FIAP GRADUAÇÃO

FIAP

ENTERPRISE APPLICATION DEVELOPMENT

Prof. THIAGO T. I. YAMAMOTO

#02 – MAPEAMENTO OBJETO-RELACIONAL

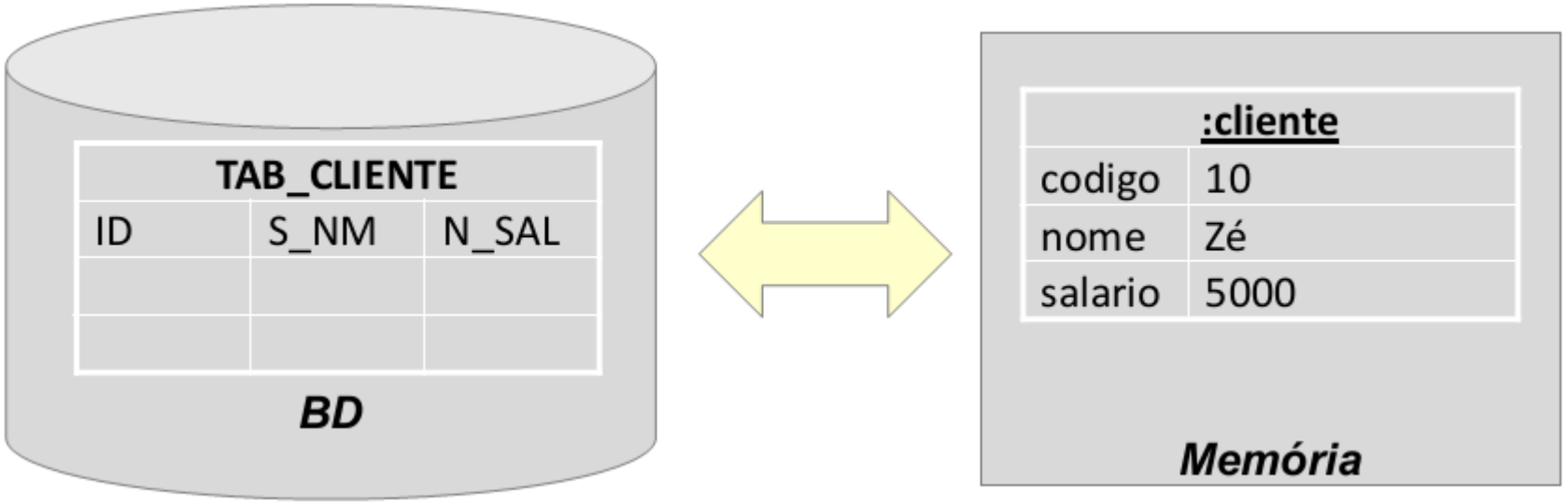
 thiagoyama  thiagoyama@gmail.com

AGENDA

FIAP

- Conceitos de Mapeamento ORM
- Anotações Java
- APIs de persistência
- Entidades
- Mapeamento simples de entidades

