JPA - PRÁTICA.

LISTA DE EXERCÍCIO 01

(Persistência a Dados, Tipos de Dados, XML, JSON e JDBC).

Grupo:

- Jennefer Sousa
- Julliana Rodrigues
- Luanna Fernandes
- Maria Angela
- Gustavo Henriques

Resposta:

1- A persistência de dados é a capacidade de armazenar informações de forma duradoura, permitindo sua recuperação após o encerramento de um programa ou desligamento do computador. Isso é essencial para garantir a disponibilidade e integridade dos dados ao longo do tempo. Métodos comuns incluem bancos de dados, arquivos, armazenamento em nuvem e sistemas de arquivos distribuídos, dependendo dos requisitos específicos do sistema. A escolha da abordagem depende da quantidade de dados, velocidade de acesso e outros fatores, visando garantir a segurança e confiabilidade dos dados.

2 -Objetos transientes são temporários, existem apenas na memória enquanto um programa está em execução e não são armazenados permanentemente. Por outro lado,

objetos persistentes têm uma representação permanente em algum meio de armazenamento, como um banco de dados ou um arquivo, permitindo que suas informações sobrevivam a diferentes sessões ou reinicializações do programa. A diferença principal está na durabilidade das informações e na capacidade de persistência além da execução do programa.

3 - A principal diferença entre um Banco de Dados Relacional e um Banco de Dados Orientado a Objetos está na forma como organizam e representam os dados:

Banco de Dados Relacional usa tabelas com linhas e colunas para armazenar dados relacionais, sendo adequado para sistemas com estruturas de dados bem definidas e relacionamentos complexos.

Banco de Dados Orientado a Objetos armazena dados como objetos, seguindo uma abordagem mais próxima da programação orientada a objetos, o que é útil em aplicações onde os objetos são diretamente relacionados aos dados.

A escolha entre eles depende das necessidades específicas do projeto e da estrutura dos dados a serem gerenciados.

4- O mapeamento objeto-relacional (ORM) tem como objetivo facilitar a interação entre sistemas de gerenciamento de bancos de dados relacionais e linguagens de programação orientadas a objetos. Ele abstrai a camada de dados, elimina a necessidade de lidar diretamente com SQL, aumenta a produtividade, simplifica a manutenção, ajuda a gerenciar relacionamentos e melhora a segurança. Além disso, torna o código mais portátil e independente do sistema de gerenciamento de banco de dados, facilitando a migração entre diferentes sistemas. Em suma, o ORM simplifica o desenvolvimento de aplicativos ao criar uma ponte entre o modelo de dados relacional e o modelo orientado a objetos da aplicação.

5-a) Dados estruturados: São dados organizados em um formato predefinido e consistente, geralmente armazenados em tabelas de banco de dados com colunas bem definidas. Exemplos incluem bancos de dados relacionais e planilhas.

b) Dados semiestruturados: São dados que não seguem uma estrutura rígida, mas ainda possuem algum grau de organização. Eles frequentemente contêm metadados ou são representados em formatos como JSON, XML ou documentos HTML, onde a estrutura pode variar entre os registros.

c) Dados não estruturados: São dados que não possuem uma organização predefinida e não podem ser facilmente armazenados em tabelas ou estruturas semelhantes. Isso inclui texto livre, imagens, áudio e vídeo, onde a estrutura é ambígua e requer processamento avançado para extrair informações significativas.

6- A biblioteca responsável pela extração e captura de dados disponíveis em arquivos HTML ou XML, frequentemente utilizada em Python, é chamada de Beautiful Soup. Ela permite analisar e manipular documentos HTML e XML de forma eficiente, criando uma árvore de análise que facilita a extração de informações específicas, como textos, links e tabelas, tornando-a valiosa para tarefas de web scraping e análise de dados estruturados em formato de marcação.

7- (A):

- **Sintaxe inicial na primeira linha do arquivo.xml** <?xml version="1.0" encoding="UTF-8"?>

- Os dados são organizados em formato hierárquico ou tabular? Os dados em um documento XML são organizados em formato hierárquico. XML é uma linguagem de marcação que utiliza tags para definir elementos, e esses elementos podem conter outros elementos, criando uma estrutura em árvore hierárquica. Isso permite representar relacionamentos complexos entre dados, o que é especialmente útil para modelar dados com estruturas aninhadas.

