

CURSO JAVA EE

JAVA PERSISTENCE API (JPA)



Ing. Ubaldo Acosta

Por el experto: Ing. Ubaldo Acosta



CURSO JAVA EE

www.globalmentoring.com.mx

Hola, te saluda nuevamente Ubaldo Acosta. Espero que estés listo para comenzar con esta lección.

Vamos a revisar como integrar el Api de Persistencia de Java (JPA) con Java EE.

¿Estás listo? ¡Vamos!

¿QUÉ ES JAVA PERSISTENCE API?

- Java Persistence API, mejor conocido como JPA, es el estándar de persistencia de Java. JPA implementa conceptos de frameworks ORM (Object Relational Mapping)



www.globalmentoring.com.mx

La mayoría de la información de las aplicaciones empresariales es almacenada en bases de datos relacionales. La persistencia de datos en Java, y en general en los sistemas de información, ha sido uno de los grandes temas a resolver en el mundo de la programación.

Al utilizar únicamente JDBC tenemos el problema de crear demasiado código para poder ejecutar una simple consulta. Por lo tanto, para simplificar el proceso de interacción con una base de datos (select, insert, update, delete), se ha utilizado desde hace ya varios años el concepto de frameworks ORM (Object Relational Mapping), tales como Hibernate.

Como podemos observar en la figura, un framework ORM nos permite "mapear" una clase Java con una tabla de Base de Datos. Por ejemplo, la clase Persona, al crear un objeto en memoria, podemos almacenarlo directamente en la tabla de Persona, simplemente ejecutando una línea de código: `em.persist (persona)`. Esto ha simplificado enormemente la cantidad de código a escribir en la capa de datos de una aplicación empresarial.

En esta lección estudiaremos la tecnología Java Persistence API, mejor conocido como JPA. Esta tecnología de Java es el estándar de persistencia en Java, y la buena noticia es que cuenta con varias implementaciones, tales como Hibernate, EclipseLink, OpenJPA, entre algunas más. Así que si ya conoces Hibernate, o algún framework de persistencia Java, te será muy familiar muchos de los conceptos que estudiaremos en esta lección. Si aún no tienes conocimientos de esta tecnología y quieres profundizar más te invitamos a estudiar el curso Hibernate y JPA que tenemos disponible en Global Mentoring.

Aprenderemos temas como: Una visión general de JPA, Manejo y Ciclo de Vida de las entidades en JPA, JP Query Language para realizar consultas en la base de datos y Manejo de Transacciones utilizando EJB's y JPA, entre otros temas. Esta lección será crucial para generar una capa de datos robusta, flexible, y que pueda comunicarse con cualquier base de datos relacional.

CARACTERÍSTICAS DE JPA

- ✓ Persistencia utilizando POJOs.
- ✓ No Intrusivo.
- ✓ Consultas utilizando Objetos Java.
- ✓ Configuración Simplificada.
- ✓ Integración.
- ✓ Testing.



