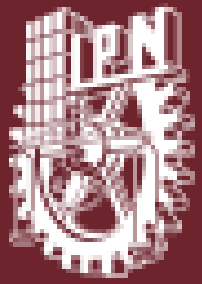


Instituto Politécnico Nacional

Escuela Superior de Cómputo

Java Persistence API

M. en C. José Asunción Enríquez Zárate

jenriquezz@ipn.mx

Contenido

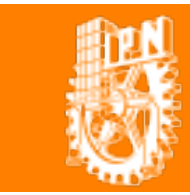
Contenido

Introducción EZJA

Introducción

La API de Persistence Java (JPA), es un framework ligero que aprovecha los POJOS (Plain Old Java Object) para persistir objetos Java que representan datos relacionales (normalmente en una base de datos).

- JPA está construido en una capa superior a JDBC y aborda la complejidad de la administración de código SQL y Java.
- Diseñado para facilitar el mapeo Objeto-Relacional.
- Funciona en entorno Java SE y Java EE.
- Los Conceptos clave incluyen:
  - Entidades.
  - Unidad de Persistencia.
  - Contexto de Persistencia.



Object Relational-Mapping

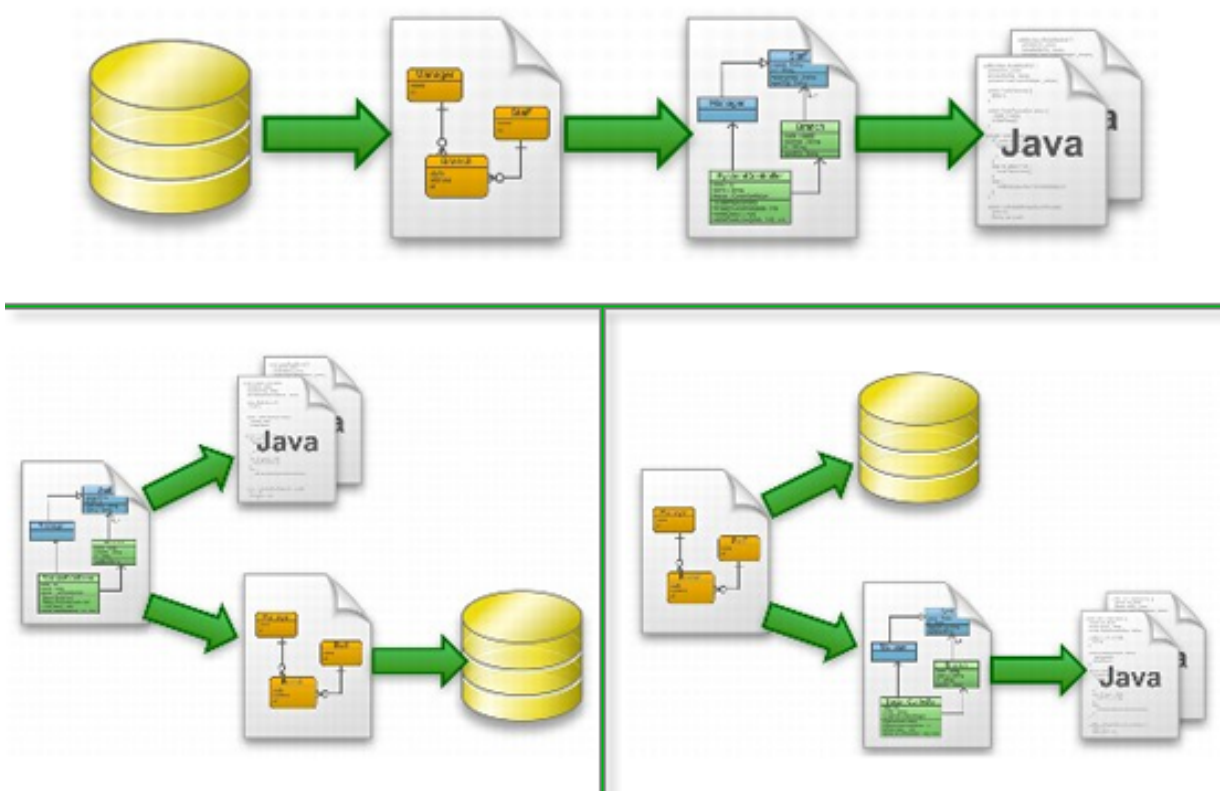


Figure: Object Relational-Mapping

Object Relational-Mapping

Object Relational-Mapping

- El mecanismo de almacenamiento que es más utilizado por las aplicaciones es una base de datos relacional.
- Una base de datos relacional
  - Almacenar datos en tablas.
  - Pueden o no tener relaciones.
- Un objeto de dominio Java puede incluir datos parciales de una tabla de base de datos única o puede incluir datos de varias tablas en función de la normalización de la base de datos relacional.
- Los mecanismos de persistencia son el código que permite a los programadores persistir (almacenar) datos en una base de datos relacional.
  - JDBC y JPA son dos ejemplos de mecanismos de persistencia.

