Laboratório de Programação de Web Sites Dinâmicos

JavaEE + JPA

Tassio Sirqueira – 2019/02

JavaEE + JPA | Introdução

- A integração com bancos de dados é um requisito comum para a maioria das aplicações.
- Com a popularização do Java no meio corporativo, observou-se o grande esforço dedicado à criação de SQL e manipulação JDBC.
- Além da produtividade foram observados outros problemas:
  - Diferença entre a sintaxe SQL de SGBDs diferentes.
  - Desafio no mapeamento objeto-relacional

JavaEE + JPA | Introdução

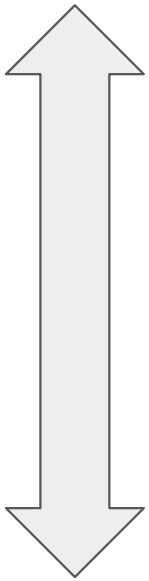
- Diferentes alternativas para solucionar esse problema surgiram.
- Entre elas o **Hibernate** se popularizou,se tornando, desde então, líder de mercado.
- O hibernate é um framework ORM (Mapeamento Objeto Relacional), e busca resolver o que conhecemos como **impedância objeto** (representação seguindo princípios OO) **relacional** (representação normalizada).

JavaEE + JPA | Introdução

- Por sua popularidade e forte presença em diversos projetos Java, uma proposta de especificação de um framework inspirado no Hibernate foi proposto.
- O JPA, **Java Persistence API,** define uma série de classes, interfaces e padrões para a criação de frameworks ORM.
- O Hibernate é um dos frameworks que implementam o padrão JPA, assim como o **EclipseLink (referência)** e o OpenJPA.

JavaEE + JPA | Arquitetura

Modelo Orientado à Objetos



Modelo Relacional

Servidor de Aplicação
(Glassfish, Wildfly...)

WEB
(Servlets, JSP, JSF)

Aplicação Java

API JPA

Implementação JPA
(Hibernate, EclipseLink, OpenJPA)

API JDBC

Implementação do Driver JDBC

Banco de Dados Relacional
(SQL Server, Oracle, Postgres, MySQL, Derby)

JavaEE + JPA | Revisão de JPA

- Para utilizar o JPA, o primeiro passo é definir uma unidade de persistência (persistent unit).
- Deve ser definida no arquivo persistence.xml
  - Que deve ser estar na pasara META-INF, na raiz do classpath.
    - src/META-INF/persistence.xml
    - src/main/resources/META-INF/persistence.xml **(maven)**
- Uma unidade de persistência define:
  - As informações de conexão JDBC com o banco (URL, Driver...)
  - As informações necessárias para a implementação (Hibernate, EclipseLink...)

JavaEE + JPA | Revisão de JPA | Exemplo de persistence.xml

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.1"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
 <persistence-unit name="estacionamento_unit"  transaction-type="RESOURCE_LOCAL">
   <description> Hibernate JPA Estacionamento</description>
   <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
   <properties>
     <property name="javax.persistence.jdbc.driver" value="org.apache.derby.j
     <property name="javax.persistence.jdbc.url" value="jdbc:derby:estacionam
     <property name="javax.persistence.jdbc.user" value="root"/>
     <property name="javax.persistence.jdbc.password" value
     <property name="hibernate.show_sql" value="
     <property name="hibernate.format_sql" value
     <property name="hibernate.hbm2ddl.auto" val
   </properties>
 </persistence-unit>
</persistence>
```

Definição da unidade de persistência. Uma aplicação java pode ter várias unidades de persistência.

Definição da implementação do JPA que deve ser utilizada nessa unidade de persistência. Nesse caso, a classe que define que utilizaremos o Hibernate é: **org.hibernate.jpa.HibernatePersistenceProvider**

JavaEE + JPA | Revisão de JPA | Exemplo de persistence.xml

