Campus: Asa Norte

Curso: Desenvolvimento Full Stack

Disciplina: Vamos Integrar Sistemas

Matrícula: 2023.09.96862-2

Semestre Letivo: 3º Semestre

Integrantes: André Luis Soares de Oliveira

Implementação de Sistema Cadastral com Interface Web utilizando Servlets, JPA e JEE

Objetivo da Prática

Desenvolver uma aplicação Java Web com funcionalidades de cadastro, aplicando persistência com JPA, controle de negócios com EJB e interface gráfica com JSP e Servlets. Melhorar o design com Bootstrap e trabalhar com o banco de dados SQL Server.

Códigos Solicitados

Configuração de conexão:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;


public class DatabaseConnection {

   private static final String URL = "dbc:sqlserver://

localhost:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;";

   private static final String USER = "loja";

   private static final String PASSWORD = "loja";


   public static Connection getConnection() throws SQLException {

      return DriverManager.getConnection(URL, USER, PASSWORD);

   }

}
```

Configuração do persistence.xml:

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

<persistence xmlns="https://jakarta.ee/xml/ns/persistence"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="https://jakarta.ee/xml/ns/persistence
```

```
                    https://jakarta.ee/xml/ns/persistence/persistence_3_0.xsd"
        version="3.0">
  <persistence-unit name="LojaPU" transaction-type="JTA">
    <jta-data-source>jdbc/LojaDS</jta-data-source>

    <properties>
      <!-- Configuração de Dialeto do SQL Server -->
      <property name="jakarta.persistence.jdbc.url"
value="jdbc:sqlserver://localhost:1433;databaseName=loja"/>
      <property name="jakarta.persistence.jdbc.user" value="seu_usuario"/>
      <property name="jakarta.persistence.jdbc.password" value="sua_senha"/>
      <property name="jakarta.persistence.jdbc.driver"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
      <property name="hibernate.dialect"
value="org.hibernate.dialect.SQLServerDialect"/>

      <!-- Configurações adicionais (opcionais) -->
      <property name="jakarta.persistence.schema-generation.database.action"
value="create"/>
      <property name="hibernate.show_sql" value="true"/>
      <property name="hibernate.format_sql" value="true"/>
    </properties>
  </persistence-unit>
</persistence>
```

Análise e Conclusão

Como é organizado um projeto corporativo no NetBeans:

No **NetBeans**, um projeto corporativo para a plataforma Java EE segue uma estrutura modular, onde diferentes camadas de uma aplicação ficam organizadas em pacotes distintos, como **modelo**, **controle** e **visão**. A organização padrão envolve:

- **Módulo de Persistência**: onde são configurados os mapeamentos de entidades JPA e o arquivo `persistence.xml` que define o banco de dados e as configurações de persistência.
- **Módulo de Negócio**: aqui residem os **Session Beans (EJBs)**, que contêm as regras de negócio e são responsáveis por operações que envolvem lógica de negócios, transações e segurança.
- **Camada Web**: composta por **Servlets** e **JSPs** para gerenciar as requisições HTTP e a interface do usuário.
- **Dependências e Bibliotecas**: onde são gerenciados os recursos externos, como drivers de banco de dados, bibliotecas de logging, entre outros.

Essas divisões ajudam a organizar um projeto corporativo, facilitando a manutenção, escalabilidade e modularidade do código.

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

JPA (Java Persistence API): JPA é uma especificação para o mapeamento de objetos relacionais (ORM). Seu papel é simplificar a persistência de dados, permitindo que classes Java sejam mapeadas para tabelas de um banco de dados relacional. A JPA permite manipular dados de forma mais direta e eficiente, usando entidades e consultas sem que o desenvolvedor precise lidar diretamente com SQL em muitos casos.

EJB (Enterprise JavaBeans): EJBs são componentes de servidor que encapsulam a lógica de negócios e fornecem funcionalidades de transações, segurança e gerenciamento de recursos. Eles são ideais para manipular operações mais complexas que envolvem várias etapas transacionais. Dentro do contexto de uma aplicação web, EJBs oferecem o backend de negócios, recebendo dados dos servlets, processando-os e interagindo com a camada de persistência (JPA) para armazenar ou buscar informações do banco de dados.

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

O **NetBeans** oferece uma série de ferramentas e assistentes para facilitar o uso de JPA e EJB:

- **Assistentes de Código**: o NetBeans possui geradores de código para criar classes de entidade JPA a partir das tabelas do banco de dados, poupando o desenvolvedor de escrever mapeamentos manualmente.

- **Configuração Automática do `persistence.xml`**: ao configurar um projeto JPA, o NetBeans ajuda a gerar e configurar o arquivo `persistence.xml` com as definições de unidade de persistência, o que reduz erros e simplifica o processo de configuração.
- **Facilidade na Criação de EJBs**: o NetBeans permite criar EJBs rapidamente por meio de modelos prontos, configurando automaticamente as anotações necessárias e ajudando a configurar a injeção de dependências.
- **Testes e Debugging Integrados**: o ambiente NetBeans oferece uma integração completa com o GlassFish e outros servidores de aplicação, permitindo que o desenvolvedor depure, teste e monitore EJBs e operações JPA diretamente, o que melhora a produtividade e reduz o tempo de desenvolvimento.

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 permitem que aplicações Java EE respondam a requisições HTTP. Eles são componentes fundamentais para aplicações Web, processando requisições (como `GET` e `POST`), manipulando dados, e retornando respostas ao cliente, geralmente redirecionando para páginas JSP ou outras visualizações.

O **NetBeans** facilita a criação e gerenciamento de servlets com:

- **Modelos e Assistentes**: NetBeans oferece modelos para criar Servlets rapidamente, incluindo métodos como `doGet` e `doPost`, pré-configurados para receber e responder a requisições HTTP.
- **Mapeamento de URLs**: ao criar um Servlet, o NetBeans permite definir o mapeamento de URL direto, facilitando o roteamento das requisições.
- **Debugging e Testes**: o NetBeans fornece suporte direto para debugging de servlets, permitindo que o desenvolvedor insira breakpoints e acompanhe a execução de requisições HTTP.

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

A comunicação entre **Servlets** e **Session Beans** é feita por meio da **Injeção de Dependência**. No Servlet, um Session Bean é injetado usando a anotação `@EJB`. Assim, o servlet pode chamar métodos diretamente do EJB para realizar operações de negócios.

Link Github: https://github.com/andreluissdo/Missao-Pratica-4.git