Argentina programa 4.0

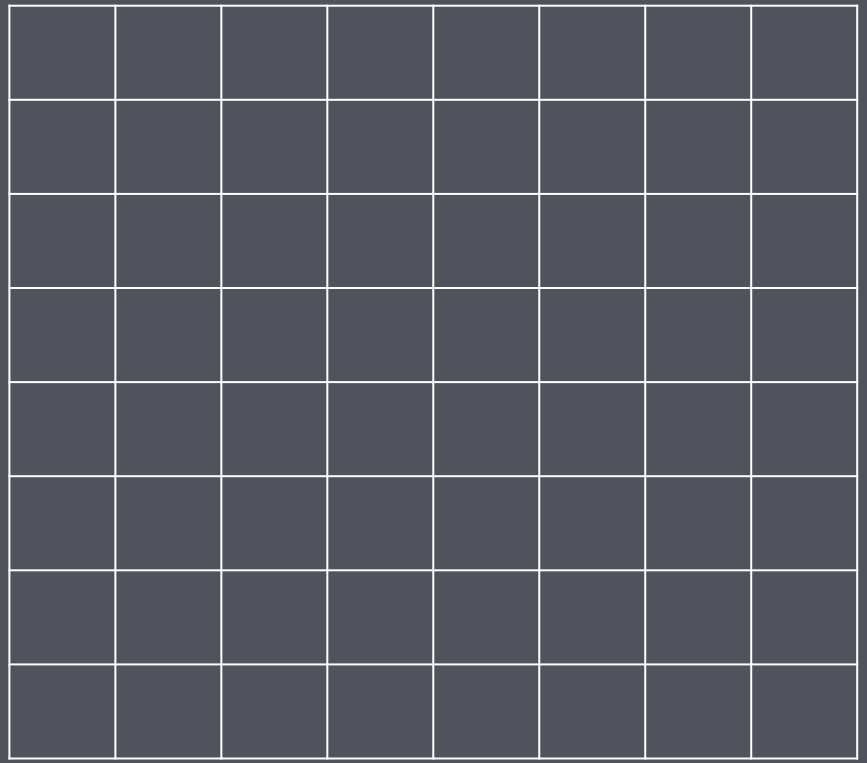
UNIVERSIDAD TECNOLOGICA NACIONAL

ORM - I

Etapa 2 - Desarrollador Java Intermedio

ORM - I

- Impedance Mismatches - Parte I
- Mapeo - Parte I



Estrategias de Persistencia

Argentina programa 4.0

Persistencia en Bases de Datos Relacionales

- El estado de un Sistema puede ser persistido en una, o varias, Base de Datos Relacional.
- La persistencia de un Sistema que está pensado y escrito bajo el Paradigma Orientado a Objetos, en una Base de Datos Relacional no es directa.

Mapeo Objeto - Relacional

Argentina programa 4.0

El Mapeo Objeto-Relacional es un técnica usada para convertir los tipos de datos con los que trabaja un lenguaje orientado a objetos a tipos de dato con los que trabaja un sistema de base de datos relacional.

Mientras que ***ORM*** es el nombre de la técnica, también se suele conocer como ORMs a los frameworks que la implementan.

Mapeo Objeto - Relacional

Argentina programa 4.0

Estas herramientas introducen una capa de abstracción entre la base de datos y el desarrollador, lo que evita que el desarrollador tenga que escribir consultas “a mano” para recuperar, insertar, actualizar o eliminar datos en la base.

Mapeo Objeto - Relacional

Argentina programa 4.0

Al tratar de encontrar una correspondencia entre estos “mundos”, sucede que existen características que son difíciles de imitar.

A este “desajuste” se lo conoce como ***Impedance Mismatch***

Impedance Mismatch - Parte I

Argentina programa 4.0

Identidad

- Un objeto cumple con la característica de Unicidad.

- Un registro, en una tabla de una base de datos relacional, necesita una identificación unívoca.

Impedance Mismatch - Parte I

Argentina programa 4.0

Identidad

Las claves posibles para una entidad en una BDR son:

- *Clave natural*: algún campo propio de la entidad que lo identifique de forma unívoca. Por ejemplo, CUIT del cliente, número de teléfono del abonado, etc.

- *Clave subrogada*: clave ficticia (al azar, generada automáticamente, etc.).

