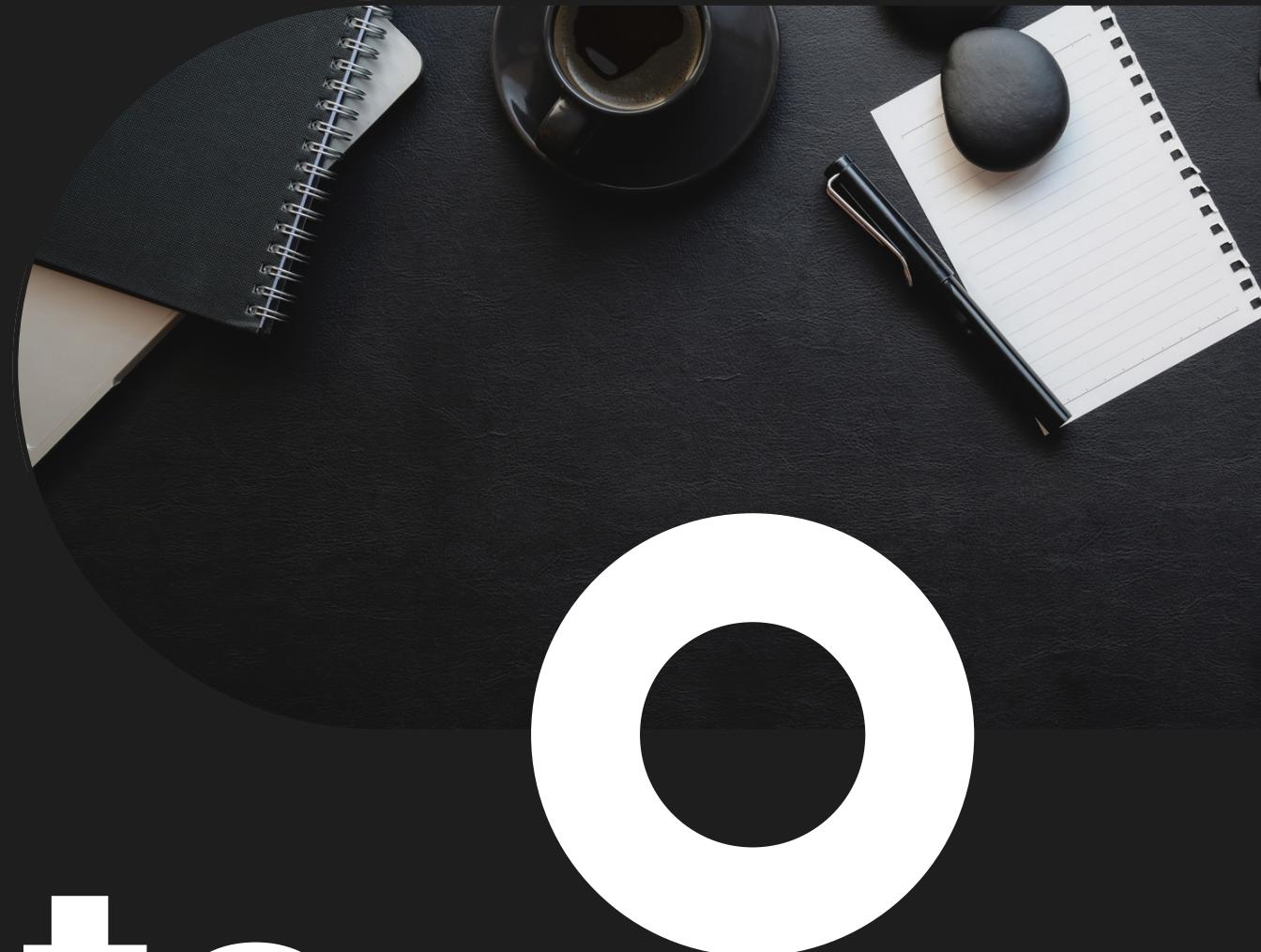
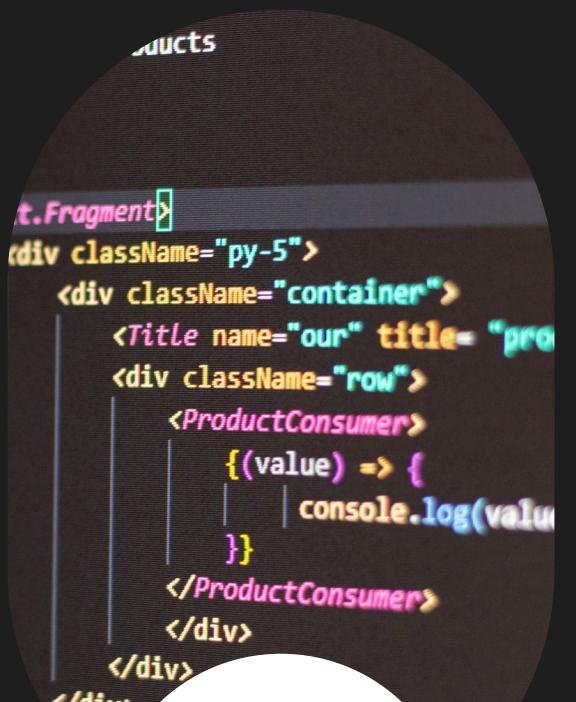


