*Aluna: Isabelle Queiroz Gomes de Assis*

*Turma: Informática 2° ano – 2M*

*Programação Orientada a Objetos – Professor Max Miller*

***Padrões de Representação de Dados***

# XML

XML vem de eXtensible Markup Language (em tradução livre, Linguagem Extensível de Marcação) e é uma ‘parente’ próxima do HTML. Feita para armazenar e transportar dados ao mesmo tempo que consegue ser facilmente compreendida tanto por máquinas quanto por humanos, o núcleo do XML consiste exclusivamente em fornecer uma espécie de padrão de listagem para os dados, enquanto HTML foca apenas em design.

Os dados em XML são conhecidos como auto descritivos, o que significa que a estrutura dos dados é incorporada aos mesmos. Portanto, não há necessidade de pré-construir o esqueleto de armazenamento; ele é entendido de modo dinâmico no próprio XML.

As marcações do XML não são predefinidas, ou seja, elas são criadas a partir dos dados que estão sendo armazenados. O XML armazena os dados em uma forma simples de texto, provendo uma maneira independente de organizar essas informações. O bloco básico do XML é um elemento que é definido por tags. Cada um desses elementos tem uma marca de abertura e de fechamento, e todos estão contidos dentro de um ‘elemento-raiz’.

Uma imagem contendo interior

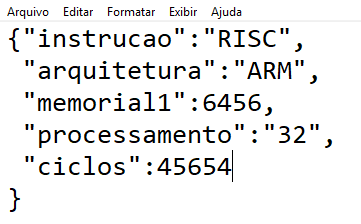
Descrição gerada com alta confiança

# JSON

JSON (JavaScript Object Notation) é um tipo de sintaxe para armazenar informações em formato textual de maneira organizada e de fácil acesso, usada como uma alternativa mais compacta do XML.

Um JSON é constituído por uma coleção de pares de nomes e valores; esse arranjo de ‘objetos’ é comumente representado como uma array, vetor, lista ou sequência. Com sua estrutura simples e a ausência de notações complicadas, o JSON é fácil de ser conhecido, até mesmo por pessoas com menos experiência em áreas de software, o que alavancou a disseminação do formato.

Como o XML, o JSON não é compatível com apenas uma linguagem e pode ser inserido em praticamente qualquer uma delas.



# Diferenças entre XML e JSON

Apesar de suas inúmeras similaridades, essas representações também possuem divergências claras.

JSON, ao contrário do XML, não usa uma marcação para finalizar a que foi criada. Assim, ela acaba tendo uma sintaxe mais curta, que é rápida tanto para escrever quanto para ler. Esse padrão, portanto, é simplesmente uma maneira de representar estruturas de dados, em oposição a uma linguagem de marcação completa como o XML ou o HTML.

Essas características fazem com que os documentos em JSON sejam relativamente mais leves e executados rapidamente no servidor da Web, enquanto arquivos em XML são mais complexos e tem uma padronização madura em suas validações e transformações.

***Formas de Armazenamento de Dados***

# Arquivos

Arquivos são formas de abstração usadas pelos sistemas operacionais para criar uma interação uniforme entre a execução do código e dispositivos exteriores a ele. Normalmente, essa ligação de um código com um dispositivo dado por meio de arquivos transcende três passos sequenciais importantes: a abertura do arquivo, o transporte dos dados e o fechamento do arquivo. Na linguagem Java, a classe File é responsável por conceber os arquivos em tal plano de abstração.

# Banco de Dados

Um banco de dados é uma coleção de informações organizadas para que possam ser facilmente acessadas, gerenciadas e atualizadas.

Os dados são organizados em linhas, colunas e tabelas e são indexados para facilitar a localização de informações relevantes. Os mesmos são mudados, expandidos e excluídos à medida que novas informações são adicionadas.

# JDBC

A API JDBC (Java Database Connectivity) é o padrão do setor para conectividade independente de banco de dados entre a linguagem de programação Java e uma ampla variedade de bancos de dados SQL e outras fontes de dados tabulares, como planilhas ou arquivos simples.

A tecnologia JDBC permite que você use códigos em Java para explorar os recursos "Escreva uma vez, Execute em Qualquer Lugar" para aplicativos que necessitam de acesso a dados corporativos. Com um driver habilitado para tecnologia JDBC, você consegue conectar todos os dados mesmo em um ambiente heterogêneo.

