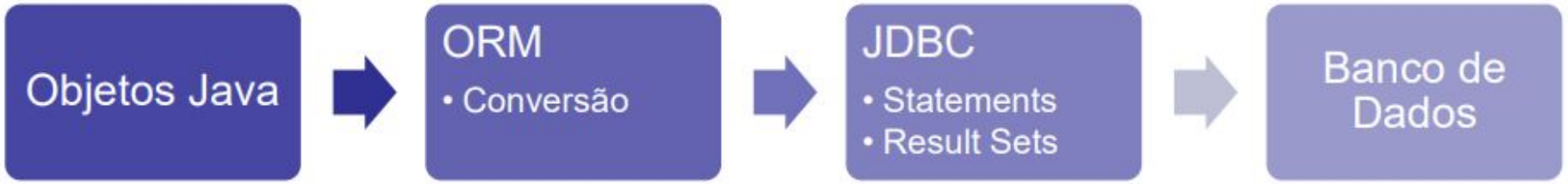
HIBERNATE

JPA - Hibernate

- JPA - Java Persistence API
- Especificação padrão para mapeamento objeto-relacional e gerenciamento de persistência da plataforma Java EE.
- Possui amplo suporte pela maioria dos grandes players do mercado: Apache, BEA, JBoss

ORM - Mapeamento Objeto-Relacional



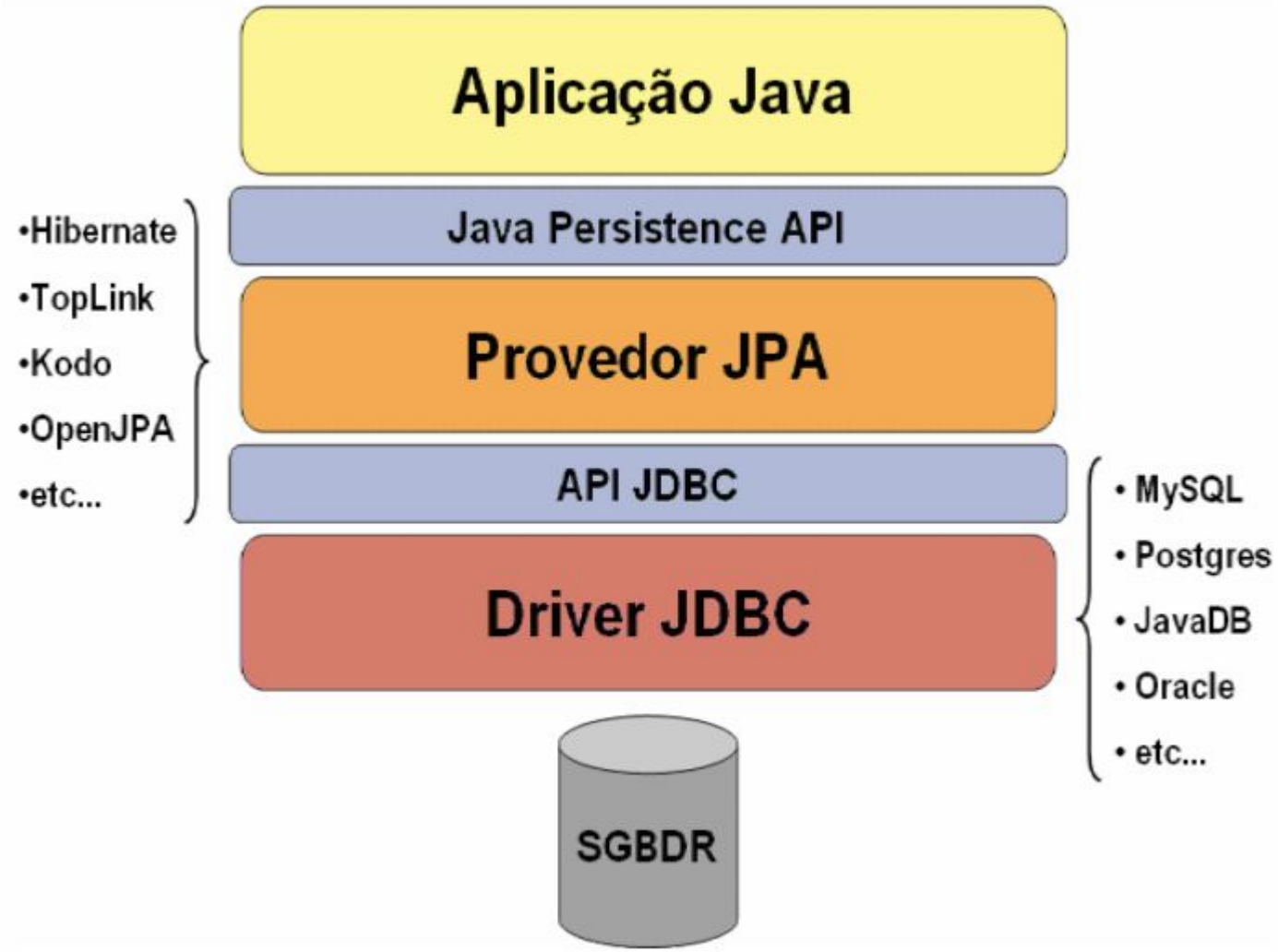
MODELO OO	MODELO RELACIONAL
> Classe	> Tabela
> Objeto	> Linha
> Atributo	> Coluna
> Associação	> Chave Estrangeira