Contenido

Entidades JPA

Entidades JPA

- Una Entidad describe un concepto o algo concreto que puede ser representado por datos (atributos) que posiblemente tienen relaciones con otras entidades.
  - Las entidades están destinadas a ser persistidas, (almacenadas en una base de datos relacional).
  - Una Entidad
    - Es un POJO que se crea mediante el uso de la palabra clave ***new***.
    - Soporta Herencia y Polimorfismo.
    - Debe ser Serializable (Es posible utilizar objetos Separados (detached)).
    - Debe tener un constructor con argumento cero.
  - Las entidades pueden ser consultadas.
  - Las entidades se gestionan en tiempo de ejecución a través del ***Entity Manager***.

Entidades JPA

Caracterśticas de las Entidades JPA

- Una entidad JPA es un objeto Java que representa algo que debe persistir: un cliente, un empleado, un libro, etc. Básicamente, todo lo que normalmente se representa en una base de datos relacional puede ser representado por un POJO.
  - ***Persistencia*** Las entidades que se crean se pueden almacenar en una base de datos (persistida) y tratarse como objetos.
  - ***Identidad*** Cada entidad tiene una identidad de objeto única, generalmente asociada con una clave en el almacén persistente (base de datos). Normalmente, la clave es la llave principal de la base de datos.
  - ***Transacciones*** JPA requiere un contexto transaccional para las operaciones que envían cambios a la base de datos.
- Las entidades son administradas por la API de EntityManager.

Ejemplo: POJO y Entidad

```
public class Escuela implements Serializable {
    private Integer idEscuela;
    private String nombreEscuela;
    private String calle;
    private String colonia;
    private Integer numero;
    private int codigoPastal;
    private String telefono;
    private String email;
    public Escuela() {}
    public Integer getIdEscuela() {...3 lines }
    public void setIdEscuela(Integer idEscuela) {...3 lines }
    public String getNombreEscuela() {...3 lines }
    public void setNombreEscuela(String nombreEscuela) {...3 lines }
    public String getCalle() {...3 lines }
    public void setCalle(String calle) {...3 lines }
    public String getColonia() {...3 lines }
    public void setColonia(String colonia) {...3 lines }
    public Integer getNumero() {...3 lines }
    public void setNumero(Integer numero) {...3 lines }
    public int getCodigoPastal() {...3 lines }
    public void setCodigoPastal(int codigoPastal) {...3 lines }
    public String getTelefono() {...3 lines }
    public void setTelefono(String telefono) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }
    @Override
    public String toString() {...3 lines }

}
```

Escuela	
idEscuela	int(11)
nombreEscuela	varchar(50)
calle	varchar(35)
colonia	varchar(35)
numero	int(11)
codigoPastal	int(11)

Ejemplo:Clase Entidad JPA Escuela

```
import javax.persistence.Id;
import javax.persistence.Entity;
@Entity
public class Escuela implements Serializable {
   @Id
   private Integer idEscuela;
   private String nombreEscuela;
   private String calle;
   private String colonia;
   private Integer numero;
   private int codigoPostal;
   private String telefono;
   private String email;

   public Escuela() {
   }
   ...
}
```