- Elementos: A representação mais comum em XML é por meio de elementos. Um elemento é definido por uma tag de abertura (<tag>) e uma tag de fechamento (</tag>) que envolve o conteúdo associado ao elemento.

  ```
  <person>
    <name>John Doe</name>
    <age>30</age>
  </person>
  ```

- Atributos: Além dos elementos, XML permite atributos que fornecem informações adicionais sobre um elemento. Os atributos são definidos dentro da tag de abertura e têm um nome e um valor.

  ```
  <book title="Harry Potter" author="J.K. Rowling" />
  ```

  A sintaxe inicial com a declaração XML é necessária para informar ao interpretador como processar o documento XML, incluindo a versão e a codificação de caracteres.
  A organização hierárquica de dados em formato de árvore permite representar estruturas complexas e relacionamentos entre dados, tornando o XML adequado para modelar informações de maneira flexível.
  As duas formas de representação, elementos e atributos, oferecem flexibilidade para escolher como estruturar os dados de acordo com as necessidades específicas do documento XML, sendo úteis em diferentes contextos de uso.

  8 - código:

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

```
<produto>
   <categoria>Eletrônicos</categoria>
   <nome>Smartphone Modelo A</nome>
   <caracteristicas>
      <cor>Preto</cor>
      <tamanho>Tela de 6 polegadas</tamanho>
      <memoria>128 GB</memoria>
      <camera>20 MP</camera>
      <preco>799.99</preco>
   </caracteristicas>
</produto>
<produto>
   <categoria>Eletrônicos</categoria>
   <nome>Tablet Modelo B</nome>
   <caracteristicas>
      <cor>Branco</cor>
      <tamanho>Tela de 10 polegadas</tamanho>
      <memoria>64 GB</memoria>
      <camera>Sem câmera</camera>
      <preco>299.99</preco>
   </caracteristicas>
</produto>

<!-- Móveis -->
<produto>
   <categoria>Móveis</categoria>
   <nome>Sofá de Couro</nome>
   <caracteristicas>
      <cor>Marrom</cor>
      <material>Couro genuíno</material>
      <estilo>Contemporâneo</estilo>
      <tamanho>3 lugares</tamanho>
      <preco>1499.99</preco>
   </caracteristicas>
</produto>
<produto>
   <categoria>Móveis</categoria>
   <nome>Mesa de Jantar</nome>
   <caracteristicas>
      <cor>Preto</cor>
      <material>MDF</material>
      <estilo>Clássico</estilo>
      <tamanho>8 pessoas</tamanho>
      <preco>499.99</preco>
   </caracteristicas>
</produto>
```

```
<!-- Imóveis -->
<produto>
  <categoria>Imóveis</categoria>
  <nome>Apartamento no Centro</nome>
  <caracteristicas>
    <localizacao>Centro da cidade</localizacao>
    <quartos>3 quartos</quartos>
    <banheiros>2 banheiros</banheiros>
    <area>120 metros quadrados</area>
    <preco>350000.00</preco>
  </caracteristicas>
</produto>
<produto>
  <categoria>Imóveis</categoria>
  <nome>Casa na Praia</nome>
  <caracteristicas>
    <localizacao>Beira-mar</localizacao>
    <quartos>4 quartos</quartos>
    <banheiros>3 banheiros</banheiros>
    <area>200 metros quadrados</area>
    <preco>750000.00</preco>
  </caracteristicas>
</produto>

<!-- Roupas -->
<produto>
  <categoria>Roupas</categoria>
  <nome>Camiseta Casual</nome>
  <caracteristicas>
    <cor>Azul</cor>
    <tamanho>M</tamanho>
    <material>Algodão</material>
    <estilo>Camiseta básica</estilo>
    <preco>19.99</preco>
  </caracteristicas>
</produto>
<produto>
  <categoria>Roupas</categoria>
  <nome>Calça Jeans Slim</nome>
  <caracteristicas>
    <cor>Jeans Azul</cor>
    <tamanho>32</tamanho>
    <material>Jeans</material>
    <estilo>Calça slim</estilo>
    <preco>49.99</preco>
  </caracteristicas>
</produto>
```

</produtos>

9 - Um documento JSON (JavaScript Object Notation) é um formato leve e independente de plataforma usado para representar dados estruturados. Ele utiliza uma sintaxe simples de pares chave-valor, suporta tipos de dados variados e permite representar informações hierarquicamente. JSON é amplamente utilizado em aplicativos web, APIs e outros contextos de programação devido à sua simplicidade, eficiência e facilidade de leitura por humanos.