- Mapeamento via XML ou Annotations

Funcionalidades do JPA

- Padroniza Mapeamento Objeto-Relacional
- Utiliza POJO's ao invés de Entity Beans
- Pode ser usado com Java SE e Java EE
- Suporta utilização de diferentes Providers
- Possui uma linguagem de consulta extendida
- Suporta herança, polimorfismo

FrameWorks


Aplicação Java
Java Persistence API
Provedor JPA
API JDBC
Driver JDBC
SGBDR
•Hibernate
•TopLink
•Kodo
•OpenJPA
•etc...
• MySQL
• Postgres
• JavaDB
• Oracle
• etc...

JPA Provider - Hibernate

- No Drive, na pasta “Hibernate” baixar o arquivo “hibernate-release-5.4.1.Final.zip” e o arquivo do conector MySQL “mysql-connector-java-5.1.47.jar”.
- Descompactar os arquivos no diretório do seu projeto JAVA.
- Abra o Eclipse e entre no Menu >Window >Preferences >Java >Build Path >User Libraries
- Clicar no botão “New...”, escreva um nome para a biblioteca “Hibernate”;

JPA Provider - Hibernate

- Clique no botão “Add External JARs...” e na pasta descompactada do hibernate, vá até a pasta ...\lib\required e marque todos os arquivos.

- Clique no botão “New...” e vamos criar a biblioteca do conector MySQL, dê o nome de “MySQL”;
- Clique no botão “Add External JARs...” e na pasta descompacta do conector MySQL, adicione o arquivo “mysql-connector-java-5.1.47.jar”

- Clique no botão “Apply and Close”

JPA Provider - Hibernate

- Crie um projeto java chamado “Hibernate5”;

- Nosso projeto irá trabalhar com a biblioteca do Hibernate e do conector MySQL. Basta adicionar as mesmas no projeto:
  - Botão Direito no nome do projeto >Build Path >Configure Build Path.
  - Abra a aba “Libraries” e clique no botão “Add Library” >User Library >Next e adicione as duas bibliotecas que adicionamos no eclipse, >Finish e >Apply and Close;

Configurando o arquivo persistence.xml

- No Drive, na pasta “Hibernate” baixar o arquivo “persistence.xml”;
- No Eclipse, criar uma pasta “META-INF” dentro da pasta “src”;
- Copiar o arquivo “persistence.xml” para dentro desta pasta;

