



Desenvolvimento Full Stack

Aluno: Victor de A. Costa

Missão Prática | Nível 4 | Mundo 3 - 2024.3
VAMOS MANTER AS INFORMAÇÕES!

Implementação de sistema cadastral com interface Web, baseado nas tecnologias de Servlets, JPA e JEE.

1º Procedimento | Alimentando a Base

---> CÓDIGOS UTILIZADOS

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>
</web-app>
```

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 jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import cadastroee.controller.ProdutoFacadeLocal;
import cadastroee.model.Produto;

/**
 *
 * @author victorcosta
 */
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 Produto</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>

}

```

---> RESULTADO

The screenshot shows a web browser window at localhost:8080/CadastroEE-war/ServletProduto. Below the browser is a SQL Server Management Studio (SSMS) interface.

Lista de Produtos

- nome produto
- nome produto
- Camiseta Azul
- Calça jeans
- Tênis Esportivo

SQL Query Results:

```

SELECT TOP (1000) [idProduto]
      ,[nome]
      ,[quantidade]
      ,[precoVenda]
  FROM [Loja2].[dbo].[Produto]

```

	idProduto	nome	quantidade	precoVenda
1	1	nome produto	50	5000
2	2	nome produto	50	5000
3	3	Camiseta Azul	564	7000
4	4	Calça jeans	15	400
5	5	Tênis Esportivo	55	4000

---> ANÁLISE E CONCLUSÃO

A. Como é organizado um projeto corporativo no NetBeans?

A estrutura básica de um projeto corporativo no NetBeans geralmente inclui:

Páginas JSP (JavaServer Pages) / Classes Java / Arquivos CSS e JavaScript / Configurações do aplicativo (ex: web.xml)

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

As tecnologias JPA (Java Persistence API) e EJB (Enterprise JavaBeans), facilitam o gerenciamento de dados e a lógica de negócios, com suas contrubuições. Da JPA, pode-se destacar a Persistência de Dados, a Flexibilidade, e a Independência de Banco de Dados, já da EJB pode-se destacar a Lógica de Negócios. Transações, maior preocupação com a Segurança, e Escalabilidade e Gerenciamento de Recursos.

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

O NetBeans é uma IDE poderosa que facilita o trabalho com JPA e EJB, melhorando a produtividade dos desenvolvedores de várias maneiras: Assistentes e Templates, Integração com Servidores de Aplicação, Ferramentas de Persistência, Integração e Testes, Assistentes de Código e Refatoração, Ferramentas Gráficas.

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

Servlets são componentes do lado do servidor que tratam requisições e respostas HTTP em uma aplicação web Java. Eles são utilizados para criar aplicações web dinâmicas, processando a entrada dos usuários (como formulários) e gerando a resposta (como páginas HTML). Basicamente, os servlets atuam como intermediários entre o cliente (navegador web) e o servidor (aplicação web). O NetBeans oferece um ambiente robusto para o desenvolvimento de servlets, facilitando várias etapas do processo, como Criação de Projetos Web, Criação de Servlets, Deploy e Debug, Editor de Código, Visualização e Testes, Configuração e Descrição.

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

A comunicação entre Servlets e Session Beans (EJBs) é uma parte fundamental do desenvolvimento de aplicações corporativas em Java. Esta comunicação permite que um servlet (que trata requisições HTTP) utilize a lógica de negócios encapsulada nos EJBs. Ela pode ser feita através de Injeção de Dependência (Utilizando a anotação @EJB, você pode injetar uma instância do Session Bean no servlet), ou através de uma busca JNDI.