La clase entidad tiene el mismo nombre que la tabla

Primary Key

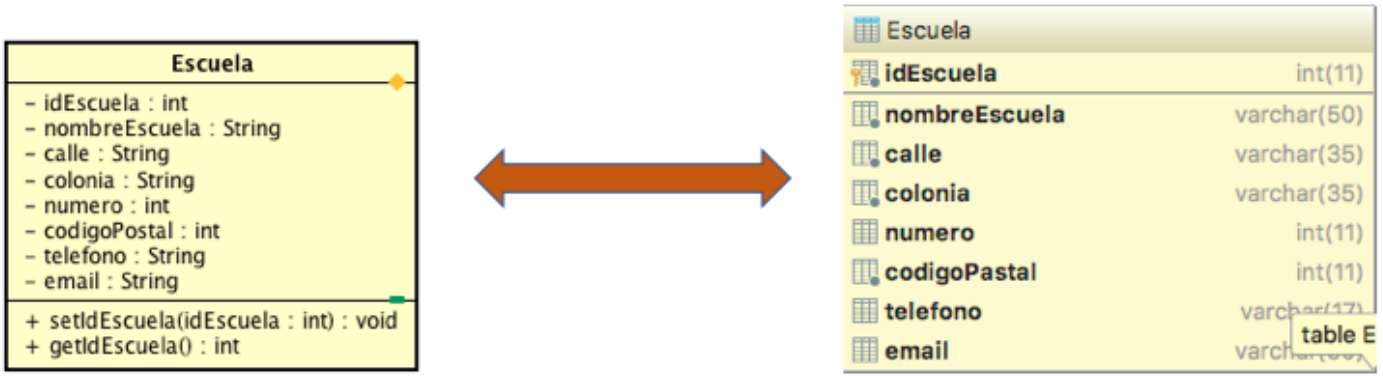
El nombre del atributo debe coincidir con los atributos de la tabla

Requisitos de una clase Entidad en JPA

Requisitos de una clase Entidad en JPA

- La clase Entidad debe contener la anotación @Entity.
- La clase Entidad debe tener un constructor sin argumentos.
- El constructor sin Argumentos deben ser público o protegido.
- El estado de persistencia de una entidad está representada por campos de instancias.
- Las Entidades soportan herencia y no debe ser final.

Mapeo de Entidad



idEscuela	nombreEscuela	calle
1	Escuela superior de Cómputo	Juan de Dios Batiz
2	Escuela Superior de Medicina	Juan de Dios Batiz
3	Escuela Superior de Economía	Juan de Dios Batiz

Llaves Primarias (Primary Key)

Llaves Primarias (Primary Key)

- Todas las Entidades deben tener una llave primaria. La llave primaria se utiliza para diferenciar entre una instancia y otra.
  - La llave primaria provee la identidad de una Entidad y es utilizada por el Entity Manager.
  - La anotación @Id se utiliza para definir una llave primaria.
  - Lo común es tener una llave primaria de tipo entero, pero es posible utilizar una clase personalizada que corresponda a varias columnas de la entidad de la base de datos (clave compuesta).
  - La llave primaria puede ser generada automáticamente.
- @Id private int idEscuela;

Sobreescribir la Asignación de Mapeos

Overriding

- @Table
  - Se utiliza para anular la asignación predeterminada entre una clase y una tabla.
- @Column
  - Se utiliza para reemplazar la asignación de nombres de columnas.

```java
@Entity
@Table(name = "Escuela")
public class Escuela implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idEscuela")
    private Integer idEscuela;
    @Basic(optional = false)
    @Size(min = 1, max = 50)
    @Column(name = "nombreEscuela")
    private String nombreEscuela;
}
```

Sobreescribir la Asignación de Mapeos

Posibles valores a utilizar en la anotación @Column

- columnDefinition (String):
  - Un fragmento de SQL que se utiliza para generar el DLL para la columna.
- length (int):
  - Longitud de la columna.
- nullable (boolean):
  - Indica si la columna puede tomar o no valores nulos.

Sobreescribir la Asignación de Mapeos

Posibles valores a utilizar en la anotación @Column

- unique (boolean):
  - Indica si la columna es una clave única.
- insertable (boolean):
  - Si esta columna se incluye o no en las instrucciones INSERT generadas por el proveedor de persistencia.
- updatable (boolean):
  - Si esta columna se incluye o no en las instrucciones UPDATE generadas por el proveedor de persistencia.

Tipos de Datos Persistentes

Tipos de Datos

- Las propiedades o campos persistentes pueden ser de los siguientes tipos de datos:
  - Tipos de Datos Primitivos y Wrappers Java.
    - Integer, Boolean, Float, andDouble.
  - Tipos de Serializables.
    - String, BigDecimal, and BigInteger.
  - Arreglos de Bytes y Carácteres .
    - byte[] and Byte[], char[] and Character[].
  - Temporal types.
    - java.util.Date,java.util.Calendar.
    - java.sql.Date, and java.sql.TimeStamp
  - Enums y Collections.
  - Otras Entidades.

Contenido

Unidades de Persistencia y Entity Manager

Entity Manager

La Interfaz EntityManager (EM) trabaja con entidades persistentes.

- Las entidades a las que una instancia del EM tiene referencia son administradas por el EM.
- Un conjunto de entidades dentro del EM en algún momento se denomina Contexto de Persistencia.
- Las entidades en el contexto de persistencia son únicas

Unidades de Persistencia y Entity Manager

Entity Manager

La Interfaz EntityManager (EM) trabaja con entidades persistentes.