Configurando o arquivo persistence.xml

<persistence-unit name="cursopcv" transaction-type="RESOURCE_LOCAL">

- "unit name" □ define para a aplicação a configuração que será utilizada;
- "transaction-type" □ define o nosso tipo de transação, no nosso caso "local", se fosse web poderia usar o conteinner Jboss por exemplo.
- Podemos trabalhar com várias "persistence-unit";

<provider>org.hibernate.ejb.HibernatePersistence</provider>

- Indica qual será o provider de nossa aplicação. Usaremos o Hibernate

<property name="hibernate.show_sql" value="true" />

- Habilita a exibição do SQL gerado no console

Configurando o arquivo persistence.xml


```
<property name="hibernate.connection.driver_class" value="com.mysql.jdbc.Driver" />
<property name="hibernate.connection.url" value="jdbc:mysql://localhost/bancohiber5" />
<property name="hibernate.connection.user" value="root" />
<property name="hibernate.connection.password" value="123456" />
```

- Configuração do driver (MySQL), da url, usuário e senha

```
<property name="hibernate.dialect" value = "org.hibernate.dialect.MySQL5InnoDBDialect" />
```

- Dialeto que o hibernate irá utilizar, pesquisar sobre os dialetos, como por exemplo, como gerar script do banco de dados de modo automático;

```
<property name="hibernate.hbm2ddl.auto" value="update"/>
```

- Definida como ativada, o Hibernate atualizará suas tabelas quando necessário. Por exemplo, se a tabela não existir ele irá criar (CREATE TABLE), se adicionar uma coluna nova (ALTER TABLE).

Mapeamento Objeto-Relacional


```java
package com.cursopcv.entidades;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="dinossauros")
public class Dinossauros {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int codigo;

    @Column
    private String genero;

    @Column
    private String especie;
```

Entidades

- @Entity
  - Especifica que uma classe é uma entidade;
  - Uma entidade é um objeto que pode ser persistido;
  - Representa uma tabela no banco de dados relacional.

- @Table
  - Especifica nome da tabela no banco de dados.

```
@Entity
@Table(name="dinossauros")
public class Dinossauros {
```

Atributos

- @Column
  - Mapeia um atributo ou uma propriedade (getter) a um campo do banco de dados;
  - Possui diversas opções de validação
    - Lanca javax.persistence.PersistenceException

```
@Column(name = "GENERO", nullable = false, length = 45, unique = false)
private String genero;

@Column
private String especie;
```

Chave Primária Simples

- @Id
  - Cada entidade precisa possuir uma chave primária
  - Mapeia uma chave primária simples
  - Chave pode ser gerada automaticamente
    - IDENTITY, AUTO, SEQUENCE, TABLE

```
@Id
@GeneratedValue(strategy=GenerationType.IDENTITY)
private int codigo;
```

Chave Primária Composta

- @Embeddable
  - Define que uma classe pode fazer parte de uma entidade;
- @Embeddedld
  - Define uma propriedade que é embeddable como chave primária;

Iniciando o projeto

- Criar no MySQL Workbench um database chamado “bancohiber5”;
- Para nosso projeto, no arquivo “persistence.xml”, vamos alterar o nome da “persistence-unit” para “hiberMySQL” e também o nome do seu banco na propriedade “hibernate.connection.url”
- No nosso projeto “Hibernate5,” criar um pacote chamado “com.cursopcv.entidade” para armazenarmos todas as entidades do nosso projeto;
- Neste pacote criar uma classe chamada “Dinossauros”;

Entidade - Dinossauros


```java
package com.cursopcv.entidades;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="dinossauros")
public class Dinossauros {

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private int codigo;

    @Column(name = "GENERO", nullable = false, length = 45, unique = false)
    private String genero;

    @Column
    private String especie;

}
```

- Criar os Getter's e Setter's

Repositório - Persistir Acesso BD

> javax.persistence.EntityManager

- Gerencia o ciclo de vida das entidades
  - NEW, MANAGED, DETACHED, REMOVED

- Utilizado para criar e remover entidades, buscar entidades pela chave primária e fazer consultas

- O Conjunto de entidades que podem ser gerenciados por um EntityManager é definido dentro da Persistence Unit;