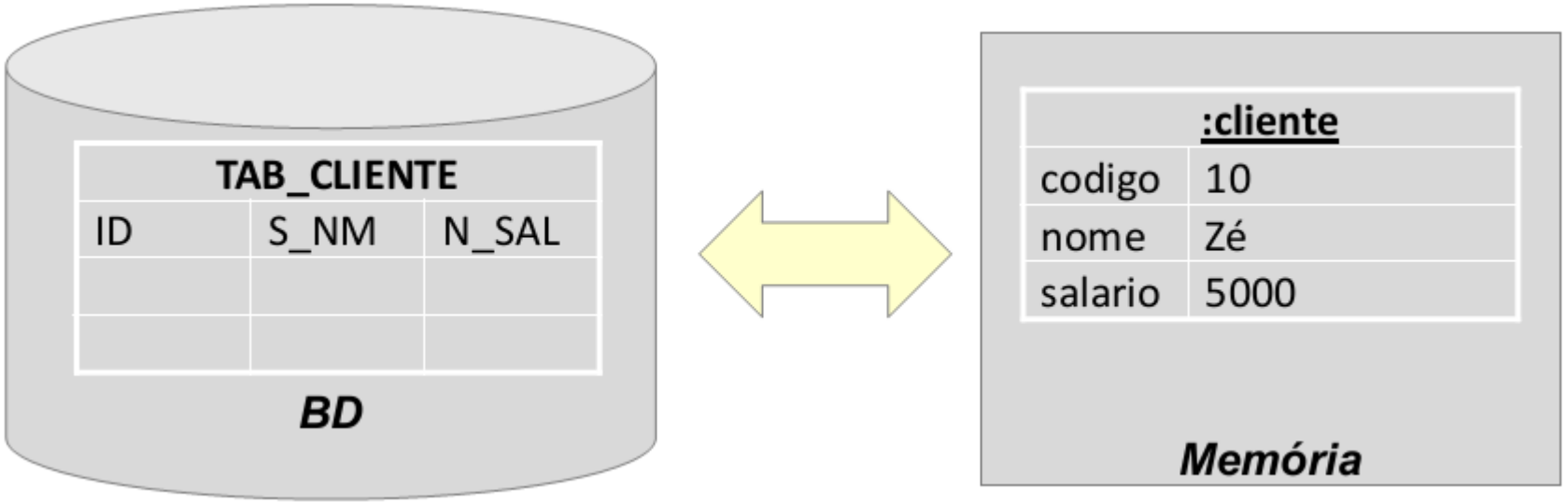
PERSISTÊNCIA




```
String sql = “INSERT INTO TAB_CLIENTE (ID, S_NM, N_SAL) VALUES (?, ?, ?)”;

PreparedStatement ps = conn.prepareStatement(sql);

ps.setInt(1, cliente.getCodigo());

ps.setString(2, cliente.getNome());

ps.setFloat(3, cliente.getSalario());

...
```

PERSISTÊNCIA

FIAP



Mapeamento:

Cliente -> TAB_CLIENTE

codigo -> ID

nome -> S_NM

salario -> N_SAL

Soluções:

- XML
- ANOTAÇÕES JAVA

FIAP

ANOTAÇÕES JAVA

ANOTAÇÕES JAVA

FIAP

- São textos inseridos diretamente no código fonte que expressam informações complementares sobre o uma classe, seus métodos, propriedades, etc...;
- Tais informações podem ser acessadas via API Reflection por elementos fora do código fonte, por exemplo, APIs de persistência;
- Disponíveis no Java 5 (JSR-175);
- Permitem especificar metadados dentro do código;
- Pode-se criar novas anotações a qualquer momento sendo um processo bastante simples;
- Alternativa aos descritores de deployment (XML);
- Não é possível alterar as anotações caso não se tenha o código fonte;
- Se utilizadas em excesso tendem a “poluir” o código fonte.

ANOTAÇÕES JAVA

FIAP

- Objetos são instanciados a partir de classes anotadas;
- Processador reconhece as anotações encapsuladas nos objeto;
- Os resultados são produzidos a partir das informações contidas nas anotações.

ANOTAÇÕES JAVA - SINTAXE

FIAP

- Podem ser inseridos antes da declaração de pacotes, classes, interfaces, métodos ou propriedades;
- Iniciam com um “@”;
- Uma anotação tem efeito sobre o próximo elemento na sequência de sua declaração;
- Mais de uma anotação pode ser aplicada a um mesmo elemento do código (classe, método, propriedade, etc...)
- Podem ter parâmetros na forma (param1=“valor”, param2=“valor”, ...);

Exemplo:

```
@Override
@SuppressWarnings("all")
public String toString() {
      return “Thiago";
}
```

ANOTAÇÕES JAVA NATIVAS

FIAP

Algumas anotações são nativas, isto é, já vem com o JDK:

@Override - Indica que o método anotado sobrescreve um método da superclasse;

@Deprecated - Indica que um método não deve mais ser utilizado;

@SuppressWarnings (tipoAlerta) - desativa os alertas onde tipoAlerta pode ser “all”, “ cast ”, “null”, etc...;