# JPA

A Java Persistence API (JPA) é uma especificação Java para acessar, persistir e gerenciar dados entre objetos/classes Java e um banco de dados relacional. A JPA agora é considerada a abordagem padrão do setor para mapeamento relacional de objeto (OR) na indústria de Java.

O JPA permite que o POJO (Plain Old Java Objects) seja facilmente persistido sem requerer que as classes implementem quaisquer interfaces ou métodos. O JPA permite que os mapeamentos relacionais de objeto de um objeto sejam definidos por meio de anotações padrão ou XML, definindo como a classe Java mapeia para uma tabela de banco de dados relacional.

O JPA em si é apenas uma especificação, não um produto; não pode executar persistência ou qualquer outra coisa por si só.

# Hibernate

Hibernate ORM é, por si só, uma ferramenta de mapeamento relacional de objeto para Java. Ele fornece uma estrutura para mapear um modelo de domínio orientado a objeto para um banco de dados relacional. O Hibernate manipula problemas de incompatibilidade de impedância objeto-relacional, substituindo acessos de banco de dados diretos e persistentes com funções de manipulação de objetos de alto nível.

Um código de aplicativo Java é composto por todas as classes que definem a lógica de negócios do aplicativo. Essas classes se comunicam com o Hibernate e ele usa o Java core API, o JDBC, o JTA, e o JNDI para se comunicar com banco de dados para persistir o estado do objeto, criar, ler, atualizar, excluir.

# Diferenças entre JDBC, JPA e Hibernate

No JDBC é usado o SQL nativo para trabalhar diretamente com os registros, enquanto que no JPA trabalhamos com objetos criados a partir desses registros. Assim, é possível concluir que a performance no acesso ao banco de dados da primeira tecnologia é maior que a da segunda. No entanto, JPA é mais eficiente e menos trabalhoso em termos de desenvolvimento.

Já o Hibernate é a implementação real das diretrizes da JPA. Quando o Hibernate implementar a especificação JPA, ela será certificada pelo grupo JPA ao seguir todos os padrões mencionados na especificação. Em suma, o Hibernate é um provedor JPA. Quando há novas alterações na especificação, o Hibernate libera sua implementação atualizada para a especificação JPA.

# Singleton

Na programação orientada a objetos, uma classe Singleton é uma classe que pode ter apenas um objeto (uma instância da classe) por vez. Ele é usado em cenários em que um usuário deseja restringir a instanciação de uma classe a apenas um objeto. Isso é útil geralmente quando um único objeto é necessário para coordenar ações em um sistema.

Após a primeira vez, se tentarmos instanciar a classe Singleton, a nova variável também apontará para a primeira instância criada. Assim, quaisquer que sejam as modificações que fizermos em qualquer variável dentro da classe por meio de qualquer instância, ela afeta a variável da instância única criada e é visível se acessarmos essa variável por meio de qualquer variável desse tipo de classe definida.

# DAO

**O padrão Data Access Object (DAO) é um padrão estrutural que nos permite isolar a camada de aplicativo da camada de persistência (geralmente um banco de dados relacional) usando uma API abstrata.**

**A funcionalidade dessa API é ocultar do aplicativo todas as complexidades envolvidas na execução de operações no mecanismo de armazenamento subjacente. Isso permite que ambas as camadas evoluam separadamente sem saber nada uma da outra.**

**Ao mapear chamadas de aplicativo para a camada de persistência, o DAO fornece algumas operações de dados específicas sem expor detalhes do banco de dados. Esse isolamento apoia o princípio da responsabilidade única.**

# MVC

Model View Controller (MVC) é um padrão de arquitetura de software, comumente usado para implementar interfaces de usuário: é, portanto, uma opção popular para arquitetar aplicativos da web. Em geral, separa a lógica da aplicação em três partes separadas (Model, Controller e View), promovendo modularidade e facilidade de colaboração e reutilização. Também torna os aplicativos mais flexíveis e receptivos a iterações.

Diversos tipos de aplicações em desktop, web ou mobile podem ser feitas com o modelo MVC. O conceito central dessa arquitetura é o isolamento de seus componentes e do programa. Muitas dessas características podem ser diretamente relacionadas com a própria programação orientada a objetos, escondendo os objetos, mas os manipulando através de uma espécie de interface.

Seguindo esse conceito, uma grande vantagem do MVC é que as aplicações podem ser reutilizadas e adaptadas com pouco esforço para manutenção e/ou inovações.