## 1. Alguns conceitos importantes

### 1.1 O que é ORM?

Object-Relational Mapping (ORM), em português, mapeamento objeto-relacional, é uma técnica para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional. O uso da técnica de mapeamento objeto-relacional é realizado através de um mapeador objeto-relacional que geralmente é a biblioteca ou framework que ajuda no mapeamento e uso do banco de dados.

Quando estamos trabalhando com aplicações orientadas a objetos que utilizam banco de dados relacionais para armazenamento de informações, temos um problema chamado impedância objeto-relacional devido às diferenças entre os 2 paradigmas.

O banco de dados relacional trabalha com tabelas e relações entre elas para representar modelos da vida real. Dentro das tabelas temos várias colunas e a unidade que temos para representação no modelo relacional é uma linha:

**TABELA: PRODUTO**

| ID | NOME | PREÇO | DESCRIÇÃO |
|---|---|---|---|
| 12 | BICICLETA | R$800 | ENGRENAGEM FIXA, AZUL, RÁPIDA |
| 13 | CAPACETE | R$20,99 | PRETO, AJUSTÁVEL |
| 14 | UNIFORME | R$35 | PEQUENO (FEMININO), VERDE E BRANCO |

O paradigma orientado a objetos possui um modo um pouco diferente de trabalhar. Nele nós temos diversos elementos como classes, propriedades, visibilidade, herança e interfaces. A unidade quando falamos de orientação a objetos é o objeto que representa algo do mundo real, seja abstrato ou concreto:

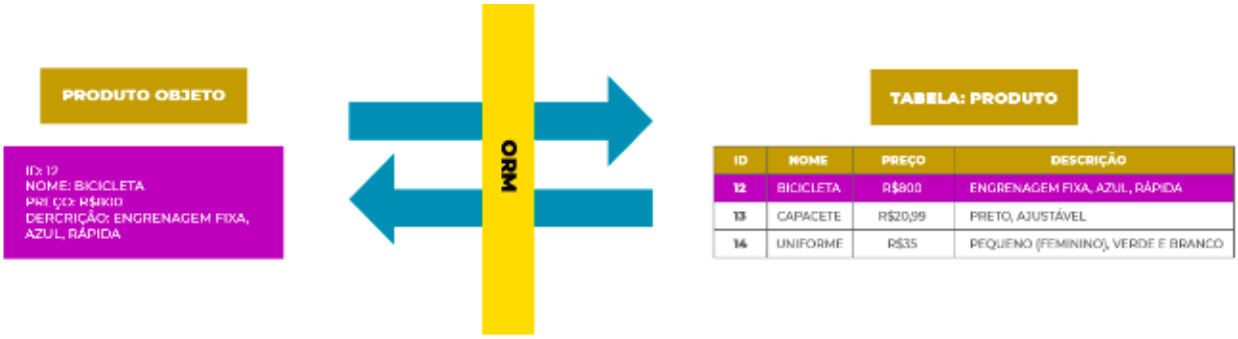
**PRODUTO OBJETO**

ID: 12
NOME: BICICLETA
PREÇO: R$800
DERCRIÇÃO: ENGRENAGEM FIXA, AZUL, RÁPIDA

As principais dificuldades que essas diferenças entre paradigmas causa:

- Representação dos dados e do modelo, já que as estruturas são distintas;
- Mapeamento entre os tipos de dados da linguagem de programação e do banco de dados;
- Modelo de integridade relacional do banco relacional;

Dessa forma, desenvolvedores necessitam criar mecanismos para converter dados em objetos e vice-versa. Pensando nos problemas descritos acima, o ORM define uma técnica para realizar a conciliação entre os 2 modelos. Uma das partes centrais é através do mapeamento de linhas para objetos:




As bibliotecas ou frameworks ORM definem o modo como os dados serão mapeados entre os ambientes, como serão acessados e gravados. Isso diminui o tempo de desenvolvimento, uma vez que não é necessário desenvolver toda essa parte.

No Java os principais padrões de ORM utilizados são:

- Hibernate
- EclipseLink
- ActiveJPA

Durante o nosso curso utilizaremos o Hibernate, que é o framework de mapeamento objeto/relacional mais utilizado do mercado.

### 1.2 Hibernate

Trabalhar com softwares orientados a objetos e banco de dados relacionais pode ser enfadonho e consumir muito tempo de desenvolvimento. Quando a linguagem de programação Java começou a tomar força no mercado, várias soluções começaram a surgir para resolver este problema. A vencedora dessas soluções, que desbancou até mesmo o padrão EJB 2.0, foi o Hibernate, uma ferramenta para mapeamento objeto/relacional para ambientes Java.

O Hibernate hoje é o framework Java para mapeamento objeto relacional mais conhecido no mercado. Sua principal função é abstrair o mapeamento, economizando esforço e preocupações concernentes a tal tarefa. Com uma arquitetura simples, de fácil configuração, e com funções de fácil entendimento, o Hibernate simplifica bastante a tarefa do desenvolvedor.

O Hibernate implementa a especificação JPA (Java Persistence API) através do conceito de anotações (implementada a partir do JDK5), o que facilita ainda mais o mapeamento objeto-relacional, que pode agora ser feito diretamente na classe.

O termo mapeamento objeto/relacional (ORM) refere-se à técnica de mapeamento de uma representação de dados em um modelo de objetos para um modelo de dados relacional baseado em um esquema E/R. O Hibernate não cuida somente do mapeamento das classes Java para tabelas do banco de dados (e dos tipos de dados Java para os tipos de dados SQL), mas também provê facilidades para consultar e retornar os dados da consulta, e pode reduzir significativamente o tempo de desenvolvimento em contrapartida ao alto tempo gasto pelas operações manuais dos dados feitas com SQL e JDBC.