ANOTAÇÕES JAVA - CRIAÇÃO

FIAP

- Semelhante a declaração de uma interface (não define implementação);
- Métodos definem os parâmetros aceitos pela anotação;
- Parâmetros possuem tipos de dados restritos (String, Class, Enumeration, Annotation e Arrays desses tipos);
- Parâmetros podem ter valores default;

Exemplo Anotação:

```
public @interface Mensagem {
        String texto() default "vazio";
}
```

Exemplo Utilização:

```
@Mensagem(texto="Alo Classe")
public class Teste {
        @Mensagem(texto="Alo Metodo")
        public void teste() { }
}
```

ANOTAÇÕES JAVA – META ANOTAÇÕES

Anotações para se criar anotações:

@Retention indica onde e por quanto tempo anotações com este tipo serão mantidas:

- RetentionPolicy.SOURCE - Nível código fonte;
- RetentionPolicy.CLASS - Nível compilador;
- RetentionPolicy.RUNTIME - Nível JVM;

@Target indica o escopo da anotação:

- ElementType.PACKAGE - Pacote;
- ElementType.TYPE - Classe ou Interface;
- ElementType.CONSTRUCTOR - Método Construtor;
- ElementType.FIELD - Atributo;
- ElementType.METHOD - Método;
- ElementType.PARAMETER - Parâmetro de método;

ANOTAÇÕES JAVA - EXEMPLO

FIAP

Criação da anotação:

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE, ElementType.METHOD})
public @interface Mensagem {
        String texto() default "vazio";
}
```

Utilização:

```
@Mensagem(texto=“Alo Classe”)
public class Teste {
        @Mensagem(texto=“Alo Metodo”)
        public void teste() { }
}
```

ACESSANDO AS ANOTAÇÕES

FIAP

Acesso em tempo de execução via **API Reflection**;

- Para uma anotação de **classe**:

```
Mensagem m = obj.getClass().getAnnotation(Mensagem.class);
```

- Para uma anotação de **método**:

```
Method[] metodos = obj.getClass().getDeclaredMethods();
for (int i = 0; i < metodos.length; i++)
      System.out.println(metodos[i].getAnnotation(Mensagem.class));
```

- Para uma anotação de **propriedades**:

```
Field[] att = obj.getClass().getDeclaredFields();
for (int i = 0; i < att.length; i++)
      System.out.println(att[i].getAnnotation(Mensagem.class));
```

ANOTAÇÕES JAVA - PRÁTICA

FIAP

Escreva uma classe que tenha um método capaz de receber como parâmetro um objeto e gerar código SQL automaticamente capaz de selecionar todos os registro de uma tabela.

Criar uma anotação **Tabela** que possua um parâmetro **nome** indicando o nome da tabela na qual a classe será mapeada.

Via API Reflection gerar automaticamente o código SQL necessário.

Exemplo