JPA

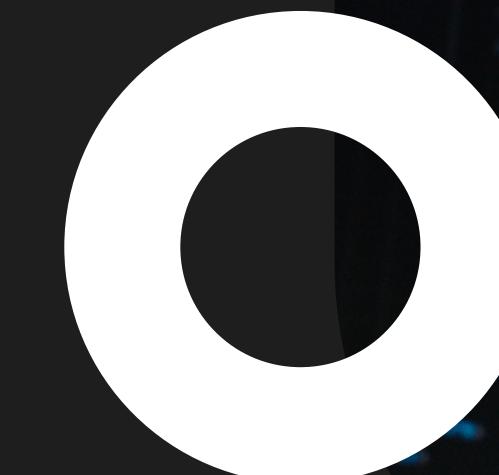
Hibernate

Conceitos Básicos

Prof. Juliane Correia



TODOList



```
6 .box{  
7   position: absolute;  
8   top: 50%;  
9   left: 50%;  
10  transform:  
11    translate(-50%, -50%);  
12  width: 400px;  
13  padding: 40px;  
14  background:  
15    radial-gradient(  
16      center, #fff 40px,  
17      transparent 40px,  
18      transparent 60px,  
19      #fff 60px);  
20  }  
21 .box h2{  
22   margin: 0 0 10px;  
23   padding: 0;  
24   color: #fff;  
25   text-align: center;  
26 }  
27 .box h3{  
28   margin: 0 0 10px;  
29   padding: 0;  
30   color: #fff;  
31   text-align: center;  
32 }  
33 .box .inputBox{  
34   position: relative;
```

