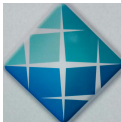


# Programação II

## JDBC

Evandro J.R. Silva<sup>1</sup>

21 de setembro de 2022



# Sumário

**1** JDBC

**2** MVC

**3** DAO

**4** FIM

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.
- A API já vem com o JDK.

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.
- A API já vem com o JDK.
- É bastante comum ser referido como um **middleware**, sendo uma ponte entre aplicação em desenvolvimento (front end) e um banco de dados (back end).

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.
- A API já vem com o JDK.
- É bastante comum ser referido como um **middleware**, sendo uma ponte entre aplicação em desenvolvimento (front end) e um banco de dados (back end).

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.
- A API já vem com o JDK.
- É bastante comum ser referido como um **middleware**, sendo uma ponte entre aplicação em desenvolvimento (front end) e um banco de dados (back end).
- É complementado por outra API chamada **JDBC Driver**. Cada driver é implementado por alguma empresa responsável por algum banco de dados específico.

# Introdução

## ■ JDBC — *Java Database Connectivity*

- É uma API que contém uma série de classes e interfaces para realizar a comunicação entre uma aplicação desenvolvida em Java e o banco de dados relacional utilizado por ela. A API é composta pelos pacotes `java.sql` e `javax.sql`.
- A API já vem com o JDK.
- É bastante comum ser referido como um **middleware**, sendo uma ponte entre aplicação em desenvolvimento (front end) e um banco de dados (back end).
- É complementado por outra API chamada **JDBC Driver**. Cada driver é implementado por alguma empresa responsável por algum banco de dados específico.
- Através do driver, a aplicação Java acessa as implementações de classes e interfaces que vão permitir a execução dos comandos SQL em uma base de dados.



# Introdução

## ■ Tipos de drivers

- **Tipo 1 - Ponte JDBC-ODBC:** Utiliza um driver ODBC para se conectar ao banco de dados.

# Introdução

## ■ Tipos de drivers

- **Tipo 2 - API Nativa:** Usa bibliotecas do lado do cliente para se comunicar com o banco de dados.

# Introdução

## ■ Tipos de drivers

- **Tipo 3 - Protocolo de Rede:** Adota uma camada adicional de software para converter as chamadas JDBC no protocolo utilizado pelo banco de dados.

# Introdução

## ■ Tipos de drivers

- **Tipo 4 - *Thin Driver* (ou Driver Fino):** converte as chamadas JDBC de acordo com o banco de dados do fabricante.

# Conexão e Manipulação de Dados

- O primeiro passo é a conexão da aplicação com o banco de dados.
- Roteiro básico (geral):
  - 1 Criar [objeto] conexão;
  - 2 Executar comandos SQL
    - É necessário um [objeto] cursor para lidar com os comandos.
    - Em Java isso é feito com a classe `Statement`.
  - 3 Encerrar cursor e depois a conexão.
- Códigos ...

# MVC

## ■ MVC — *Model View Controller*

- É um padrão de arquitetura de software.
- A ideia é permitir que uma mesma lógica de negócios possa ser acessada e visualizada através de várias interfaces.
- Isso é possível ao se separar os dados ou lógica de negócios (Model) da interface do usuário (View) e do fluxo da aplicação (Controller).

# MVC

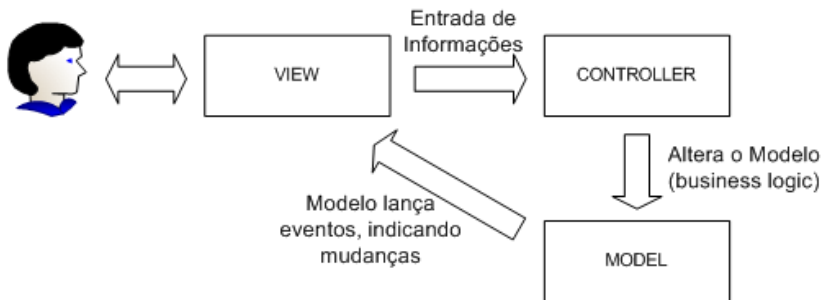


Figura 2: Fluxo de eventos e informações em uma arquitetura MVC

Fonte: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/arqu/mvc/mvc.htm>

# MVC

## ■ Model ou Modelo

- Classe, ou conjunto de classes, cuja responsabilidade é gerenciar e controlar a forma como os dados se comportam por meio das funções, lógica e regras de negócios estabelecidas.
- É a camada responsável pelas regras de negócios, persistência com o banco de dados e as classes de entidades.
- Recebe as requisições do Controller, valida se elas estão corretas e envia a resposta mais adequada.