Impedance Mismatch - Parte I

Argentina programa 4.0

Conversión de Tipos de Datos

Los tipos de datos de ambos “mundos” no tienen una correlación directa.

- ***String*** sin limitaciones de tamaño vs. el VARCHAR/CHAR que requiere definirle longitud.
- ***Campos numéricos*** respecto a la precisión de decimales, o el rango de valores máximos y mínimos; incluso cuando el dominio es “un número de 0 a 3000”
- Las ***fechas*** tienen tipos no siempre compatibles entre sí: LocalDate de Java vs. Date-Datetime-Time-tinyblob del motor.
- ***Booleanos***
- ***Enumerados*** (como los enums de Java)

Impedance Mismatch - Parte I

Argentina programa 4.0

Cardinalidad

En el POO, las relaciones pueden ser unidireccionales y bidireccionales; con navegabilidad bidireccional.

Las relaciones entre las entidades de una base de datos relacional no son bidireccionales (salvo algunos casos).

JPA - Persistencia en Java

Argentina programa 4.0

“Java™ Persistence API (JPA) proporciona un mecanismo para gestionar la persistencia.”

“JPA crea un estándar para la importante tarea de la correlación relacional de objetos mediante la utilización de anotaciones (annotations) o XML para relacionar objetos con una o más tablas de una base de datos.”

JPA - Persistencia en Java

Argentina programa 4.0

- *La API **EntityManager** puede actualizar, recuperar, eliminar o aplicar la persistencia de objetos de una base de datos.*

- *La API **EntityManager** y los **metadatos** de correlación relacional de objetos (annotations) manejan la mayor parte de las operaciones de base de datos sin que sea necesario escribir código JDBC o SQL para mantener la persistencia.*

- *JPA proporciona un lenguaje de consulta, conocido también como JPQL, el cual se puede utilizar para recuperar objetos sin generar consultas SQL específicas en la base de datos con la que se está trabajando.*

Hibernate como ORM

Argentina programa 4.0

Hibernate es un ORM para Java que implementa JPA.

A continuación, comenzaremos a ver algunas anotaciones importantes para poder mapear nuestro dominio de Objetos.

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@Entity

Especifica que la clase es una entidad. Esta anotación se puede aplicar en Clases.
IMPORTANTE: una entidad NO necesariamente es una tabla en el modelo de datos.

```
import javax.persistence.Entity;

@Entity
public class Empleado {
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@Table

Especifica la tabla en la base de datos con la que se mapea esta entidad. En el siguiente ejemplo, los datos se almacenarán en la tabla "empleado". El atributo *name* se usa para especificar el nombre de la tabla.

```
import javax.persistence.*;

@Entity
@Table(name=”empleado”)
public class Empleado {
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@Column

Cada atributo acompañado de esta anotación se transformará en una columna en la DB. El atributo de *name* de esta anotación se usa para especificar el nombre de la columna de la tabla.

```
import javax.persistence.*;

@Entity
@Table(name=”empleado”)
public class Empleado {
  @Column(name=”nombre”)
  private String nombre;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@Id

Esta anotación especifica la clave principal (PK) de la entidad.

```
import javax.persistence.*;

@Entity
@Table(name=”empleado”)
public class Empleado {
  @Id
  @Column
  private int id;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@GeneratedValue

