

---

---

# Persistência

---

---

---

---

**Como persistir uma aplicação  
orientada a objetos?**

---

---

# A realidade dos bancos de dados

Relacionais

Vs

Orientado à objetos

# Banco de dados Orientados a Objetos

Um **banco de dados orientado a objetos** é um banco de dados em que cada informação é armazenada na forma de objetos, ou seja, utiliza a Estrutura de dados denominada Orientação a objetos.

# Banco de dados Orientados a Objetos

- Object Definition Language (**ODL**)
- Object Query Language (**OQL**)

# Bancos de dados Orientados a Objetos

**CACHÉ:** Java, .Net, C++, XML e outras. É um banco de dados comercial.

**VERSANT:** Java e C++;

**DB4Objects:** Java e .Net. Sua linguagem de Consulta é a Object Query Language (OQL);

O2: C, C++ e o ambiente O2. Sua linguagem de Consulta: O2Query, OQL;

# Bancos de dados Orientados a Objetos

**GEMSTONE:** Java, C++, C#, XML e outras. Sua linguagem de Consulta é o DML. - JASMINE: Web, suporte à linguagem Java.

**MATISSE:** Java, C#, C++, VB, Delphi, Perl, PHP, Eiffel, SmallTalk.

**Objectivity/DB:** trabalha com as seguintes linguagens: C#; C++; Java; Python, Smalltalk; SQL++ e XML.

**Ozone:** Java e XML.

---

---

# JPA com Hibernate

— Persistência por Mapeamento  
objeto-relacional —

---

---



# JPA

**Java Persistence API** (ou simplesmente **JPA**) é uma API padrão da linguagem Java que descreve uma interface comum para frameworks de persistência de dados. A JPA define um meio de mapeamento objeto-relacional para objetos Java simples e comuns (POJOs), denominados *beans de entidade*

# O que é mapeamento Objeto-Relacional

Técnica de desenvolvimento utilizada para reduzir a impedância da programação orientada aos objetos utilizando bancos de dados relacionais.

# Hibernate

O Hibernate é um framework para o mapeamento objeto-relacional escrito na linguagem Java, mas também é disponível em .Net como o nome NHibernate.



**JPA**



**HIBERNATE**



SQL SERVER



POSTGRES SQL



MYSQL



ORACLE

# Na prática

1. Para usar hibernate é necessário baixar suas dependências;  
**[www.hibernate.org](http://www.hibernate.org)**
2. Copiar aos arquivos .jar da pasta required e o .jar do jpa para o projeto.
3. Adicionar todos ao *classpath* do projeto.

# Mapeando

```
13 @Entity
14 public class Evento {
15
16     @Id
17     @GeneratedValue
18     private Long id;
19
20     private String descricao;
21     private boolean finalizado;
22
23     @Temporal(TemporalType.DATE)
24     @Column(name = "data_finalizacao", nullable = true)
25     private Calendar dataFinalizacao;
```

# persistense.xml

```
<persistence-unit name="Evento">

    <!-- provedor/implementacao do JPA -->
    <provider>org.hibernate.ejb.HibernatePersistence</provider>

    <!-- entidade mapeada -->
    <class>web.web.Evento</class>

    <properties>
        <!-- dados da conexao -->
        <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
        <property name="javax.persistence.jdbc.url" value="jdbc:mysql://127.0.0.1/web" />
        <property name="javax.persistence.jdbc.user" value="root" />
        <property name="javax.persistence.jdbc.password" value="" />

        <!-- propriedades do hibernate -->
        <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5InnoDBDialect" />
        <property name="hibernate.show_sql" value="true" />
        <property name="hibernate.format_sql" value="true" />

        <!-- atualiza o banco, gera as tabelas se for preciso -->
        <property name="hibernate.hbm2ddl.auto" value="update" />
    
```

---

```
</properties>
</persistence-unit>
```

# Obtendo Fábrica e Gerente para uma Entidade

1. Cria um objeto específico para uma determinada entidade;

```
EntityManagerFactory factory = Persistence.  
    createEntityManagerFactory("Evento");
```

2. Cria um gerenciado para uma entidade a partir de uma fábrica;

```
EntityManager manager = factory.createEntityManager();
```

3. Fecha os recursos

```
manager.close();  
factory.close();
```



# Persistindo objetos

```
18  
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22  
23 EntityManager manager = factory.createEntityManager();  
24  
25 Evento E1 = new Evento();  
26 E1.setFinalizado(false);  
27 E1.setDescricao("Estudando JPA");  
28 E1.setDataFinalizacao(Calendar.getInstance());  
29  
30 manager.getTransaction().begin();  
31 manager.persist(E1);  
32 manager.getTransaction().commit();  
33
```

# Recuperando Objetos

```
10  
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22  
23 EntityManager manager = factory.createEntityManager();  
24  
25 Evento retornado = manager.find(Evento.class, 2L);  
26  
27 System.out.println(retornado.getDescricao());
```

# Removendo objetos

```
18  
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22  
23 EntityManager manager = factory.createEntityManager();  
24  
25  
26 Evento retornado = manager.find(Evento.class, 2L);  
27  
28 manager.getTransaction().begin();  
29 manager.remove(retornado);  
30 manager.getTransaction().commit();  
31
```

# Atualizando objeto persistido

```
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22  
23 EntityManager manager = factory.createEntityManager();  
24  
25 Evento Ela = new Evento();  
26 Ela.setId(2L);  
27 Ela.setFinalizado(true);  
28 Ela.setDescricao("Segundo teste da conexão com o banco - update");  
29 Ela.setDataFinalizacao(Calendar.getInstance());  
30  
31 manager.getTransaction().begin();  
32 manager.merge(Ela);  
33 manager.getTransaction().commit();
```

# Usando o recurso Query com parâmetros

1. JPA provém uma linguagem de consulta específica JPQL

```
18  
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22 EntityManager manager = factory.createEntityManager();  
23  
24 List<Evento> eventos = manager  
25     .createQuery("select e from Evento as e where e.finalizado = false")  
26     .getResultList();  
27
```

# Recuperando com where

```
19 EntityManagerFactory factory = Persistence.  
20     createEntityManagerFactory("Evento");  
21  
22 EntityManager manager = factory.createEntityManager();  
23  
24 Query q = manager.createQuery("select e from Evento as e "  
25     + "where e.finalizado = :paramFinalizado");  
26 q.setParameter("paramFinalizado", true);  
27
```