1. O que é JSP? Cite suas vantagens e desvantagens.

JSP (Java Server Pages) é uma tecnologia Java para criar páginas web dinâmicas. É semelhante ao PHP e ASP, mas permite que o código Java seja embutido diretamente em páginas HTML.

Vantagens:

Integração com Java: Fácil integração com aplicações Java e APIs.

Separação de lógica e apresentação: Ajuda na organização do código, separando a interface do usuário da lógica de negócios.

Compatibilidade com Servlets: Pode ser combinado com Servlets para maior funcionalidade.

Facilidade de manutenção: Alterações no design podem ser feitas sem mexer no código Java.

Desvantagens:

Complexidade em projetos grandes: Misturar Java com HTML pode gerar códigos difíceis de manter.

Performance: Em alguns casos, é mais lenta do que outras tecnologias, como frameworks modernos (ex.: Spring Boot).

Obsolescência: JSP tem perdido popularidade em relação a frameworks mais modernos.

---

2. O que é um Servlet em Java e qual é sua finalidade principal?

Um Servlet é um componente Java usado no lado do servidor para lidar com requisições HTTP e gerar respostas dinâmicas, como HTML, JSON ou XML. Ele é executado dentro de um servidor web ou aplicativo, como Tomcat.

Finalidade principal:

Processar requisições do cliente (GET, POST, etc.).

Gerar respostas dinâmicas, geralmente em páginas HTML ou dados JSON/XML.

Integrar a lógica de negócios com o front-end web.

---

3. Como você criaria uma conexão JDBC com um banco de dados MySQL em Java?

Passos:

1. Adicione o driver JDBC do MySQL ao projeto.

2. Configure os dados de conexão (URL, usuário, senha).

3. Use DriverManager para criar a conexão.

Código de exemplo:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConexaoMySQL {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/
meubanco";
        String user = "root";
        String password = "senha";

        try (Connection conn =
DriverManager.getConnection(url, user,
password)) {
            System.out.println("Conexão bem-
sucedida!");
        } catch (SQLException e) {
            e.printStackTrace();
```

```
        }
    }
}
```

---

4. Inserir uma dentro de uma no HTML:

Problema:

A tag é um elemento inline, enquanto é um elemento block. Inserir uma dentro de uma viola as regras de semântica e pode causar problemas na renderização e estilização.

Implicações:

Estrutura de layout inconsistente.

Problemas de acessibilidade e semântica.

Possíveis conflitos com CSS e scripts que

dependem de hierarquia válida.

---

5. Em uma arquitetura orientada a objetos, qual é o meio principal de comunicação entre componentes?

O meio principal de comunicação é através de métodos públicos e mensagens trocadas entre objetos. Os componentes expõem interfaces para interagir com outros objetos, respeitando o encapsulamento.

---

6. Principal característica das arquiteturas monolíticas:

A principal característica é a alta interdependência dos componentes. Isso dificulta a manutenção e evolução, pois qualquer mudança em um componente pode

afetar todo o sistema.

---

7. Função da camada de Modelo no padrão MVC:

A camada de Modelo:

Representa os dados e a lógica de negócios.

Garante a manipulação e persistência dos dados (geralmente com um banco de dados).

Atua como intermediária entre as camadas de Controle (Controller) e de Visualização (View).

---

8. Impacto do acoplamento em microsserviços:

Alto acoplamento dificulta a manutenção, evolução e escalabilidade dos microsserviços, além de reduzir a independência dos

componentes. Idealmente, cada microsserviço deve ser autônomo e desacoplado.

---

9. Formulário HTML e envio para um banco PostgreSQL

Formulário HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <title>Formulário</title>
</head>
<body>
    <form action="salvarUsuario" method="post">
        <label for="nome">Nome:</label>
        <input type="text" id="nome" name="nome"
required>
        <br>
        <label for="email">Email:</label>
```

```
        <input type="email" id="email" name="email"
required>
        <br>
        <label for="telefone">Telefone:</label>
        <input type="tel" id="telefone"
name="telefone" required>
        <br>
        <button type="submit">Enviar</button>
    </form>
</body>
</html>
```