```xml
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.1"
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
 <persistence-unit name="estacionamento_unit"  transaction-type="JTA">
   <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
   <properties>
     <property name="javax.persistence.schema-generation.database.action" value="rop-and-create"/>
   </properties>
 </persistence-unit>
</persistence>
```

Utilizando JPA a partir de um Servidor de Aplicações permite que as conexões e configurações de persistência sejam controladas por ele.

Assim como a escolha da implementação.

JavaEE + JPA | Revisão de JPA | Unidade de Persistência

- A partir da **Unidade de Persistência** (***Persistence Unit***), acessamos outras classes essenciais para a JPA.
- A definição de unidades de persistência desacopla detalhes específicos da implementação das definições da especificação JPA.
- Se apenas o padrão da API for utilizado, é possível substituir o provider sem impactar outras partes da aplicação.
  - Na prática é muito difícil de fazer...

JavaEE + JPA | Revisão de JPA | Entidade

- Uma **entidade (*Entity*)** é um objeto que pode ser persistido no banco de dados.
- Essas classes possuem anotações especiais indicando c**omo devem ser persistidas** em um banco de dados relacional.
  - A partir dessas informações o JPA sabe como fazer o mapeamento objeto relacional entre as tabelas e colunas do banco de dados e os objetos e atributos em Java.
- Entidades estão relacionados à uma linha de uma tabela no banco de dados.
  - Devem possuir identificadores únicos, e existem independentemente dos valores de seus atributos.

JavaEE + JPA | Revisão de JPA | Exemplo de Entidade

- Definidos basicamente através da anotação @Entity e com uma coluna definida como @Id.

```
@Entity(name = "Product")
public class Product {

    @Id
    private Integer id;

    private String sku;

    private String name;

    private String description;

    // getters e setters omitidos.
}
```

JavaEE + JPA | Revisão de JPA | EntityManager

- Uma vez configurada, o acesso à **API** é feita pelo **EntityManager**.
  - Persistir/Atualizar objetos.
  - Recuperar uma entidade pelo tipo e pelo id.
  - Realizar uma consulta utilizando **JPQL (Java Persistence Query Language)** ou **SQL.**
  - Remover objetos.
- Operações feitas sobre objetos pelo **EntityManager** são automaticamente refletidas no banco de dados quando a transação é 'commitada'.

JavaEE + JPA | Revisão de JPA | EntityManager

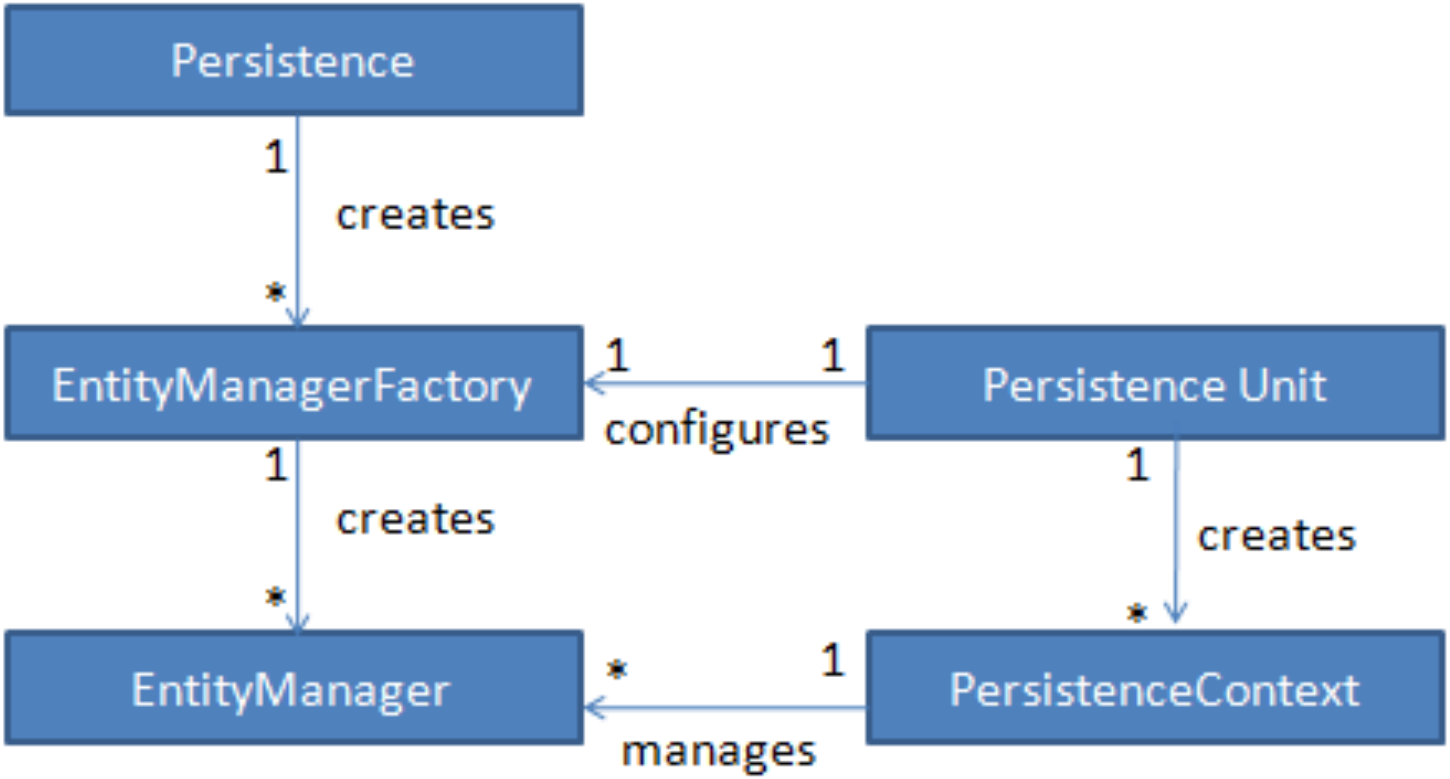
- Uma vez configurada, o acesso à **API** é feita pelo **EntityManager**.
  - Persistir/Atualizar objetos.
  - Recuperar uma entidade pelo tipo e pelo id.
  - Realizar uma consulta utilizando **JPQL (Java Persistence Query Language)** ou **SQL.**
  - Remover objetos.
- Operações feitas sobre objetos pelo **EntityManager** são automaticamente refletidas no banco de dados quando a transação é 'commitada'.

JavaEE + JPA | Revisão de JPA | Exemplo EntityManager