```
@Tabela(nome="TAB_ALUNO")
public class Aluno {
}
```

Irá gerar o SQL (somente imprimir no console do eclipse):
SELECT * FROM TAB_ALUNO

APIs DE PERSISTÊNCIA

APIs DE PERSISTÊNCIA

FIAP

- Em um projeto de software real não é necessário que o desenvolvedor crie suas próprias anotações para a persistência de objetos;

- Existem APIs já prontas que lidam com o problema, por exemplo, a especificação JPA;

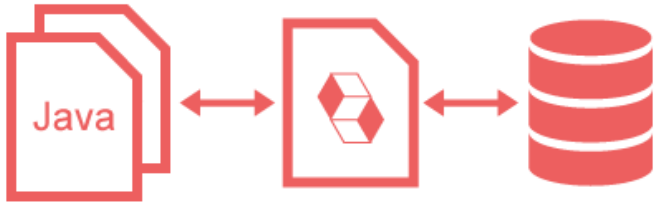
- Tais APIs são responsáveis, entre outras coisas, pela transformação dos objeto em declarações SQL (INSERT, UPDATE, DELETE, SELECT, ...).

JPA

FIAP

- Para evitar que cada API de persistência defina seu próprio conjunto de anotações uma especificação deve ser seguida;
- Este é um dos objetivos da JPA – Java Persistence API;
- Assim, a JPA oferece um padrão para mapeamento O/R por meio de anotações Java e uma API de serviços associados;

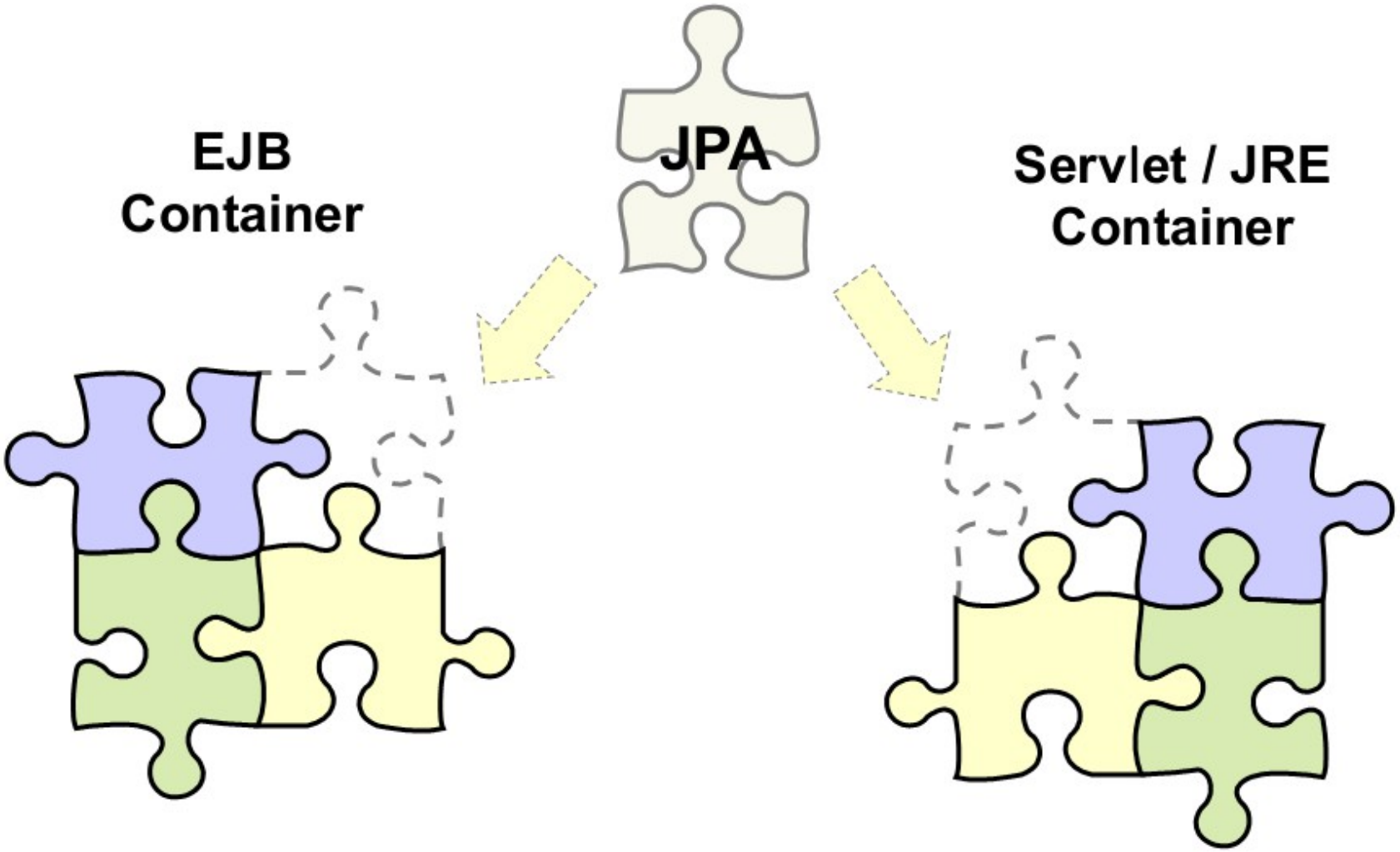
- Exemplos de implementações:
  - Hibernate
    - http://www.hibernate.org/
  - Toplink
    - http://www.oracle.com/technetwork/middleware/toplink/overview/index.html



- Dentro do ambiente EJB a JPA trata com a persistência (Entity Beans) mas pode ser utilizada de modo isolado (fora de um EJB container).

JPA – ONDE USAR?

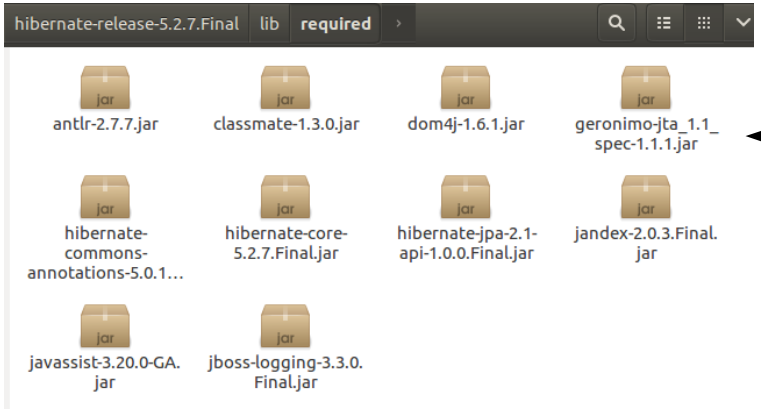
FIAP



JPA - CONFIGURAÇÃO

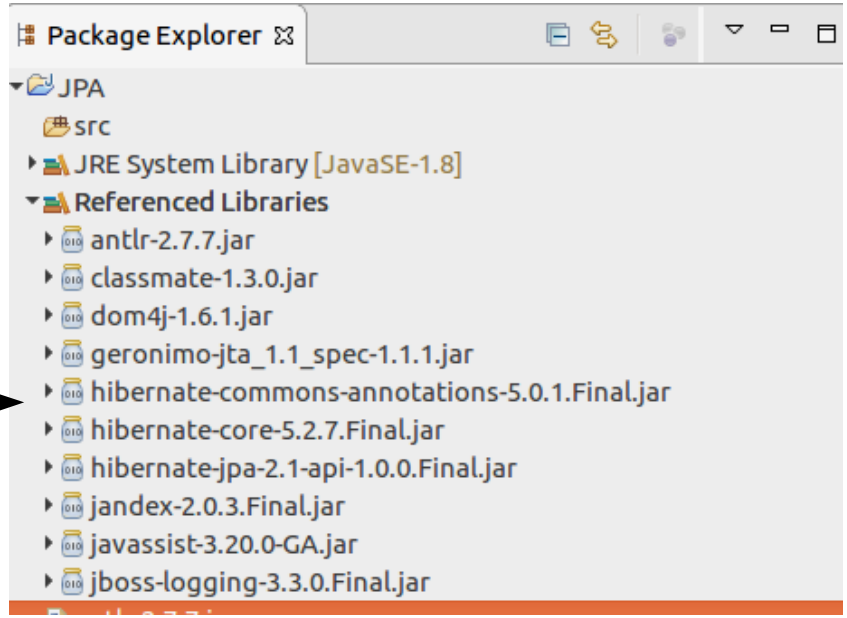
FIAP

