

EAPLI
Teórico-Prática

Mapeamento Objeto/Relacional

JPA – Parte 1

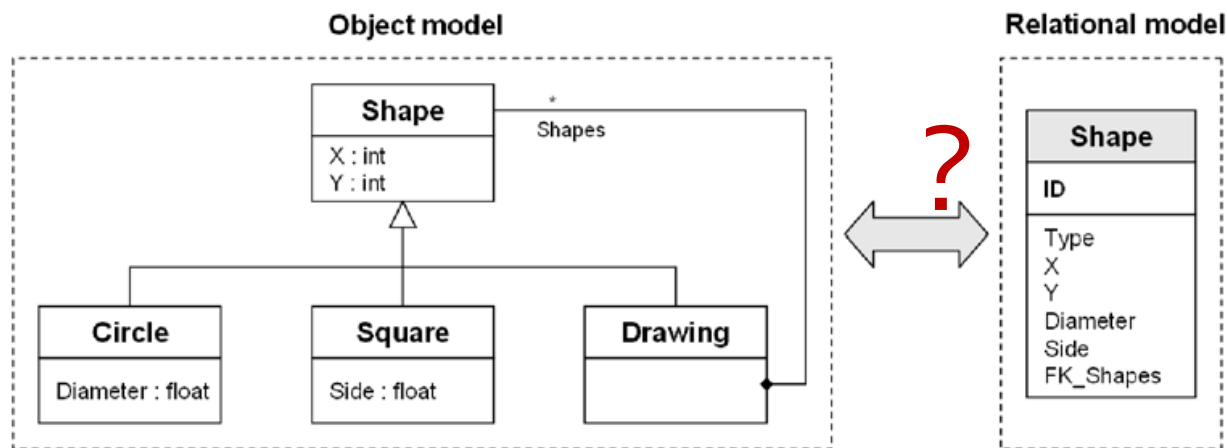
Object-Relational Mapping (ORM)

Conceitos

Mapeamento Objeto/Relacional

- **O problema:**

- Bases de dados relacionais são baseadas em algebra relacional e tratam de tabelas, linhas, colunas e SQL;
- Programação orientada a objetos é sobre classes, objetos, métodos, herança, polimorfismo e instruções de programação.



- **Como compatibilizar estes diferentes paradigmas?**
- **Como persistir (manter) a informação dos objetos num ambiente OO?**
- **Solução: Utilizar Mapeamento Objeto/Relacional**

Mapeamento Objeto/Relacional

- **Mapeamento Objeto/Relacional (ORM)** é o processo de conversão bidirecional entre objetos e tabelas de um modelo relacional.
- **Aproveita:**
 - as vantagens da programação em OO na modelação de aplicações complexas.
 - a eficiência e fiabilidade demonstradas pela utilização de bases de dados relacionais no armazenamento de grandes volumes de dados.