JPA

ORM

O que é Hibernate

Como usar o Hibernate

Anotações

Relacionamentos

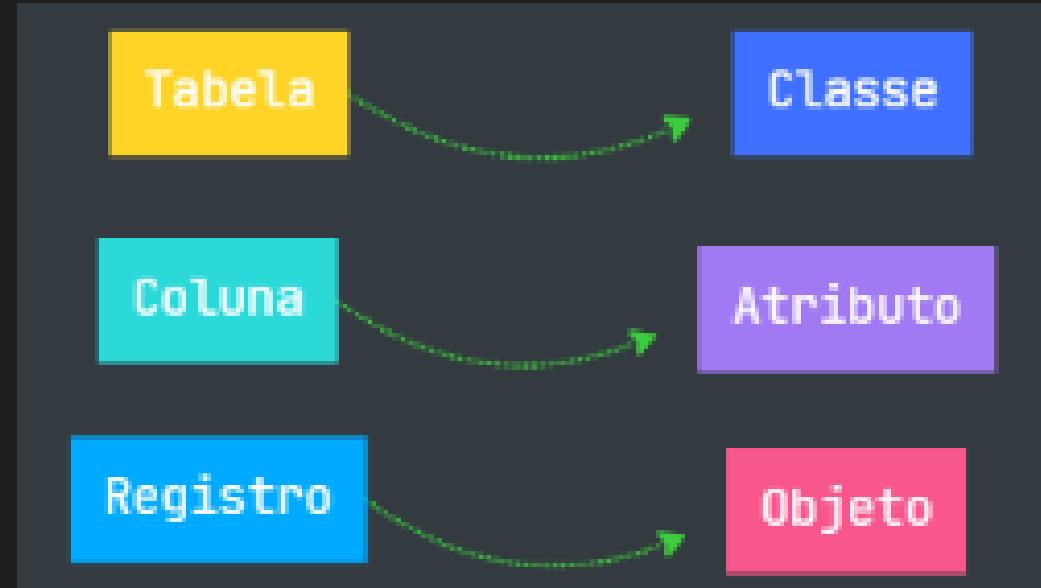
O QUE É JPA

Java Persistence API

- Java Persistence API, é uma API padrão da linguagem Java para mapeamento objeto-relacional (ORM);
- Ele fornece uma interface para persistir objetos Java em um banco de dados relacional usando ORM;
- O JPA é uma especificação e não uma implementação concreta, ou seja, ele define como as aplicações Java interagem com o banco de dados mas não fornece a implementação completa, sendo necessário escolher uma forma de implementação específica, como Hibernate;



O QUE É ORM



- ORM (Object-Relational Mapping) é uma técnica de programação que permite que os desenvolvedores de software manipulem dados armazenados em um banco de dados relacional usando objetos;
- Elimina a necessidade de escrever SQL e outros códigos de acesso a dados de forma manual;