Você encontra as bibliotecas do Hibernate / JPA no link: http://hibernate.org/orm/

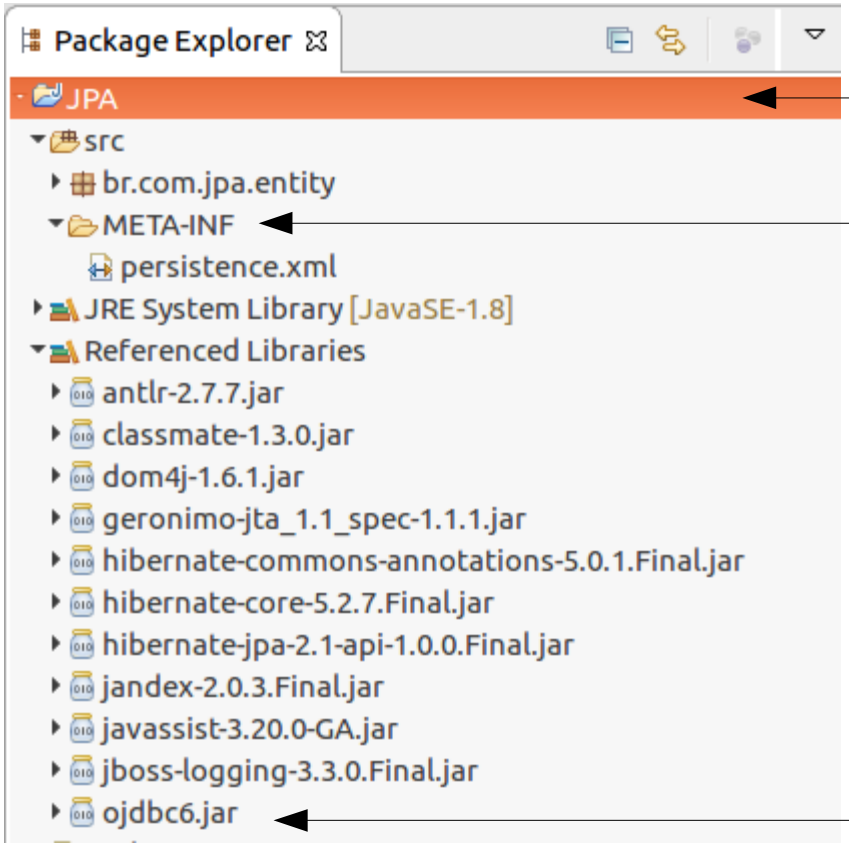


Ao descompactar o arquivo você encontra as bibliotecas no diretório lib

Para a utilização da JPA são necessárias as bibliotecas (jars) do diretório **lib/required.**



JPA - CONFIGURAÇÃO



Qualquer tipo de projeto pode utilizar a JPA

Dentro do **src** fica a pasta META-INF e o arquivo de configuração **persistence.xml**

Também é necessário o driver do banco de dados que será utilizado

JPA - ENTIDADES

FIAP

- Representam as unidades de persistência;
- Correspondem a classes simples (POJO) cujo estado pode ser persistido;
- Permitem o acesso aos dados por meio de métodos get e set;
- Possuem, obrigatoriamente, um identificador único;
- Recomendável que implementem a interface **Serializable**;
- Tais classes são mapeadas para o banco de dados por meio de anotações;
- São como espelhos do banco de dados, isto é, uma instância é criada ou alterada primeiramente em memória e posteriormente atualizada no banco de dados;
- São gerenciadas por um mecanismo de persistência denominado **Entity Manager**;

@ENTITY

FIAP

- As anotações da JPA situam-se no pacote **javax.persistence**;
- A anotação **@Entity** especifica que uma classe é uma entidade;
- O nome da tabela da entidade será o mesmo da classe com a anotação **@Entity.**

Exemplo