O Hibernate é uma ferramenta de consulta e persistência objeto/relacional de alta performance. Uma das soluções ORM mais flexíveis e poderosas no mercado, ele faz o mapeamento de classes Java para tabelas de banco de dados e de tipos de dados Java para tipos de dados SQL. Ele fornece consultas e facilidades para retorno dos dados que reduzem significativamente o tempo de desenvolvimento.

### 1.2.1 Hibernate Annotations

O Hibernate, como toda ferramenta para mapeamento objeto/relacional, requer metadados que governem as transformações dos dados de uma representação para outra (e vice-versa).

Annotations podem ser divididas em duas categorias:

- **anotações de mapeamento lógico** – permitem descrever o modelo de objetos, a associação entre classes, etc.;
- **anotações de mapeamento físico** – descreve o esquema físico, tabelas, colunas, índices, etc.

### 1.2.2 Mapeando Classes e Atributos com Anotações

Toda classe Java que será persistida em um banco de dados através do Hibernate é considerada uma entidade e é declarada utilizando a anotação @Entity (acima da declaração da classe).

A anotação @Id permite que o usuário defina qual propriedade é a chave primária da sua entidade. A propriedade pode ser preenchida pela própria aplicação ou ser gerada pelo Hibernate (recomendado). É possível definir a estratégia para geração graças à anotação @GeneratedValue.

A anotação @Table é definida em nível de classe e permite que você descreva o nome da tabela que será utilizada para o mapeamento da entidade. Se nenhum @Table for declarado, os valores padrões serão utilizados: no caso, o próprio nome da classe.

É possível mapear as colunas de uma tabela com a anotação @Column. Utilize isso para sobrescrever o valor padrão, no caso o nome do próprio atributo. O código abaixo apresenta um exemplo.

Todas essas anotações e muitas outras veremos em nossos práticos que veremos em sala de aula.

## 2. Spring

O Spring é um ecossistema de desenvolvimento para facilitar a criação de aplicações Java utilizando diversos módulos independentes. É um framework Java que trás os conceitos de Inversão de Controle e Injeção de Dependências. Quando o utilizamos, temos à nossa disposição uma tecnologia que nos fornece não apenas recursos necessários à grande parte das aplicações, como módulos para persistência de dados, integração, segurança, testes, desenvolvimento web, como também um conceito a seguir que nos permite criar soluções menos acopladas, mais coesas e, consequentemente, mais fáceis de compreender e manter.

### Inversão de controle e Injeção de dependência

Para entender esses conceitos, saiba que a Inversão de Controle (IoC - Inversion of Control) permite delegar a outro elemento o controle sobre como e quando um objeto deve ser criado e quando um método deve ser executado, por exemplo. Assim, essas responsabilidades, isto é, o controle sobre a execução de alguns comportamentos, passa a ser gerenciado por esse elemento, não cabendo mais a nós, programadores, definirmos isso.No caso do Spring, chamamos esse elemento de container. Temos, assim, o princípio da inversão do controle.

E a Injeção de Dependência? A Injeção de Dependência (DI - Dependency Injection) é uma das maneiras de implementar a Inversão de Controle. Com a Injeção de Dependência a classe deixa de se preocupar em como resolver as suas dependências. Ela passa a manter o foco apenas no uso dos recursos das dependências para realizar as tarefas que precisa. E isso leva a uma das características mais conhecidas quando programamos com Spring: não precisamos utilizar o new para criar os objetos por ele gerenciados, pois isso passa a ser feito pelo framework.

Como a ampla utilização do Spring, surgiu a ideia de modularizar as principais funcionalidades do Spring Framework e assim facilitar a integração com outros projetos que utilizavam outros frameworks. O Spring se tornou um conjunto de projetos em que cada um deles tem a finalidade de solucionar um problema específico, dentre os mais populares são:

- **Spring Boot:** Módulo do Spring que facilita a configuração e publicação das aplicações desenvolvidas, visando reduzir a quantidade de configurações iniciais que geralmente são fundamentais em projetos java;
- **Spring Data:** Módulo do Spring que visa facilitar a forma de acesso aos dados por parte da aplicação. Possui suporte desde o JDBC até o JPA;
- **Spring MVC:** Permite a criação e desenvolvimento de aplicações utilizando o padrão MVC;
- **Spring Security:** Módulo do Spring responsável por gerenciar toda a parte de autenticação e autorização de uma aplicação;
- **Spring Mobile:** Facilita a criação de aplicações web que também serão executadas em dispositivos móveis.

Os projetos do ecossistema Spring são independentes, isso quer dizer que podemos utilizá-los de forma isolada, como por exemplo, em um sistema web podemos fazer uso do Spring

Framework mas se porventura precisarmos de mais funcionalidades, podemos também combiná-los conforme a nossa necessidade.

Outra característica do Spring é que seus projetos cuidam da infraestrutura da aplicação. E o que seria essa infraestrutura? São partes da aplicação, importantes para o seu funcionamento, mas que não estão diretamente ligadas às suas regras de negócio. Por exemplo, imagine que no front-end da sua aplicação existe um formulário. No back-end, certamente teremos uma classe pra esse formulário.

O Spring foi feito para aplicações corporativas, que podem começar pequenas, mas tendem a crescer à medida em que vão cobrindo mais áreas do negócio. Na prática, isso quer dizer que é possível criar aplicações menores, tipo um portal, blog, etc. Sendo assim trabalhar com o Spring é um modo simples e confiável de construir aplicações em Java.