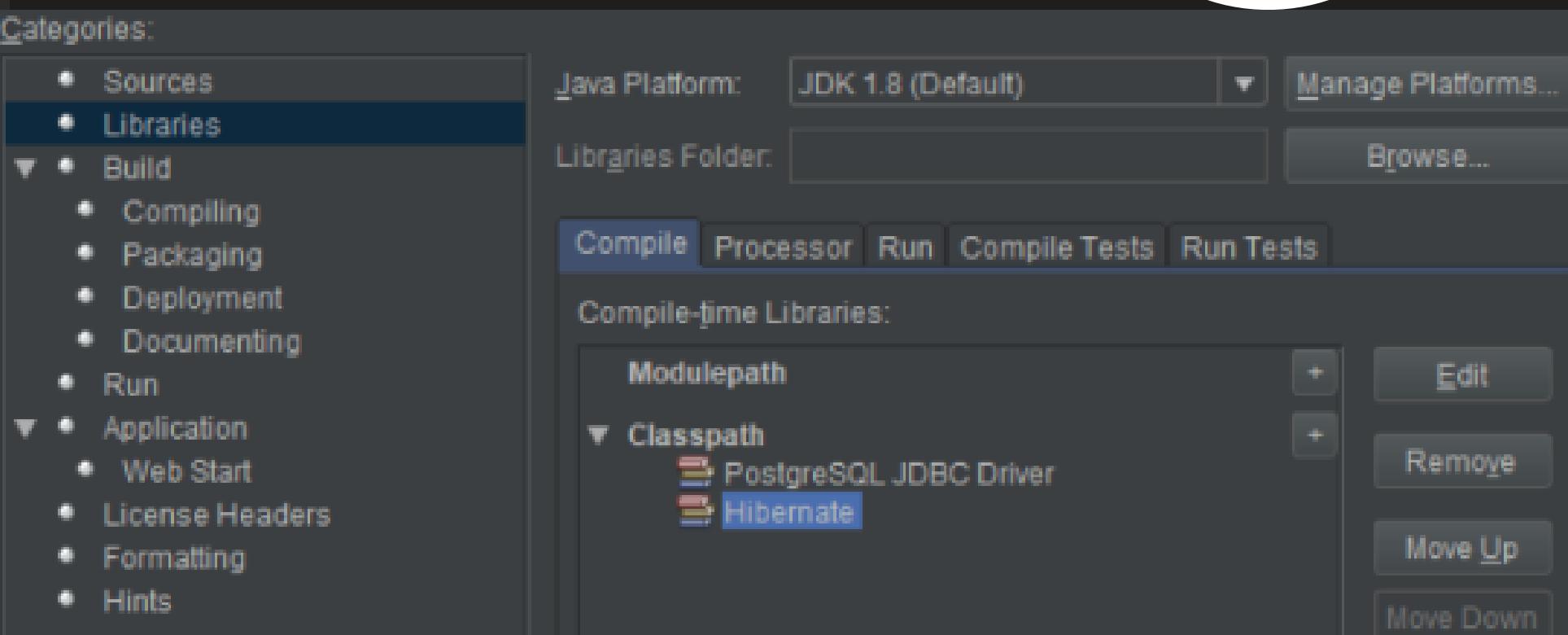
O QUE É Hibernate

- Hibernate é um framework ORM de código aberto para a linguagem de programação Java;
- Com o Hibernate, os desenvolvedores podem mapear objetos Java para tabelas de banco de dados, definir relacionamentos entre as entidades, escrever consultas usando HQL (Hibernate Query Language) e executar operações CRUD;

COMO USAR Hibernate

- Importar as dependências no pom.xml ou adicionar as bibliotecas manualmente ao classpath

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.15.Final</version>
</dependency>
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.6.15.Final</version>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.7.2</version>
</dependency>
```



COMO USAR Hibernate

- Configurar o arquivo persistence.xml.
- É importante lembrar que deve estar na pasta META-INF (src/main/resources).

```
<persistence-unit name="HibernateMaven">  
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>  
    <properties>  
        <property name="hibernate.show_sql" value="true" />  
        <property name="hibernate.format_sql" value="true" />  
        <property name="hibernate.hbm2ddl.auto" value="update" />  
        <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://localhost:5434/BASE"/>  
        <property name="javax.persistence.jdbc.user" value="postgres"/>  
        <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>  
        <property name="javax.persistence.jdbc.password" value="SENHA"/>  
    </properties>  
</persistence-unit>
```

identifica qual é a unidade de persistência

identifica qual é o provedor, no caso Hibernate

create: cria e sobrescreve
validate: verifica se está em conformidade
update: verifica e atualiza (melhor para ambiente dev)
none: não faz nada (melhor para ambiente produção)

Anotações

Principais anotações de mapeamento:



- @Entity**: indica que a classe é uma entidade que será mapeada para uma tabela no banco de dados;
- @Table**: permite especificar nome e atributos específicos da tabela;
- @Id**: indica que o atributo é uma chave primária da entidade;
- @GeneratedValue**: permite definir a estratégia de geração de valores para a chave primária (auto, identity(mais eficiente), sequence e table);
- @Column**: permite especificar suas propriedades como tipo, tamanho e restrições;

Anotações - Relacionamentos

@ManyToOne: define os relacionamentos da entidade muitos para um

@OneToMany: define os relacionamentos da entidade de um para muitos

@JoinColumn: utilizado para especificar a coluna de chave estrangeira

mappedBy: atributo do @JoinColumn utilizado para especificar o relacionamento bidirecional entre entidades

```
@Entity  
public class Endereco {  
  
    @ManyToOne  
    @JoinColumn(name = "id_cliente")  
    private Cliente cliente;
```

```
@Entity  
public class Cliente {  
  
    @OneToMany(mappedBy = "cliente", cascade = CascadeType.ALL,  
              orphanRemoval = true)  
    private List<Endereco> enderecos = new ArrayList<>();
```