## ■ View ou Visão

- É a camada responsável por apresentar as informações de forma visual ao usuário.
- É onde ficam os recursos ligados a aparência como mensagens, botões ou telas.
- É também a *interface* por onde o usuário insere seus dados.

## ■ Controller ou Controlador

- A camada que é responsável por intermediar as requisições enviadas pelo View com as respostas fornecidas pelo Model, processando os dados que o usuário informou e repassando para outras camadas.



## Benefícios de se utilizar o MVC

- **Segurança:** O Controller funciona como uma espécie de filtro capaz de impedir que qualquer dado incorreto chegue até a camada modelo.

# Benefícios de se utilizar o MVC

- **Organização:** Esse método de programação permite que um novo desenvolvedor tenha muito mais facilidade em entender o que foi construído, assim como os erros se tornam mais fácil de serem encontrados e corrigidos.

# Benefícios de se utilizar o MVC

- **Eficiência:** Como a arquitetura de software é dividida em 3 componentes , sua aplicação fica muito mais leve, permitindo que vários desenvolvedores trabalhem no projeto de forma independente.

# Benefícios de se utilizar o MVC

- **Tempo:** Com a dinâmica facilitada pela colaboração entre os profissionais de desenvolvimento, o projeto pode ser concluído com muito mais rapidez, tornando o projeto escalável.

# Benefícios de se utilizar o MVC

- **Transformação:** As mudanças que forem necessárias também são mais fluidas, já que não será essencial trabalhar nas regras de negócio e correção de bugs.

# DAO

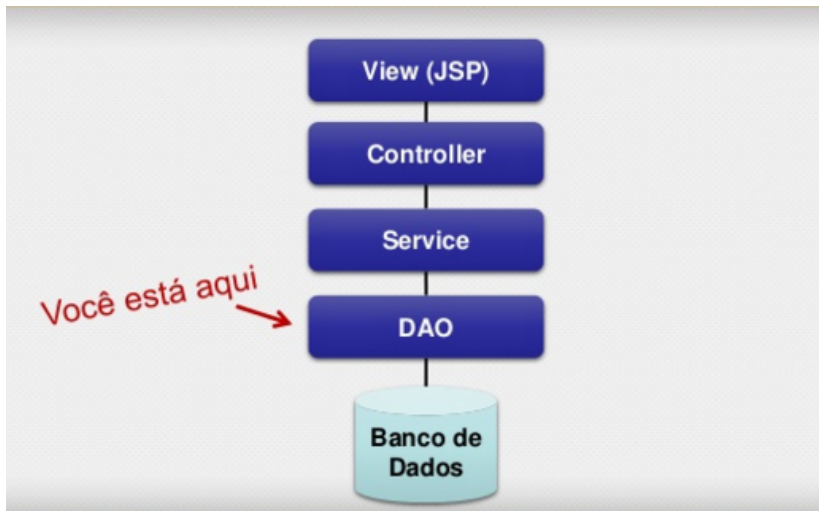
## ■ *Data Access Object*

- É o padrão utilizado para abstrair e encapsular todos os acessos à fonte dos dados (normalmente um banco de dados).
- Gerencia a conexão com a fonte dos dados para obter e armazenar informações.
- Surgiu com a necessidade de se separar a lógica de negócios da lógica de persistência de dados.
- Portanto, o padrão DAO permite que se possa mudar a forma de persistência sem que isso influencie em nada na lógica de negócio, além de tornar as classes mais legíveis.

# DAO

- Classes DAO são responsáveis por trocar informações com o SGBD e fornecer operações CRUD e de pesquisas.
- Devem ser capazes de buscar dados no banco e transforma-los em objetos ou lista de objetos, e também receber objetos e enviá-los ao banco de dados via instruções SQL.

# DAO



Fonte: <https://www.slideshare.net/utluiz/introducao-ao-jdbc>



## FIM

Mais detalhes (oficiais):

<https://docs.oracle.com/javase/tutorial/jdbc/index.html>

# Code Coach

A teoria é fundamental para a prática!

Até a próxima!