```
@Entity
public class Cliente {
    private int id;
    private String nome;
    // métodos get e set
}
```

@TABLE

FIAP

- Podemos alterar o nome da tabela assoociada a entidade através do atributo name da annotation **@Table.**

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class Cliente {
    private int id;
    private String nome;
    // métodos get e set
}
```

@ID

- Deve-se sempre definir o atributo que representará a chave primária;
- As únicas anotações obrigatórias para uma entidade são o **@Entity** e **@Id**;
- As anotações podem ser feitas tanto nas propriedades da classe quanto nos métodos **get**;

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class Cliente {

    @Id
    private int id;

    private String nome;

    // métodos get e set
}
```

@GENERATEDVALUE

FIAP

- Especifica a estratégia de geração de valores automáticos para propriedades;

- Normalmente utilizado em conjunto com o atributo anotado com **@Id**;

- Parâmetros:
  - **generator**: nome do gerador de chaves;
  - **strategy**: indica o tipo de estratégia utilizada;

- Tipos mais comuns de estratégias de geração:
  - **GeneratorType.SEQUENCE**: baseado em sequence;
  - **GeneratorType.IDENTITY**: campos identidade;

@SEQUENCEGENERATOR

FIAP

- Define um gerador de chave primária baseado em sequence de banco de dados;

- Possui uma associação com o **@GeneratedValue** definido com a estratégia **GenerationType.SEQUENCE**;

- Parâmetros:
  - **name:** nome a ser referenciado pelo @GeneratedValue;
  - **sequenceName:** nome da sequence de banco de dados;
  - **allocationSize** : incremento

@SEQUENCEGENERATOR

FIAP

Exemplo