```java
Person person = new Person();
entityManager.persist( person );

Phone phone = new Phone( "123-456-7890" );
phone.setPerson( person );
entityManager.persist( phone );

entityManager.flush();
phone.setPerson( null );
```

```sql
INSERT INTO Person ( id )
VALUES ( 1 )

INSERT INTO Phone ( number, person_id, id )
VALUES ( '123-456-7890', 1, 2 )

UPDATE Phone
SET    number = '123-456-7890',
       person_id = NULL
WHERE  id = 2
```

JavaEE + JPA | Revisão de JPA



JavaEE + JPA | Utilizando JPA no JavaEE

- Em um ambiente de aplicação gerenciado não é necessário instanciar e configurar o JPA manualmente.
  - O próprio servidor de aplicação configura o JPA.
- Podemos obter os objetos configurados do **contexto** da aplicação.
- Para isso é utilizado o padrão **Injeção de Dependência** por meio de **Inversão de Controle.**

JavaEE + JPA | Utilizando JPA no JavaEE | Inversão de Controle

- Quando uma classe instância diretamente suas dependências, com o operador **new**, introduz alto acoplamento.
  - Conhece e depende de uma implementação concreta.
  - Não está preso apenas à interface, e sim à implementação.
  - Precisa conhecer os detalhes da classe para instanciá-la.
  - Precisa gerenciar o ciclo de vida dessa classe.
- Quando uma classe recebe no construtor, métodos ou diretamente nos atributos suas dependências, há inversão de controle.
  - Baixo acoplamento, depende apenas da interface da classe.
  - Não conhece detalhes de implementação / instanciação.
  - Ciclo de vida independente.

JavaEE + JPA | Utilizando JPA no JavaEE | Inversão de Controle

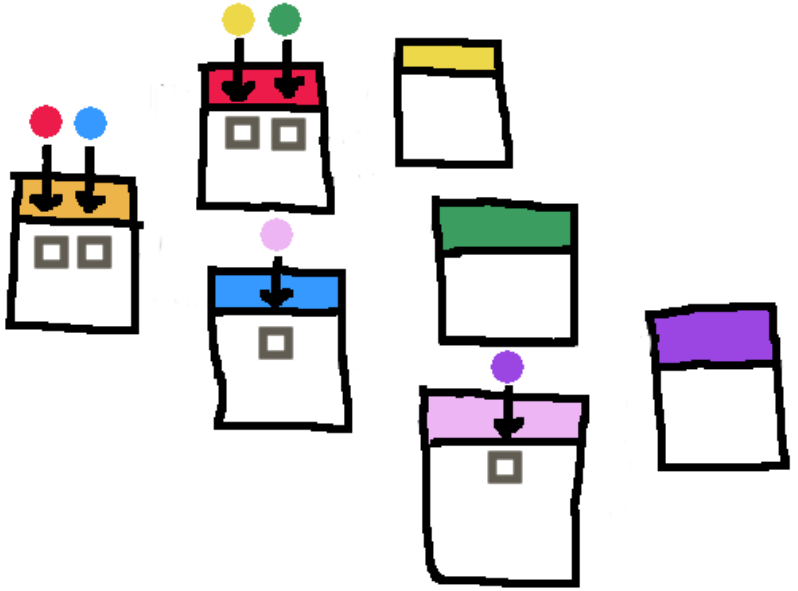
```java
class Relatorio {