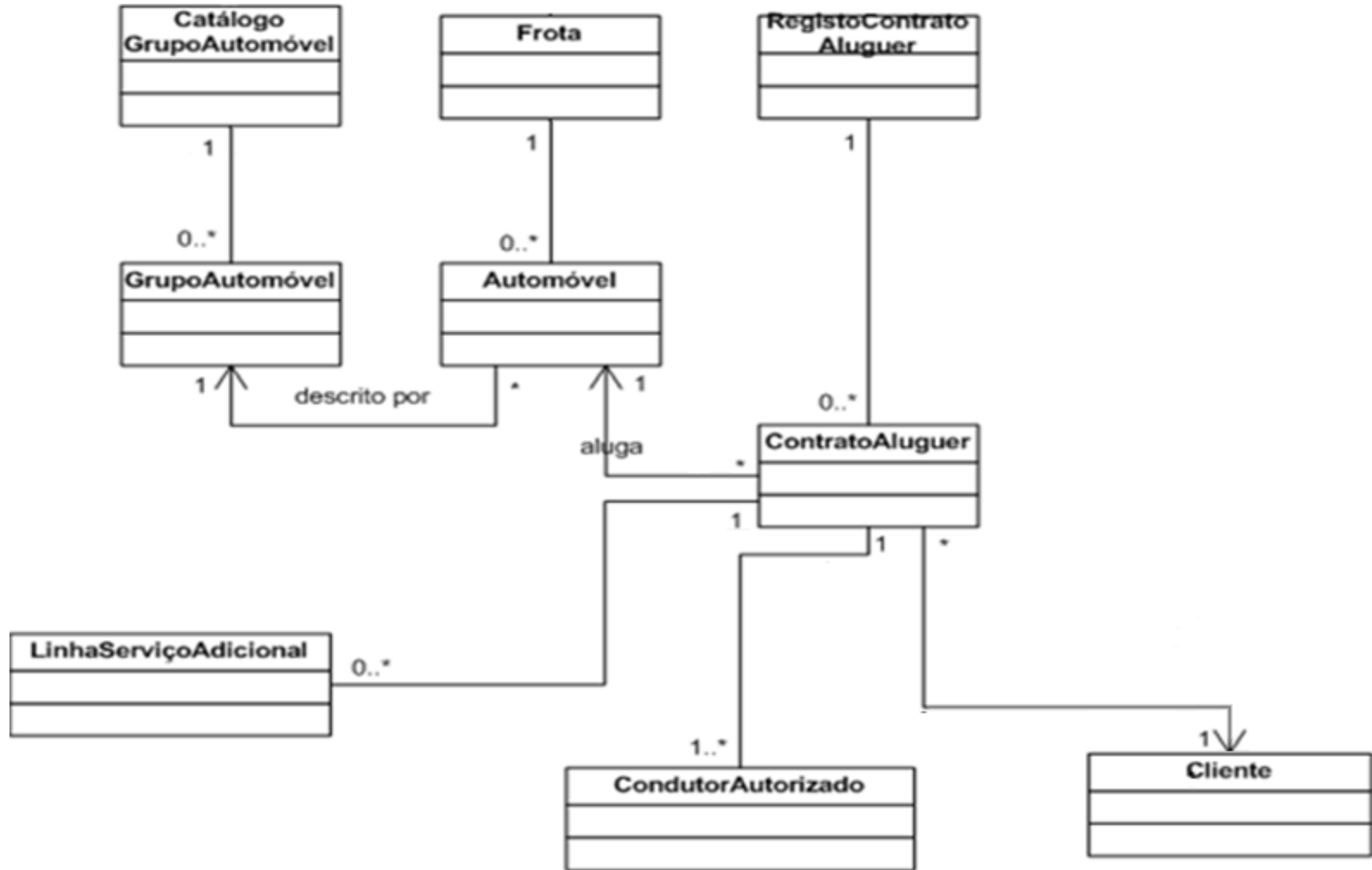
Mapeamento Objeto /Relacional

- A automatização deste processo de conversão conduziu ao desenvolvimento de **frameworks** ORM
 - *Hibernate* (Java)
 - *NHibernate* (C#)
 - *iBATIS* (Java / .NET)
 - *TopLink* (Oracle)
 - *Entity Framework – EF* (.Net)
 - *Java Persistence API – JPA* (Java)
 - ...

Object-Relational Mapping (ORM)

JPA - Java Persistence API

Modelo de domínio para os exemplos



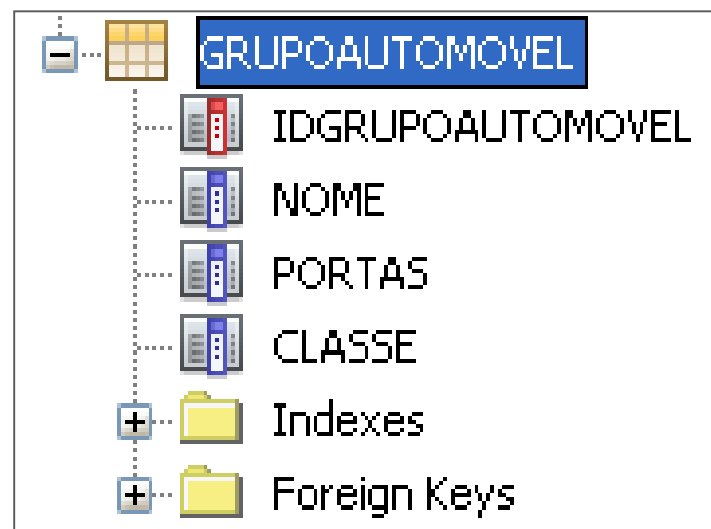
Mapeamento ORM

■ Objetivo:

```
public class GrupoAutomovel {  
    private String nome;  
    private Integer portas;  
    private String classe;  
    (...)  
}
```