10 –
Serialização (JSON.stringify) é o processo de converter dados de um objeto ou estrutura de dados em formato JSON, representado como uma sequência de caracteres. Desserialização (JSON.parse) é o processo inverso, convertendo uma string JSON de volta em um objeto ou estrutura de dados no formato nativo da linguagem. Esses processos são cruciais para a comunicação e armazenamento de dados estruturados em sistemas diferentes.

11 - Codigo:

```
{
 "servico": "Loja Online",
 "descricao": "Uma plataforma de comércio eletrônico para venda de produtos variados.",
 "fundacao": 2010,
 "categorias": ["Eletrônicos", "Roupas", "Móveis", "Alimentos"],
 "avaliacao": {
  "nota": 4.5,
  "quantidadeAvaliacoes": 5000
 },
 "produtos": [
  {
   "nome": "Smartphone Modelo X",
   "categoria": "Eletrônicos",
   "preco": 799.99,
   "estoque": 100
  },
  {
   "nome": "Sapatos de Couro",
   "categoria": "Roupas",
   "preco": 99.99,
   "estoque": 200
  },
  {
   "nome": "Sofá Reclinável",
   "categoria": "Móveis",
   "preco": 699.99,
   "estoque": 50
  },
```

```
    {
      "nome": "Alimentos Orgânicos",
      "categoria": "Alimentos",
      "preco": 49.99,
      "estoque": 300
    }
  ],
  "endereco": {
    "rua": "Avenida Principal",
    "cidade": "Cidade Virtual",
    "pais": "Mundo Digital"
  }
}
```

12-

- **Sintaxe:** JSON usa pares chave-valor simples, enquanto XML utiliza marcações (tags) para envolver os dados.

- **Legibilidade:** JSON é mais legível para humanos devido à sua sintaxe direta, enquanto XML pode ser mais verboso.
- **Flexibilidade:** JSON é menos flexível em termos de tipos de dados, enquanto XML permite definir tipos personalizados.
- **Hierarquia de Dados:** Ambos suportam hierarquia, mas XML é projetado especificamente para isso.
- **Tamanho do Arquivo:** JSON geralmente resulta em arquivos menores.
- **Uso em Web Services:** JSON é mais comum em serviços web modernos.
- **Suporte de Linguagens:** Ambos são amplamente suportados, mas JSON é mais fácil de usar em muitas linguagens.
- **Validação:** XML oferece recursos de validação mais robustos.
- A escolha entre eles depende das necessidades do projeto, legibilidade, tamanho de arquivo, tipos de dados e ecossistema de linguagem de programação. Ambos são amplamente usados em diferentes contextos.

13 - O JDBC (Java Database Connectivity) é uma API em Java que permite que aplicativos Java se conectem e interajam com Sistemas de Gerenciamento de Banco de Dados (SGDBs). Ele é usado para estabelecer conexões, executar consultas SQL, gerenciar transações e acessar dados armazenados em bancos de dados. O JDBC desempenha um papel crucial na integração de aplicativos Java com bancos de dados, sendo amplamente usado em aplicativos empresariais, web e desktop para acessar e manipular dados de forma eficiente e segura.

14 - Durante a implementação do JDBC (Java Database Connectivity), os principais componentes incluem o driver JDBC (que atua como uma ponte entre o aplicativo Java e o banco de dados), a URL de conexão (para especificar os detalhes da conexão), interfaces como Connection (para gerenciar conexões e transações), objetos Statement (para executar consultas SQL) e ResultSet (para armazenar resultados de consultas), além de tratamento de exceções e configurações específicas do banco de dados. Esses componentes são essenciais para permitir que aplicativos Java se conectem e interajam com Sistemas de Gerenciamento de Banco de Dados de forma eficiente.

15 - Dependência de drivers JDBC para bancos de dados específicos.

Segurança, evitando injeção de SQL e protegendo contra vulnerabilidades.

Desempenho, otimizando consultas e usando índices quando necessário.

Gerenciamento de conexões, para evitar vazamentos e considerar pooling de conexões.

Compatibilidade do JDBC com o banco de dados utilizado.

Tratamento adequado de exceções, especialmente SQLException.

Manutenção e atualização das dependências, como drivers de banco de dados.

Planejamento para escalabilidade em sistemas de alto tráfego.

Considerar esses pontos ajudará a garantir uma utilização eficaz e segura do JDBC em sistemas modernos.