Servlet Java para processar o formulário:

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class SalvarUsuario extends HttpServlet {
    protected void doPost(HttpServletRequest
request, HttpServletResponse response)
            throws ServletException, IOException {
        String nome =
request.getParameter("nome");
        String email =
request.getParameter("email");
        String telefone =
request.getParameter("telefone");

        String url = "jdbc:postgresql://
localhost:5432/meubanco";
        String user = "postgres";
        String password = "senha";

        try (Connection conn =
DriverManager.getConnection(url, user,
password)) {
            String sql = "INSERT INTO usuarios
(nome, email, telefone) VALUES (?, ?, ?)";
            PreparedStatement stmt =
conn.prepareStatement(sql);
            stmt.setString(1, nome);
            stmt.setString(2, email);
```

```
            stmt.setString(3, telefone);
            stmt.executeUpdate();
            response.getWriter().println("Usuário
salvo com sucesso!");
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("Erro ao
salvar o usuário.");
        }
    }
}
```

Observação:

Certifique-se de que a tabela usuarios exista no banco com os campos nome, email, telefone.

Configure corretamente o servidor web (como Tomcat) para processar o servlet.

10. O que é um driver JDBC? Existem diferentes tipos?

Driver JDBC: Componente que permite que aplicativos Java se conectem a um banco de dados.

Tipos:

1. Driver JDBC-ODBC Bridge.

2. Driver nativo (específico para banco).

3. Driver de protocolo de rede.

4. Driver puro Java (mais usado).

---

11. O que é JDBC e sua principal função?

JDBC (Java Database Connectivity) é uma API que permite interagir com bancos de dados em Java.
Função principal: Executar operações como conexão, consultas e atualizações.

---

12. Passos básicos para estabelecer uma conexão JDBC:

1. Carregar o driver.

2. Estabelecer a conexão via DriverManager.

3. Criar e executar consultas SQL.

4. Fechar conexão.

---

13. Propósito da classe DriverManager no JDBC:

Gerencia conexões com bancos de dados,

identificando o driver correto com base na URL fornecida.

---

14. Executar uma consulta SQL com JDBC:

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT *
FROM usuarios");
while (rs.next()) {
    System.out.println(rs.getString("nome"));
}
```

---

15. Papel principal do JSP:

Renderizar conteúdo dinâmico, combinando HTML com dados processados no servidor.

---

16. Função de um Servlet em relação às requisições:

Receber, processar e responder a requisições HTTP, conectando o cliente ao back-end.

---

17. Geração de uma página HTML com JSP:

O servidor compila o JSP em um Servlet, que processa dados e gera o HTML para o cliente.

---

18. Classe Controlador em uma aplicação Java:

Gerencia o fluxo de dados entre o modelo e a visão, implementando a lógica de negócios.

---

19. Função da classe Entidade:

Representar objetos do modelo de negócio, geralmente mapeados para tabelas do banco de dados.

20. O que é um CRUD? Como usar SGBD para realizar essas operações?

CRUD significa:

Create: Criar novos registros.

Read: Ler dados existentes.

Update: Atualizar registros existentes.

Delete: Excluir registros.

Como usar SGBD para realizar CRUD:

1. Create: Usar INSERT INTO para adicionar dados.
Exemplo:

INSERT INTO usuarios (nome, email, telefone)
VALUES ('João', 'joao@example.com',
'123456789');

2. Read: Usar SELECT para consultar dados.
Exemplo:

SELECT * FROM usuarios WHERE email =
'joao@example.com';

3. Update: Usar UPDATE para modificar dados.
Exemplo:

UPDATE usuarios SET telefone = '987654321'
WHERE email = 'joao@example.com';

4. Delete: Usar DELETE para remover dados.
Exemplo:

DELETE FROM usuarios WHERE email =
'joao@example.com';

No JDBC, você executa essas operações usando PreparedStatement e SQL.