   private List<Dado> dados;

   public Relatorio(List<Dado> dados){
       this.dados = dados;
   }

   public void gerar() {

       try (OutputStream out =
         new BufferedOutputStream(
            new FileOutputStream(OUTPUT_FILE),1024){
           out.write(bytes);
           for (Dado dado : dados){
               out.write(dado.getBytes());
           }
           out.flush();
       } catch (IOException e) {
           throw new RuntimeException(e);
       }
   }
}
```

```java
class Relatorio {

   private List<Dado> dados;

   public Relatorio(List<Dado> dados){
       this.dados = dados;
   }

   public void gerar(OutputStream out) {

       try {
           for (Dado dado : dados){
               out.write(dado.getBytes());
           }
           out.flush();


       } catch (IOException e) {
           throw new RuntimeException(e);
       }
   }
}
```

JavaEE + JPA | Utilizando JPA no JEE | Injeção de Dependência

- Em um ambiente não gerenciado, temos controle direto dos objetos instanciados

- Em um ambiente gerenciado, as dependências são instanciadas e seu ciclo de vida controlado por um contexto.



JavaEE + JPA | Utilizando JPA no JavaEE | Inversão de Controle

```
class Cofre {

   private ControleAcess acesso;

   public Relatorio(){
       SQLControleAcesso sql =
                        new SQLControleAcesso();
       sql.conecta();
       this.acesso = sql;
   }

   public boolean podeAcessar(Credencial cred) {
      return acesso.autoriza(cred);
   }
}
```

```
class Cofre {


   @Inject
   private ControleAcess acesso;


   public boolean podeAcessar(Credencial cred) {
      return acesso.autoriza(cred);
   }
}
```

JavaEE + JPA | Utilizando JPA no JavaEE

- Entre as tecnologias JEE, temos uma especificação para injeção de dependência, o **CDI**.
  - Não vamos estudar neste curso.
- Vamos utilizar conceitos de injeção em diversas especificações dentro da plataforma.

- Vamos alterar o projeto **todo list** para salvar as tarefas no banco de dados.
  - https://drive.google.com/open?id=0B0c1aylmwuLUUF9qZ2lUUHFxaXM
- Versão Alterada:
  - https://drive.google.com/open?id=0B0c1aylmwuLUcW16TjFfZkxaeW8

NetBeans

JavaEE + JPA | ATIVIDADE

- Utilizando o script https://drive.google.com/file/d/0B0c1aylmwuLUTnBsaHhxUGU4Nm8/view?usp=sharing
- Crie entidades, DAO e salve as avaliações no Banco de Dados.
- Envie para tassio@tassio.eti.br
  com o assunto Atividade 8 - PhotArt JPA
  até .