Repositório - Persistir Acesso BD

- Crie um pacote “com.cursopcv.repositorio;
- Cria a classe “RepositorioDinossauros” que será responsável pela persistência ao banco de dados;
- Nomear a interface “EntityManagerFactory” - Controlar a unidade de persistência vai ser usada
- Nomear a interface “EntityManager” que vai fazer nossas transações.

```
EntityManagerFactory emf;
EntityManager em;
```

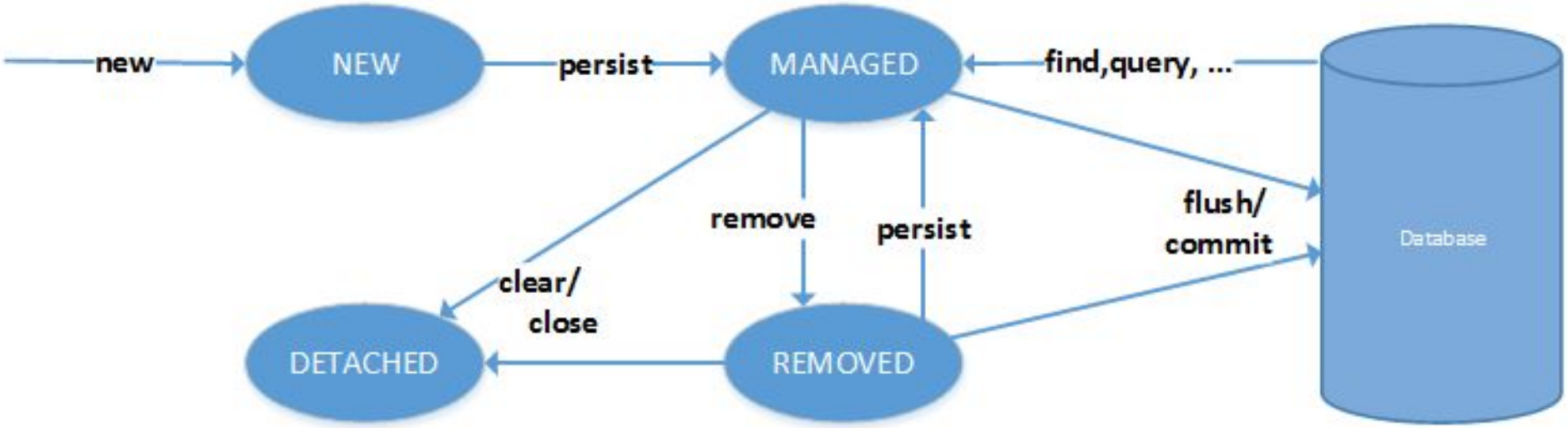
Repositório - Persistir Acesso BD

- Criar um construtor para instanciar nossos objetos;
  - O objeto do tipo EntityManagerFactory vai receber da classe “Persistence” o método “createEntityManagerFactory”, passando como parâmetro o nome definido no arquivo persistence.xml
  - O objeto do tipo EntityManager vai receber do objeto EntityManagerFactory o método “createEntityManager()”.

```
EntityManagerFactory emf;
EntityManager em;

public RepositorioDinossauros() {
    emf = Persistence.createEntityManagerFactory("cursoPCV");
    em = emf.createEntityManager();
}
```

Estados de Um Objeto JPA



Repositório - Persistir Acesso BD

- Criar o método salvar()
  - Inicia a transação > getTransaction().begin()
  - Chama um dos métodos
    - merge() - Quando uma entidade desconectada é atualizada, todas as entidades da coleção são atualizadas
    - persist() - Quando uma nova entidade é persistida, todas as entidades na coleção são persistidas;
  - Chama o método para commitar a transação e o método para fechar a conexão;

```
public void salvar(Dinossauros dinos) {
    try {
        em.getTransaction().begin();
        em.merge(dinos);
        em.getTransaction().commit();
    } finally {
        emf.close();
    }
}
```