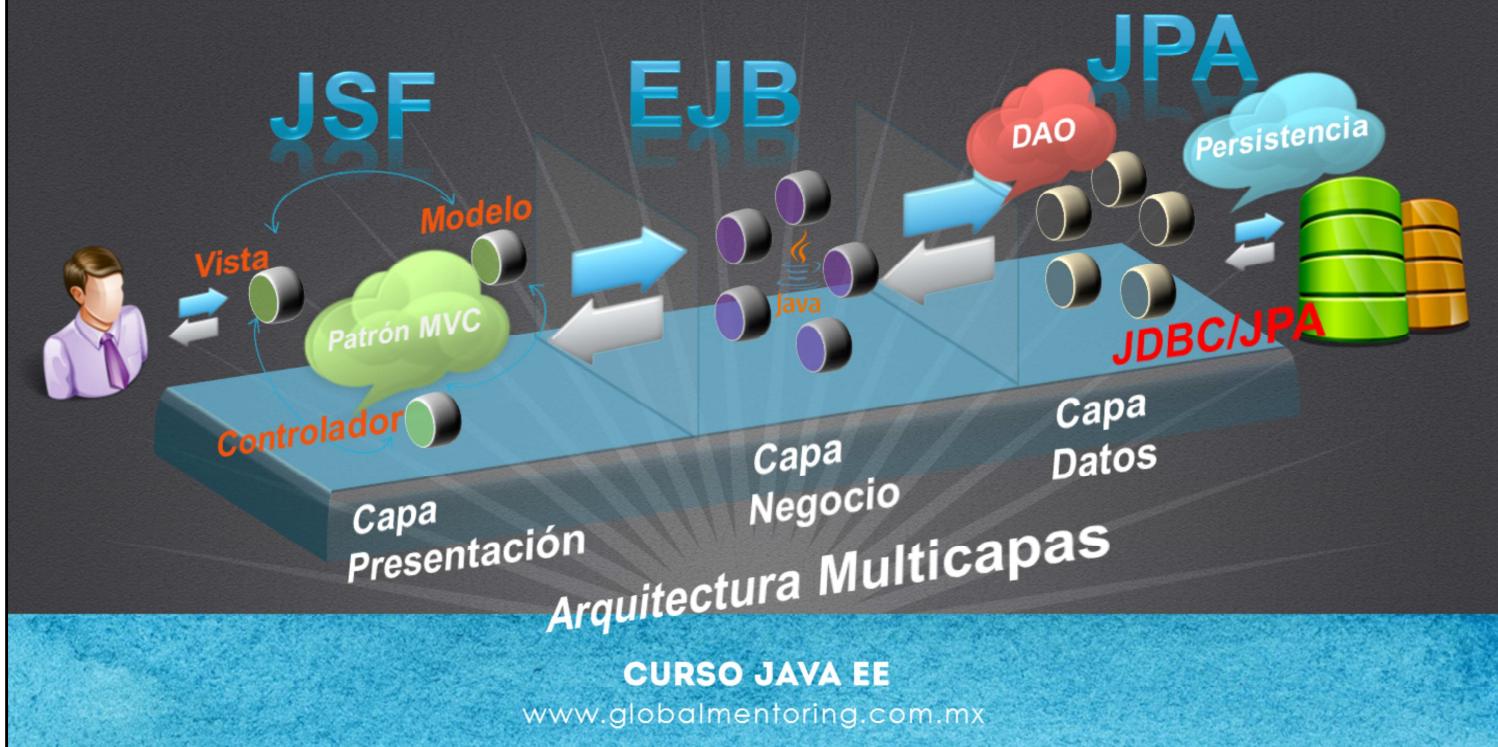
www.globalmentoring.com.mx

La idea del API JPA es trabajar con objetos Java y no con código SQL, de tal manera que podamos enfocarnos en el código Java. JPA permite abstraer la comunicación con las bases de datos y crea un estándar para ejecutar consultas y manipular la información de una base de datos.

Características de Java Persistence API:

- ✓ **Persistencia utilizando POJOs:** Este es posiblemente el aspecto más importante de JPA, debido a que cualquier clase de Java podemos convertirla en una clase de entidad, simplemente agregando anotaciones y/o agregando un archivo xml de mapeo.
- ✓ **No Intrusivo:** JPA es una capa separada de los objetos a persistir. Por ello, las clases Java de Entidad no requieren extender ninguna funcionalidad en particular ni saber de la existencia de JPA, por ello es no intrusivo.
- ✓ **Consultas utilizando Objetos Java:** JPA permite ejecutar queries expresados en términos de objetos Java y sus relaciones, sin necesidad de utilizar el lenguaje SQL. Los queries son traducidos por el API de JPA en el código SQL equivalente.
- ✓ **Configuración simple:** Muchas de las opciones de JPA están configuradas con opciones por default, sin embargo si queremos personalizarlas, es muy simple, ya sea con anotaciones o a través de archivos xml de configuración.
- ✓ **Integración:** Debido a que las arquitecturas empresariales Java son por naturaleza multicapas, una integración transparente es muy valiosa para los programadores Java y JPA permite hacer la integración con las demás capas de manera muy simple.
- ✓ **Testing:** Con JPA ahora es posible realizar pruebas unitarias, o utilizar cualquier clase con un método main fuera del servidor, simplemente utilizando la versión estándar de Java. Esto permite reducir los tiempos de desarrollo de las aplicaciones empresariales de manera considerable.

ARQUITECTURA EMPRESARIAL CON JPA



En la figura podemos observar el rol de JPA en una arquitectura Java Empresarial. Esta tecnología aplica directamente en la capa de datos, la cual se encarga de tareas tales como:

- Recuperación de información a través de consultas (select)
- Manejo de información de objetos Java en las tablas de base de datos respectivas (insert, update, delete)
- Manejo de una unidad de persistencia (Persistence Unit) para la creación y destrucción de conexiones a la base de datos
- Manejo de transacciones, respetando el esquema de propagación definido en la capa de negocio en los EJBs de Sesión.
- Portabilidad hacia otras bases de datos con un impacto menor, así como bajo acoplamiento con las otras capas empresariales.
- Entre varias tareas más.