Esta anotación especifica las estrategias de generación de los valores de las claves primarias (claves subrogadas).

```
import javax.persistence.*;

@Entity
@Table(name=”empleado”)
public class Empleado {
  @Id
  @Column(name=”id”)
  @GeneratedValue(strategy=SEQUENCE, generator="ID_SEQ")
  private int id;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToOne

- Sirve para especificar una relación de tipo 1 - 1 entre dos entidades.
- La relación puede ser unidireccional o bidireccional a nivel objetos (también a nivel tablas, *pero no interesa esto por el momento*).
- La entidad (clase) que posea la FK a nivel tablas debe acompañar la anotación OneToOne con la anotación ***@JoinColumn***
- Si la relación es bidireccional a nivel objetos, la entidad que no posee la annotation @JoinColumn debe llevar el atributo "***mappedBy***" (dentro de OneToOne), cuyo valor debe ser el nombre del atributo en la otra entidad.

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToOne y @JoinColumn

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @OneToOne
  @JoinColumn(name=”direccion_id”, referencedColumnName=”id”)
  private Direccion direccion;
}

@Entity
@Table(name=”direccion”)
public class Direccion {
  @OneToOne(mappedBy=”direccion”)
  private Empleado empleado;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToOne

- Sirve para especificar una relación de tipo ***N - 1 (Muchos a Uno)*** entre dos entidades.

- No es necesario acompañar a esta anotación con *@JoinColumn*, ya que por defecto genera una columna con FK a la otra entidad; pero se puede utilizar (¡y es recomendable!).

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToOne

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @ManyToOne
  @JoinColumn(name=”sector_id”, referencedColumnName=”id”)
  private Sector sector;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToMany

- Sirve para especificar una relación de tipo ***1 - N (Uno a Muchos)*** entre dos entidades.

- Recordar que OneToMany es la relación inversa a ManyToOne (la relación ManyToOne vista "*del otro lado*").

- Esta anotación puede utilizarse sola (*unidireccional desde la entidad A que posee "la colección de Bs"*) o en conjunto con ManyToOne (*A posee "la colección de Bs" y B posee como atributo a A*).

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToMany

- Si se utiliza unidireccionalmente desde el lado A (donde está la colección), debe estar acompañada con *@JoinColumn* para definir la FK que la entidad B tendrá hacia la entidad A.

- Si se utiliza de forma bidireccional, el lado B (*donde está la anotación @ManyToOne*) debe definir la *@JoinColumn*. Además, el lado A (*donde está la anotación @OneToMany*) debe llevar el atributo "*mappedBy*", cuyo valor será el nombre del atributo en la entidad B que apunta hacia A.

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToMany *(unidireccional)*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @OneToMany
  @JoinColumn(name=”empleado_id”, referencedColumnName=”id”)
  private List<Fichaje> fichajes;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToMany *(bidireccional)*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @OneToMany
  private List<Fichaje> fichajes;
}

@Entity
@Table(name=”fichaje”)
public class Fichaje {
  @ManyToOne
  @JoinColumn(name=”empleado_id”, referencedColumnName=”id”)
  private Empleado empleado;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToMany

- Se utiliza para especificar una relación ***N - M (muchos a muchos***) entre dos entidades,
- La relación a nivel objetos puede ser unidireccional (de cualquier lado) o bidireccional.
- Si es bidireccional, uno de los lados debe acompañar la annotation @ManyToMany con el atributo "*mappedBy*".
- Recordar que este tipo de relación genera una "tabla intermedia" en la DB.
- La annotation se puede utilizar sola o se puede utilizar en conjunto con *@JoinTable*, la cual permite "personalizar" la creación de la tabla intermeida,

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToMany *(unidireccional)*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @ManyToMany
  private List<Rol> roles;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToMany *(unidireccional con @JoinTable)*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @ManyToMany
  @JoinTable(
   name = "rol_empleado",
   joinColumns = @JoinColumn(name = "empleado_id"),
   inverseJoinColumns = @JoinColumn(name = "rol_id"))
  )
  private List<Rol> roles;
}
```

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@ManyToMany *(bidireccional)*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @ManyToMany
  private List<Rol> roles;
}

@Entity
@Table(name=”rol”)
public class Rol {
  @ManyToMany(mappedBy=”roles”)
  private List<Empleado> empleados;
}
```

Mapeo Objeto Relacional - *Annotations* - *Cascadas*

Argentina programa 4.0

Las relaciones de entidad a menudo dependen de la existencia de otra entidad, por ejemplo, la relación Empleado – Dirección . Sin Empleado , la entidad Dirección no tiene ningún significado propio. Cuando eliminamos la entidad Empleado , nuestra entidad Dirección también debería eliminarse.

Cascada es la manera de lograr esto: ***cuando realizamos alguna acción en la entidad de destino, la misma acción se aplicará a la entidad asociada***.

Mapeo Objeto Relacional - *Annotations* - *Cascadas*

Argentina programa 4.0

Existen varios tipos de Cascadas (no todos disponibles en todos los ORMs):

- `ALL`
- `PERSIST`
- `MERGE`
- `REMOVE`
- `REFRESH`
- `DETACH`

Cada tipo de cascada propagará la(s) operación(es) correspondiente(s) involucrada del EntityManager.

Mapeo Objeto Relacional - *Annotations* - *Cascadas*

Argentina programa 4.0

- Se suele utilizar las cascadas en las relaciones @OneToOne, @ManyToOne o @ManyToMany, por comodidad.

- "*cascade*" es el atributo que se puede especificar dentro de las annotations mencionadas anteriormente, cuyo valor deberá ser uno de los enumerados de CascadeType.

Mapeo Objeto Relacional - *Annotations* - *Cascadas*

Argentina programa 4.0

El entityManager es el objeto encargado de la persistencia de las entidades.

Él podrá realizar operaciones de:

- *Persist (Inserción)*
- *Merge (Actualización)*
- *Remove (Eliminación)*
- *Find (Búsqueda)*
- *Otros...*

Mapeo Objeto Relacional - *Annotations*

Argentina programa 4.0

@OneToMany *(unidireccional) con cascada*