Classe Teste

- Criar um novo pacote “com.cursopcv.testes” e criar uma classe com o método main chamada “TesteHiber5”
- Criar um objeto do tipo “RepositorioCliente” e um objeto do tipo “Dinossauros”
- No objeto do tipo Dinossauros, atribuir 3 valores diferentes, chamar o método “salvar”, rodar o programa e verificar no MySQLWorkbench a tabela e os registros criados

```
RepositorioDinossauros repDino = new RepositorioDinossauros();
Dinossauros dino = new Dinossauros();

dino.setEspecie("Taurandios");
dino.setGenero("SufilosBasculinos");
dino.setEspecie("Tiranusauros");
dino.setGenero("SufriciosSacrifis");
dino.setEspecie("Basculanos");
dino.setGenero("Varandanilius");

repDino.salvar(dino);
```

Repositório - Persistir Acesso BD

- Criar o método remover()
  - Inicia a transação > getTransaction().begin()
  - Chama o método remove();
  - Chama o método para commitar a transação e o método para fechar a conexão;

```
public void remover(Dinossauros dinos) {
    try {
        em.getTransaction().begin();
        em.remove(dinos);
        em.getTransaction().commit();
    } finally {
        emf.close();
    }
}
```

Classe Teste

- Criar um novo método no repositório “obterPorCod” para retornar um determinado registro, chamar o método na página principal, listar o registro informado e chamar o método “remover”. Verificar na tabela se o registro foi excluido com sucesso.

```
public Dinossauros obterPorCod(int codigo) {
    em.getTransaction().begin();
    Dinossauros dino = em.find(Dinossauros.class, codigo);
    em.getTransaction().commit();
    return dino;
}
```

```
dino = repDino.obterPorCod(3);
System.out.println("Registro da Tabela Dinossauros a ser Deletado");
System.out.println("##############################################");
System.out.println("Gênero do Dinossauro.:" + dino.getGenero());
System.out.println("Espécie do Dinossauro:" + dino.getEspecie());
System.out.println("##############################################");
repDino.remover(dino);
```

Classe Teste

- Aproveitar o método que pesquisa por código e acrescentar alteração no gênero e espécie:

```
RepositorioDinossauros repDino = new RepositorioDinossauros();
Dinossauros dino = new Dinossauros();

dino = repDino.obterPorCod(1);
System.out.println("Registro da Tabela Dinossauros Antes da Alteração");
System.out.println("###############################################");
System.out.println("Gênero do Dinossauro.:" + dino.getGenero());
System.out.println("Espécie do Dinossauro:" + dino.getEspecie());
System.out.println("###############################################");
dino.setEspecie("Basculanos Alterado ");
dino.setGenero("Varandanillius Alterado");
repDino.salvar(dino);
System.out.println("Registro da Tabela Dinossauros Após Alteração");
System.out.println("###############################################");
System.out.println("Gênero do Dinossauro.:" + dino.getGenero());
System.out.println("Espécie do Dinossauro:" + dino.getEspecie());
System.out.println("###############################################");
```

Repositório - Persistir Acesso BD

- Criar o método lista dos dados – “listarTodos”
- Inicia a transação > getTransaction().begin()
- Monta o select em uma “Query” (javax.persistente)
- Joga o resultado em uma lista;
- Fecha conexão e retorna a lista;

```
public List<Dinossauros> listarTodos() {
    em.getTransaction().begin();
    Query consulta = em.createQuery("select dinos from Dinossauros dinos");
    List<Dinossauros> dinos = consulta.getResultList();
    em.getTransaction().commit();
    emf.close();
    return dinos;
}
```

Repositório - Persistir Acesso BD

- Na nossa página principal do projeto, fazer a chamada do método “listarTodos”;

```
List<Dinossauros> dinossauros = repDino.listarTodos();

for (int i = 0; i < dinossauros.size(); i++) {
    System.out.println("Genero do Dinossauro.: " + dinossauros.get(i).getGenero());
    System.out.println("Espécie do Dinossauro: " + dinossauros.get(i).getEspecie());
    System.out.println("##################################################");
}
```