja	500	R\$ 2,00	<button>Alterar</button> <button>Excluir</button>
3	Manga	800	R\$ 4,00	<button>Alterar</button> <button>Excluir</button>
4	Tangerina	600	R\$ 7,00	<button>Alterar</button> <button>Excluir</button>

Cadastrar Produto

ServletProdutoFC em execução, com uso de Bootstrap

Voltar


Nome

Quantidade

Preço de Venda

Cadastrar

Tela de Cadastro de Produto



localhost:8080/CadastroEE-war/ServletProdutoFrmAlterar?id=...

Alteração de Produto

Voltar

Nome
Tangerina

Quantidade
600

Preço de Venda
7.0

Alterar

Tela de Alteração de Produto já cadastrado.

Análise e Conclusão:

Como o framework Bootstrap é utilizado?

Bootstrap é um framework front-end poderoso e muito utilizado para desenvolvimento web responsivo e moderno. Aqui está como ele é normalmente utilizado:

1. Layouts Responsivos: Utilizando a grade (grid system) do Bootstrap, você pode criar layouts que se adaptam a diferentes tamanhos de tela, desde desktops até smartphones.
2. Componentes UI: Bootstrap oferece uma variedade de componentes prontos, como botões, formulários, modais, navegações, entre outros. Isso agiliza o desenvolvimento e garante uma aparência consistente.



3. Estilização: Inclui classes CSS para estilização rápida e eficiente. Em vez de escrever todo o CSS do zero, você pode aplicar classes Bootstrap para dar estilo aos elementos HTML.
4. JavaScript Plugins: Possui plugins JavaScript embutidos que adicionam funcionalidades interativas, como carrosséis, tooltips e popovers, sem a necessidade de escrever código JavaScript do zero.
5. Personalização: É altamente personalizável. Você pode sobrescrever as classes padrão ou criar seu próprio tema customizado.

Por que o Bootstrap garante a independência estrutural do HTML?

Bootstrap garante a independência estrutural do HTML ao fornecer classes CSS predefinidas e componentes que se separam da estrutura real do HTML. Isso significa que, ao aplicar as classes do Bootstrap, você está adicionando estilo e comportamento sem precisar alterar a estrutura básica dos seus elementos HTML.

Basicamente, você pode manter seu HTML limpo e semanticamente correto enquanto utiliza as classes do Bootstrap para adicionar estilos sofisticados e funcionalidades. Essa abordagem modular facilita a manutenção do código, promove a reutilização e permite alterações visuais rápidas sem mexer na estrutura fundamental do HTML.

Usando Bootstrap, você pode mudar a aparência de uma página inteira trocando algumas classes, sem precisar reescrever o HTML.



Qual a relação entre o Bootstrap e a responsividade da página?

Bootstrap e responsividade andam de mãos dadas. O framework foi projetado com uma grade flexível que se adapta a diferentes tamanhos de tela, tornando mais fácil criar layouts que funcionem bem em desktops, tablets e smartphones.

- **Grade Flexível:** A estrutura de colunas permite que os elementos se reorganizem conforme o tamanho da tela muda.
- **Media Queries:** Usam regras CSS para aplicar estilos diferentes com base na largura da viewport.
- **Componentes Responsivos:** Muitos componentes, como menus de navegação e carrosséis, são adaptáveis e automaticamente ajustam-se às diferentes telas.

Bootstrap simplifica a criação de páginas web modernas que oferecem uma experiência de usuário consistente, não importa o dispositivo