Objecto

Relational

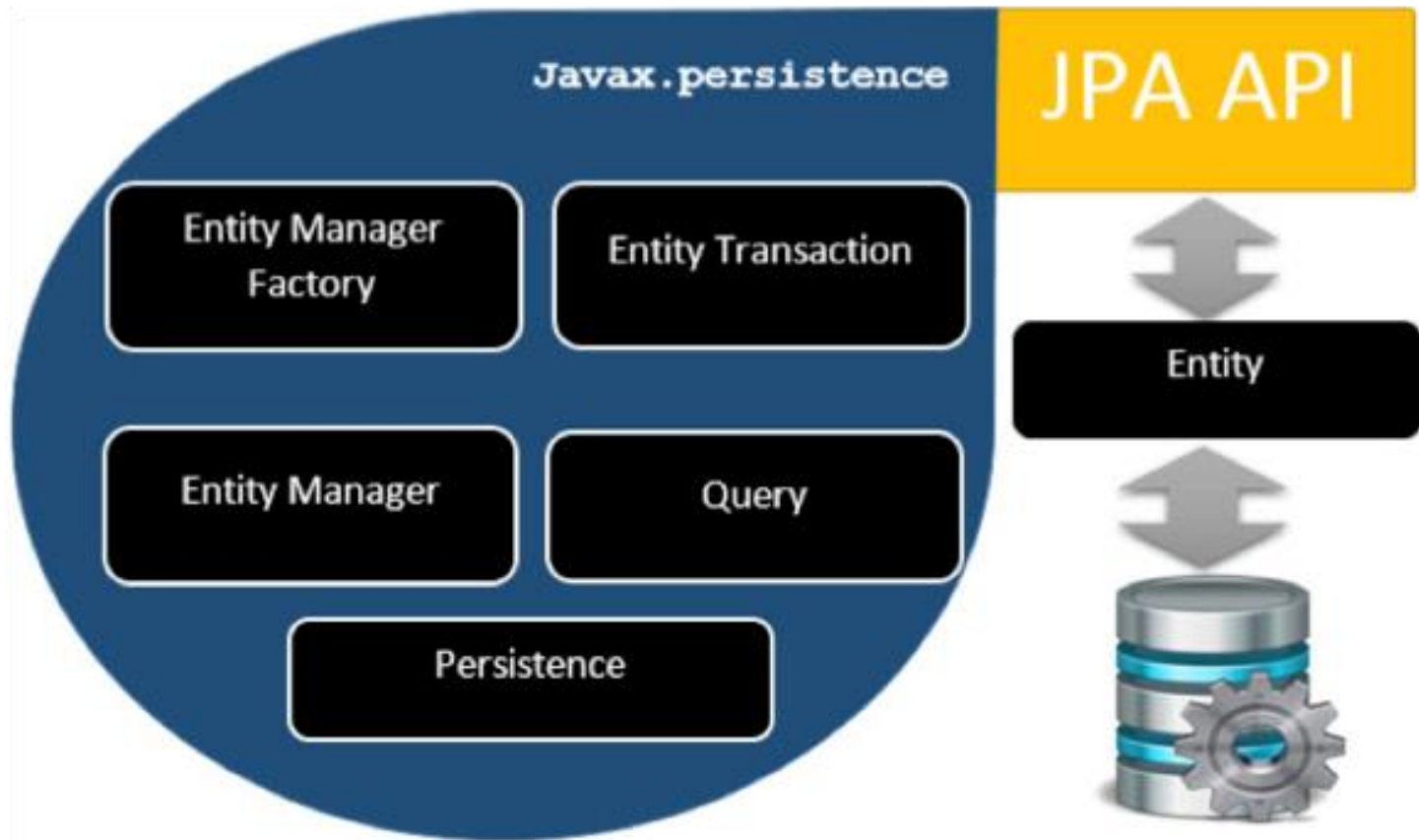


Java Persistence API

- API que **disponibiliza funcionalidades ORM** para persistir objetos java em bases de dados relacionais.
- Sustentado num conjunto de classes e de anotações de metadata definidos no *package* **javax.persistence**
- Persistência em Java implica:
 - Java Persistence API (JPA)
 - Java Persistence Query Language (JPQL)
 - Java Persistence Criteria API
 - ORM metadata

Java Persistence API

■ Class Level Architecture



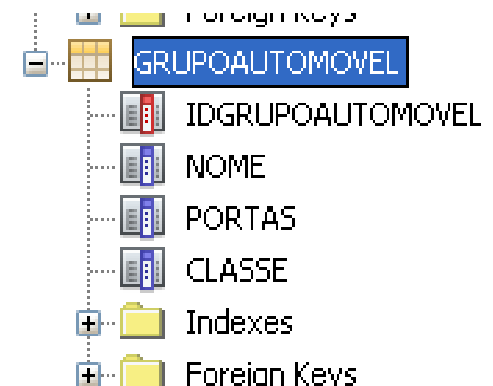
Entity - JPA

- **Entity** é um objecto do domínio.
- É uma classe em Java com metadata para descrever como o seu estado deve ser mapeado em tabelas relacionais.
 - metadata são anotações às classes e variáveis de instância.
- **Normalmente:**
 - uma Entity representa uma **tabela** na base de dados;
 - cada instância representa um **registo** dessa tabela.
- **Indica as classes que vão ser mapeadas em tabelas.**
 - Associa uma classe a uma tabela na base de dados.
 - Por omissão, o **nome da tabela é igual ao nome da classe**.
- Tem uma **persistency entity** que corresponde à chave primária da tabela na BD