```
@Entity
@Table(name=”empleado”)
public class Empleado {
  @OneToMany(cascade = CascadeType.ALL)
  @JoinColumn(name=”empleado_id”, referencedColumnName=”id”)
  private List<Fichaje> fichajes;
}
```

Mapeo Objeto Relacional - *EntityManager*

Argentina programa 4.0

El entity manager tiene dos responsabilidad fundamentales:

- *Define una conexión transaccional con la base de datos que debemos abrir y mantener abierta mientras estamos realizando operaciones. En este sentido realiza funciones similares a las de una conexión JDBC.*

- *Además, mantiene en memoria una caché con las entidades que gestiona y es responsable de sincronizarlas correctamente con la base de datos cuando se realiza un flush.*

Mapeo Objeto Relacional - *EntityManager*

Argentina programa 4.0

- El conjunto de entidades que gestiona un entityManager se denomina su ***contexto de persistencia***.

- El entityManager se obtiene a través de una fábrica del tipo ***EntityManagerFactory***, que se configura mediante la especificación de una unidad de persistencia (“*persistence unit*”) definida en el fichero ***persistence.xml***

Mapeo Objeto Relacional - *EntityManager* - *persistence.xml*

Argentina programa 4.0

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="JPA_PU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/JPATutorial?zeroDateTimeBehavior=convertToNull&amp;server
Timezone=UTC"/>
      <property name="javax.persistence.jdbc.user" value="usuarioDB"/>
      <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.password" value="contraseniaDB"/>
      <property name="javax.persistence.schema-generation.database.action" value="create"/>
    </properties>
  </persistence-unit>
</persistence>
```

Mapeo Objeto Relacional - *EntityManager* - *persistence.xml*

Argentina programa 4.0

- ***javax.persistence.jdbc.url***: colocar el String de conexión a la DB.
- ***javax.persistence.jdbc.user***: colocar el nombre de usuario de la DB.
- ***javax.persistence.jdbc.password***: colocar la contraseña de la DB.
- ***javax.persistence.jdbc.driver***: configurar el driver con el cual nos conectaremos a la DB.

Mapeo Objeto Relacional - *EntityManager* - *persistence.xml*

Argentina programa 4.0

- ***javax.persistence.schema-generation.database.action***: puede colocarse uno de los siguientes valores:
  - ***create***: crea el esquema destruyendo todos los datos previos existentes.
  - ***create-drop***: mismo caso que el anterior pero, además, cuando el programa finaliza destruye el esquema.
  - ***validate***: valida el esquema pero no impacta los cambios en la DB.
  - ***none***: no impacta cambios del esquema en la DB.

Mapeo Objeto Relacional - *EntityManager* - *Uso básico*

Argentina programa 4.0

```
public class MainExample {

    public static EntityManager getEntityManager(){
        EntityManagerFactory factory = Persistence.createEntityManagerFactory("JPA_PU");
        EntityManager manager = factory.createEntityManager();
        return manager;
    }

    public static void main(String[] args) {
        EntityManager em = getEntityManager();
        EntityTransaction tx = em.getTransaction();
        tx.begin();
        Empleado empleado = new Empleado(); //← Seteamos todos los atributos
        em.persist(empleado);
        tx.commit();
    }
}
```

Bibliografía

Argentina programa 4.0

https://www.ibm.com/docs/es/was-liberty/nd?topic=overview-java-persistence-api-jpa

https://www.digitalocean.com/community/tutorials/jpa-hibernate-annotations

https://www.baeldung.com/jpa-many-to-many

https://docs.oracle.com/javaee/6/api/javax/persistence/EntityManager.html

https://www.baeldung.com/jpa-cascade-types

Gracias!