Además, para realizar las tareas de persistencia, podemos utilizar patrones de diseño tales como

- DAO (Data Access Object): Este patrón de diseño suele definir una interfaz y una implementación de dicha interfaz, para realizar las operaciones más comunes con la Entidad respectiva. Por ejemplo, para la entidad Persona, generaremos la interfaz DaoPersona, y agregaremos los métodos agregarPersona, modificarPersona, eliminarPersona, findAllPersonas, etc
- DTO (Data Transfer Object): Este patrón de diseño permite definir una clase, que en ocasiones es muy similar a la clase de entidad, ya que contiene los mismos atributos, pero con el objetivo de transmitirla a las siguientes capas, incluso, hasta la capa Web. Por ello se les conoce como objetos de valor o de transferencia.

En la actualidad, y con la simplificación de las capas de una arquitectura empresarial, es opcional utilizar estos y otros patrones de diseño empresariales, sin embargo, muchas aplicaciones Java han sido construidas con estos patrones, así que vale la pena entender para qué se utilizan y cómo aplicarlos.

CLASES DE ENTIDAD

- ✓ Una clase de entidad es un POJO y puede configurarse por medio de anotaciones o un archivo XML.
- ✓ Ejemplo de clase de Entidad con anotaciones:

```

@Entity
public class Persona {

    @Id
    @GeneratedValue
    private Long personaId;

    @Column(nullable = false)
    private String nombre;

    private String apePaterno;

    private String apeMaterno;
    private String email;
    private Integer telefono;

    // Constructores, getters, setters
}

```

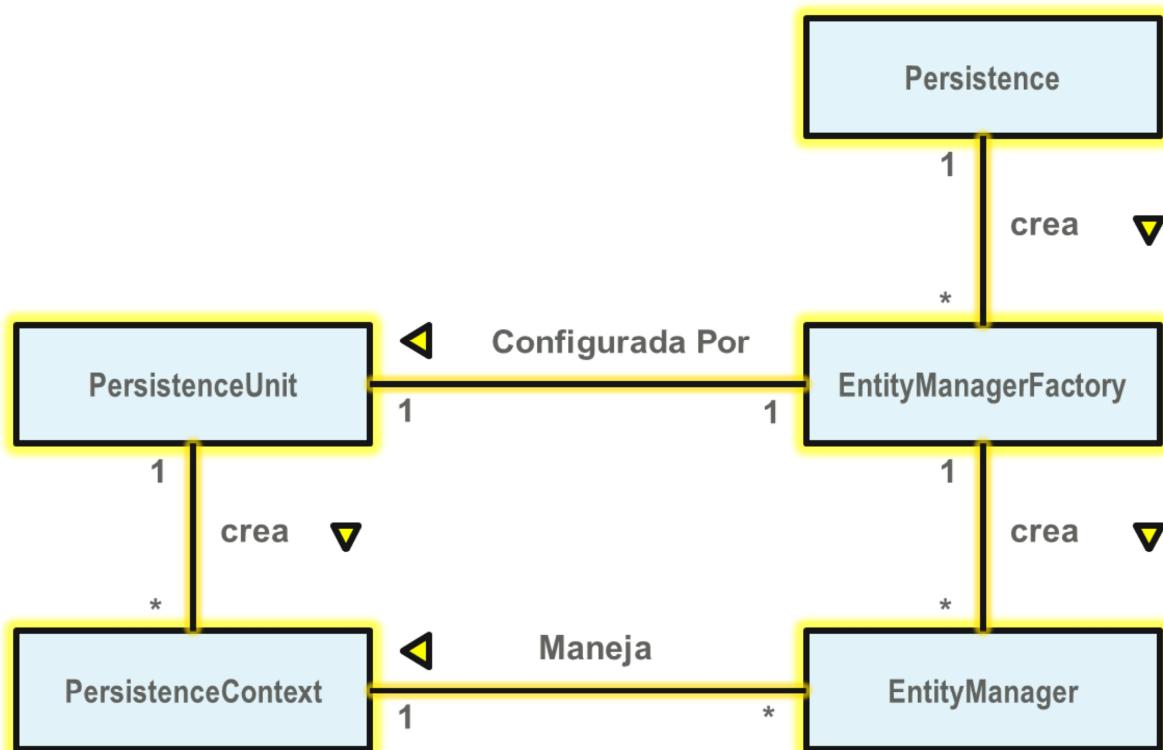
En las primeras versiones de J2EE, existía el concepto de Entity Beans para el manejo de persistencia. Sin embargo esta tecnología resultaba complicada para sistemas del mundo real, resultando en un bajo performance. Así mismo, realizar consultas (queries) utilizando los objetos Entity era muy complicado, se necesitaba un servidor de aplicaciones Java para ejecutar un EJB tipo Entity, no existían pruebas unitarias, y por lo tanto cualquier cambio en nuestro código implicaba un nuevo deploy, consumiendo mucho tiempo simplemente en probar nuestro código de persistencia, entre varios detalles más.