Entity - JPA

Annotations

```
@Entity  
@Table(Name=GRUPOAUTOMOVEL)  
public class GrupoAutomovel {  
    @Id //define a chave primária  
    @GeneratedValue(strategy=GenerationType.AUTO)  
    @Column(name="IDGRUPOAUTOMOVEL")  
    int id;  
    String nome;  
    Integer Portas;  
    String Classe;  
  
    protected GrupoAutomovel(){} //Obrigatório  
  
    (...)  
}
```



Entity - JPA

Todas as Entities têm de ter um atributo com anotação **@Id** que será mapeado para a chave primária da tabela.

```
@Entity
@Table(Name=GRUPOAUTOMOVEL)
public class GrupoAutomovel {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="IDGRUPOAUTOMOVEL")
    int id;
    String nome;
    Integer Portas;
    String Classe;

    protected GrupoAutomovel(){}

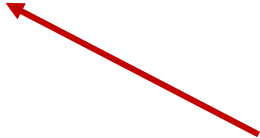
    (...)
}
```

Entity - JPA

```
@Entity
@Table(Name=GRUPOAUTOMOVEL)
public class GrupoAutomovel {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="IDGRUPOAUTOMOVEL")
    int id;
    String nome;
    Integer Portas;
    String Classe;

    protected GrupoAutomovel(){

    (...)
}
```




@GeneratedValue indica que o valor do atributo chave primária é gerado pela base de dados no momento em que um novo registo é inserido.

Entity - JPA

```
@Entity
@Table(Name=GRUPOAUTOMOVEL)
public class GrupoAutomovel {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="IDGRUPOAUTOMOVEL")
    int id;
    String nome;
    Integer Portas;
    String Classe;

    protected GrupoAutomovel(){

    (...)
}
```



A anotação **@Column** (opcional) permite especificar explicitamente o mapeamento entre uma variável de instância e a respectiva coluna da tabela.

Entity - JPA

```
@Entity
@Table(Name=GRUPOAUTOMOVEL)
public class GrupoAutomovel {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    @Column(name="IDGRUPOAUTOMOVEL")
    int id;
    String nome;
    Integer Portas;
    String Classe;

    protected GrupoAutomovel(){}

    (...)
}
```

Uma *Entity* tem de ter um **construtor vazio**.

Esse construtor deverá ser classificado como *protected* ou como *public* (se não quebrar regras de negócio).

Anotações JPA

@Entity

@Table(name="T_Pessoa")

public class Pessoa {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name="P_Nome", length=50)

private String primeiroNome;

@Column(name="U_Nome", nullable=false)

private String ultimoNome;

@Transient

private int idade;

@Temporal(TemporalType.DATE)

private Date dataNascimento;

@Enumerated(EnumType.STRING)

private Genero genero;

(...)

}

Um atributo com anotação **@Transient** não será persistido.

```
public enum Genero {  
    MASCULINO, FEMININO  
}
```

Tabela criada:

T_PESSOA(ID, P_NOME, U_NOME, DATANASCIMENTO, GENERO)

1	José	Manuel	1995-01-09	MASCULINO
---	------	--------	------------	-----------

Anotações JPA

■ Unique constraint

```
@Entity
class Automovel {
    @Id
    long pk;

    @Column(unique=true)
    String matricula;
    (...)
}
```

Entity Manager - JPA

- **Entity manager** é a classe principal da API.
- É com este objecto que se criam novas entidades e se criam *queries* para retornar conjuntos de entidades existentes.
- É através deste objecto que se realiza o CRUD na BD.

```
import javax.persistence.*;
...
EntityManagerFactory factory =
    Persistence.createEntityManagerFactory("carros");
EntityManager manager = factory.createEntityManager();

GrupoAutomovel grupo1=new GrupoAutomovel("C",3,"Utilitário");

manager.getTransaction().begin();
manager.persist(grupo1);
manager.getTransaction().commit();
manager.close();
```

Grava a entidade (objeto)
na base de dados

Persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
    http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
    <persistence-unit name="Carros">
        <!-- provedor/implementacao do JPA -->
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <exclude-unlisted-classes>false</exclude-unlisted-classes>
        <class>Model.GrupoAutomovel</class>
        <class>Model.Automovel</class>
        <!-- H2 embeded database on local folder -->
        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>
            <property name="javax.persistence.jdbc.url"
                value="jdbc:h2:./db/Automovels;create=true" />
            <property name="javax.persistence.jdbc.user" value="" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="eclipselink.ddl-generation" value="drop-and-create-tables" />
        </properties>
    </persistence-unit>
</persistence>
```

Arquitetura JPA

■ Conceitos