- Un proveedor de persistencia crea los detalles de la implementación para leer y escribir en una base de datos determinada.
- Una instancia de un EntityManager se inyecta a través de la anotación @PersistenceContext
  - @PersistenceContext (unitName="escuelaPU")
    private EntityManager em;

Unidades de Persistencia y Entity Manager

Operaciones del EntityManager

El gestor de entidades se utiliza para realizar operaciones en entidades. La interfaz EntityManager define los métodos CRUD que se utilizan para interactuar con un contexto de persistencia.

```
@PersistenceContext (unitName="escuelaPU")
private EntityManager em;
private Escuela escuela;
```

CRUD	Método del EM	Ejemplo
Create	persist(entity);	em.persist(escuela);
Retrieve	find(entityClass,primaryKey);	emp = em.find(Escuela.class, idEscuela);
Update	merge(entity);	em.merge(escuela);
Delete	remove(entity);	em.remove(escuela);

Unidades de Persistencia y Entity Manager

Unidad de Persistencia

- Definido en el archivo persistence.xml.
- Agrupa de manera lógica.
  - Un Entity Manager Factory y sus Entity Manager.
  - Un conjunto de clases administradas.
  - Mapeos de metadatos.

Unidades de Persistencia y Entity Manager

Unidad de Persistencia

```xml
<?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="DemoWebFlisolJPA2017PU" transaction-type="JTA">
    <jta-data-source>jdbc/ezjaFlisol</jta-data-source>
    <class>com.darkdestiny.model.entities.Escuela</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>
    <properties>
      <property name="eclipselink.logging.level" value="FINE"/>
    </properties>
  </persistence-unit>
</persistence>
```

Unidades de Persistencia y Entity Manager

contexto de Persistencia

- Cuando un Entity Manager administra instancias de entidad, las instancias están en un contexto de persistencia.
- El gestor de entidad para un contexto de persistencia se obtiene:
  - Utilizando inyección de dependencia.

```
@PersistenceContext
private EntityManager em;

public List<Escuela> readAll() {
    TypedQuery<Escuela> query = em.createQuery("SELECT e FROM Escuela e",
            Escuela.class);
    List<Escuela> listaEscuela = query.getResultList();
    return listaEscuela;
}
```

Contenido

Consultas JPQL

Consultas

- JPA soporta dos métodos para construir consultas.
  - Lenguaje de Consultas JPQL y SQL Nativo
    - JPQL opera sobre la entidad lógica
  - API Criteria
    - Construye las consultas con base en objetos Java.

Consultas JPQL

JPQL

- JPA soporta las siguientes sentencias.
  - SELECT
    - Devuelve filas de datos almacenadas en la base de datos como objetos de entidad.
  - UPDATE
    - Actualiza una o más instancias de entidad.
  - DELETE
    - Elimina una o más instancias de entidad.

Consultas JPQL

Filtrado de Resultados

- La cláusula WHERE se utiliza para definir condiciones sobre los datos devueltos.

```
SELECT e FROM Escuela WHERE e. nombreEscuela LIKE '%ed' AND e.calle LIKE '%Ba'
```

```
WHERE e.address.state IN ('CA', 'MA')
WHERE e.salary BETWEEN 80000 AND 120000
WHERE o.lineItems IS NOT EMPTY
WHERE e.salary < 100000
WHERE b.bid MEMBER OF a.auction
```

Consultas JPQL

Operadores

- Importante tener en cuenta que el tipo de operando depende de la operación de expresión.

Opcional	Operador	Tipos
	=, >, >=, <=,<,<>	Arithmetic
[NOT]	BETWEEN	Arithmetic, string, date, time
[NOT]	LIKE	String
[NOT]	IN	All Types
IS [NOT]	EMPTY	Collection
[NOT]	MEMBER OF	Collection
[NOT]	LIKE	String

Consultas JPQL

```
public List<Escuela> readAll() {
    TypedQuery<Escuela> query = em.createQuery("SELECT e FROM Escuela e",
            Escuela.class);
    List<Escuela> listaEscuela = query.getResultList();
    return listaEscuela;
}
```

Ejemplo de Consula JPQL

- También existen estáticas y se definen con:
  - @NamedQuery, @NamedNativeQuery

Contenido

Referencias

Edgar Martinez, Tom McGinn, Eduardo Moranchel, Anjana Shenoy, Michael Williams.
Java EE 7: Back-end Server Application Development
Oracle, 2016.

Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Kim Haase, William Markito.
ava Platform, Enterprise Edition The Java EE Tutorial, Release 7
Oracle, 2014.