Sin embargo, el concepto de los Entity Beans era muy bueno desde el punto de vista de la programación orientada a objetos. Hibernate fue uno de los primeros frameworks en incorporar muchas de las características que podemos utilizar al día de hoy con JPA. Hibernate simplificó por mucho la tecnología de los EJB de Entidad, y aunque no es un estándar, en muchas aplicaciones Java, escoger Hibernate es garantía de un excelente framework de persistencia.

Al día de hoy, una clase conocida como Entidad es simplemente un POJO, y en combinación con el uso de anotaciones, es suficiente para convertirla en una clase de Entidad, la cual representa un registro de una tabla de base de datos. Este tipo de conceptos, heredados de frameworks como Hibernate, TopLink, JDO, entre otros, contribuyó en lo que conocemos al día de hoy como el estándar de persistencia Java conocido como JPA.

El API de JPA se puede utilizar en una aplicación estándar de Java o en un servidor Web o Empresarial Java. Ahora ya es posible realizar pruebas unitarias sobre nuestras clases de Entidad y Consultas sobre los objetos de Entidad, disminuyendo dramáticamente el tiempo de desarrollo de nuestras clases de entidad, consultas, y en general en la creación de la capa de datos de una aplicación empresarial.

API DE JPA Y ENTITY MANAGER



Para que una clase de Entidad pueda ser persistida, se debe realizar una llamada al API de JPA. De hecho muchas de las operaciones se realizan a través de esta API, la cual está separada de nuestras clases de Entidad.

En la figura podemos observar el API JPA, la cual tiene como elemento principal al objeto **EntityManager**, siendo este una interfaz. Una implementación de esta interfaz es la que realmente ejecuta el trabajo de persistencia, sincronización con la base de datos, transaccionalidad, validación de mapeo, conversión de código Java a SQL, entre muchas otras tareas.

Por lo tanto, una clase Entity, desde el punto de vista descrito anteriormente, es tan solo una clase Java normal, la cual al vincularse con un **EntityManager**, se persiste en la base de datos.

El objeto **EntityManager** se obtiene de una fábrica de objetos conocida como **EntityManagerFactory**, y este objeto se asocia con un proveedor JPA, pudiendo haber seleccionado entre varios proveedores según la implementación de JPA escogida (Hibernate, EclipseLink, OpenJPA, etc).

El objeto **Persistance Unit**, se encarga de realizar la configuración del proveedor seleccionado por medio de un archivo xml, además de definir otros elementos tales como: la forma de comunicarse con la Base de Datos, las clase de Entidad en la aplicación, si se va a utilizar JTA para el manejo transaccional, entre varias características más.

A su vez, al conjunto de objetos Entity administrados por JPA en un tiempo específico de la aplicación se le conoce como **PersisteceContext**, de esta manera JPA se asegura que no existan objetos de Entidad duplicados en memoria, entre otras tareas más.

Un ejemplo de cómo utilizar el API JPA para persistir un objeto de Entidad, se muestra a continuación:

```

EntityManagerFactory emf = Persistence.createEntityManagerFactory("PersonaService");
EntityManager em = emf.createEntityManager();
Persona persona = new Persona(15);
em.persist(persona);
  
```

CONFIGURACIÓN DE UNIDAD DE PERSISTENCIA

Ejemplo de contenido del archivo persistence.xml:

```
<persistence>
  <persistence-unit name="PersonaService" transaction-type="RESOURCE_LOCAL">
    <class>domain.Persona</class>
    <properties>
      <property name="javax.persistence.jdbc.driver"
                value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="javax.persistence.jdbc.url"
                value="jdbc:derby://localhost:1527/PersonaServDB;create=true"/>
      <property name="javax.persistence.jdbc.user" value="APP"/>
      <property name="javax.persistence.jdbc.password" value="APP"/>
    </properties>
  </persistence-unit>
</persistence>
```

CURSO JAVA EE
www.globalmentoring.com.mx

Para realizar la configuración de la Unidad de Persistencia se debe utilizar un archivo xml llamado persistence.xml.

El nombre del elemento persistence-unit indica el nombre de la unidad de persistencia, y es muy importante recordarlo, ya que es el nombre que utilizaremos en nuestro código Java al momento de utilizar el objeto EntityManagerFactory.