```
@Entity
@SequeceGenerator(name="cliente",
sequenceName="SEQ_CLIENTE",allocationSize=1)
@Table(name="TAB_CLIENTE")
public class Cliente {
    @Id
    @GeneratedValue(strategy=GenerationType.SEQUENCE,generator="cliente")
    @Column(name="COD_CLIENTE")
    private int id;

    @Column(name="NOM_CLIENTE", nullable=false)
    private String nome;

    // métodos get e set

}
```

@COLUMN

FIAP

- Especifica o campo associada ao atributo da entidade;

- Caso não definido assume-se que o campo terá o mesmo nome do atributo;

- Alguns parâmetros:
  - **Name** - nome do campo;
  - **Nullable** (default true) - não permite valores nulos;
  - **Insertable** (default true) - atributo utilizado em operações de INSERT;
  - **Updatable** (default true) - atributo utilizado em operações de UPDATE;
  - **Length** (default 255) – atributo utilizado para definnir o tamanho do campo, aplicado somente para strings

@COLUMN

FIAP

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class Cliente {
    @Id
    @Column(name="COD_CLIENTE")
    private int id;

    @Column(name="NOM_CLIENTE", nullable=false)
    private String nome;

    // métodos get e set
}
```

@TRANSIENT

FIAP

- Indica que determinado atributo não deve ser persistido;

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class Cliente {

    @Id
    @Column(name="COD_CLIENTE")
    private int id;

    @Column(name="NOM_CLIENTE", nullable=false)
    private String nome;

    @Transient
    private int chaveAcesso;
}
```

@TEMPORAL

FIAP

- Especifica o tipo de dado a ser armazenado em propriedades do tipo **Date** e **Calendar**;
  - Parâmetros:
    - **TemporalType.TIMESTAMP**: data e hora;
    - **TemporalType.DATE:** somente data;
    - **TemporalType.TIME:** somente hora;

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class ClienteEntity {
    ...

    @Column(name="DAT_NASCIMENTO")
    @Temporal(value=TemporalType.DATE)
    private Calendar dataNascimento;

    // ... Métodos get / set
}
```

@LOB

- Permite mapear objetos de grande dimensão para a base de dados. Exemplo: imagens, documentos de texto, planilhas, etc...
- Os bancos de dados oferecem um tipo de dado para tais objetos. Exemplo: BLOB no Oracle;
- No objeto, o atributo mapeado normalmente é do tipo byte[] (array);

Exemplo

```
@Entity
@Table(name="TAB_NOTICIA")
public class NoticiaEntity {
    ...

    @Column(name="BLB_CONTEUDO")
    @Lob
    private byte[] conteudo;

    // ... Métodos get / set
}
```

ENUM – VALORES DE DOMÍNIO

FIAP

- Propriedades que possuem valores fixos, por exemplo, sexo (MASCULINO, FEMINIO), dia da semana (SEGUNDA, TERÇA, ...)
- Em Java podemos criar um **Enum** e utilizá-lo como tipo de dado do atributo de domínio;
- O índice associado ao valor depende da seqüência que foi declarado no Enum;

Exemplo

```
public enum Tipo {
    OURO, PRATA, BRONZE
}
```

Exemplo acima, OURO= 0, PRATA =1 e BRONZE = 2

Exemplo

```
@Entity
@Table(name="TAB_CLIENTE")
public class ClienteEntity {

    @Id
    @Column(name="COD_CLIENTE")
    private int id;

    @Column(name="IND_SEXO")
    private Sexo sexo;

    // ... Métodos get / set
}
```

FIAP

"Aquele que não luta pelo futuro que quer, deve aceitar o futuro que vier"