El atributo transaction-type especifica el tipo de transaccionalidad que se utilizará, pudiendo seleccionar JTA como el proveedor.

Se pueden especificar también las clases de Entidad del sistema, pudiendo definir varias clases. Si la aplicación se despliega en un servidor Java, no es necesario declarar estas clases, sin embargo, para aplicaciones Java SE (Standard Edition) es necesario especificar las clases de Entidad del sistema.

La sección de propiedades especifica características del proveedor a utilizar, así como los datos de conexión a la base de datos.

Al momento de empaquetar una aplicación Java, el archivo persistence.xml se debe ubicar en la carpeta META-INF/persistence.xml del archivo .jar.

UTILIZANDO LA UNIDAD DE PERSISTENCIA

Ejemplo de uso de la unidad de persistencia y del Entity Manager:

```

@Stateless
public class PersonaServiceBean implements PersonaService {

    @PersistenceContext(unitName="PersonaService")
    EntityManager em;

    public void agregarPersona(Persona persona) {
        em.persist(persona);
    }

    public Persona encontrarPersona(int idPersona) {
        return em.find(Persona.class, idPersona);
    }

    public Persona modificarNombrePersona(int idPersona, String nuevoNombre) {
        Persona persona = em.find(Persona.class, idPersona);
        if (persona != null) {
            persona.setNombre(nuevoNombre);
        }
        return persona;
    }

    public void eliminarPersona(int idPersona) {
        Employee emp = em.find(Employee.class, idPersona);
        em.remove(emp);
    }
}

```

En la figura podemos observar un ejemplo de un código donde hacemos uso de la unidad de persistencia para obtener el objeto EntityManager. Una vez que ya hemos obtenido una referencia al objeto EntityManager, podemos comenzar a realizar las operaciones con los objetos de Entidad.

Por ejemplo, podemos realizar tareas como:

- ✓ **Inserción:** Para persistir una entidad se utiliza el método **persist** del EntityManager. Con este método podemos generar un registro en la base de datos. Este registro no será guardado hasta haber concluido la transacción (commit). Recordemos que los métodos de un Session Bean son transaccionales por default, esto implica que al terminar de ejecutar el método agregarPersona y al ejecutarse en un contenedor empresarial Java, en automático se realizará el commit. En lecciones posteriores revisaremos el tema de transacciones dentro y fuera de un servidor de aplicaciones.
- ✓ **Búsqueda:** Una vez que tenemos un objeto persistido, podemos recuperar la información del registro de la base de datos utilizando el método **find** del objeto EntityManager, y basta con especificar el tipo (clase) y el id (llave primaria) que estamos buscando.
- ✓ **Modificación:** La modificación cambia un poco, debido a que JPA necesita primero saber con qué entidad se está trabajando, por ello necesitamos recuperar el objeto de entidad. Una vez recuperado, realizamos las modificaciones necesarias, y si el objeto se encuentra en una transacción activa, JPA revisará en automático si es necesario realizar alguna actualización sobre el registro. Lo interesante es que no es necesario volver a llamar al método **persist**, esta llamada es opcional.
- ✓ **Eliminación:** Similar a la modificación, primero se debe recuperar la entidad con el método **find**, y una vez en memoria, llamamos el método **remove**.

Estas son las operaciones básicas utilizando el objeto EntityManager sobre nuestros objetos de entidad. A continuación revisaremos algunos ejemplos para profundizar más en este tema.

EJERCICIOS CURSO JAVA EE

- **ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.**
- **EJERCICIO:** Persistencia de Objetos con JPA y Testing.
- **EJERCICIO:** SGA con JPA.



CURSO JAVA EE
www.globalmentoring.com.mx

CURSO ONLINE

JAVA EMPRESARIAL JAVA EE

Por: Ing. Ubaldo Acosta



Experiencia y Conocimiento para tu vida

CURSO JAVA EEwww.globalmentoring.com.mx

En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- | | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none">✓ Programación con Java✓ Fundamentos de Java✓ Programación con Java✓ Java con JDBC✓ HTML, CSS y JavaScript✓ Servlets y JSP's✓ Struts Framework | <ul style="list-style-type: none">✓ Hibernate Framework y JPA✓ Spring Framework✓ JavaServer Faces✓ Java EE (EJB, JPA y Web Services)✓ JBoss Administration✓ Android con Java✓ HTML5 y CSS3 |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Datos de Contacto:Sitio Web: www.globalmentoring.com.mxEmail: informes@globalmentoring.com.mx