ISIL APRENDE HACIENDO

Persistencia de Datos con JAVA

JPA

Utilización de JDBC mediante la API de persistencia

- La API de persistencia, que es parte de JEE pero se puede utilizar en otras aplicaciones Java, permite la especificación de manera sencilla de aplicaciones con un número elevado de usuarios simultáneos que acceden de manera eficiente a bases de datos.
- Cuando una aplicación Java utiliza la API de persistencia (JPA), se crea un ***pool de conexiones JDBC*** que se reutilizan por los distintos hilos de la aplicación.

http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software

JPA: Consultas

- El Lenguaje de Consultas de Persistencia de Java (JPQL) permite especificar consultas a bases de datos relacionales en forma parecida a SQL desde un programa Java.

JPA: Consultas, II

- Los objetos que implementan la interfaz TypedQuery<?> de JPA representan consultas del tipo anterior.
- Los objetos de la clase anterior que representan consultas que devuelven valores (tipo SELECT) al ejecutar la consultas devuelven listas de objetos.

Consultas en JPQL

- SELECT **p** FROM **Persona p**
  - Persona es una **entidad** (clase)
  - p representa una variable Java de la clase
  - El resultado de la ejecución de la consulta es una **lista** de objetos de la clase
- Se pueden añadir otras cláusulas:
  - SELECT p FROM Persona p

    WHERE p.nombre = ‘Pepe’

JPA: EntityManager

- Para crear y ejecutar consultas mediante JPA es necesario crear antes un objeto que implementa la interfaz EntityManager, que se ocupa de la gestión de los datos y de objetos que los representan.
- A continuación veremos cómo se utilizan y cómo se crean los EntityManagers.
- Tras esto veremos cómo se definen las clases de objetos que representan registros de una base de datos.

JPA: Programación de consultas

- Para crear una consulta se utiliza el método createQuery de la clase EntityManager, cuyo argumento es la cadena de caracteres que define la consulta JPQL:

```
TypedQuery<Persona> query =
        em.createQuery(
                        “SELECT p FROM Persona p”,
        Persona.class);
```

JPA: Acceso a resultados

- Para ejecutar una consulta de tipo SELECT se utiliza el método getResultList, que devuelve una lista de objetos:

```
java.util.List<Persona> pers = query.getResultList();
```

Gestión de persistencia en Java EE: Ejemplos

- Ejemplo de inyección de E.M.Factory:

  @PersistenceUnit

  private **EntityManagerFactory** emf;

  EntityManager em = emf.createEntityManager();

- Ejemplo de inyección de EntityManager:

  @PersistenceContext

  EntityManager em;

Ejemplo de fichero persistence.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<persistence version="1.0"
    xmlns="http://java.sun.com/xml/ns/persistence"
    xmlns:xsi="http://www.w3.org/...“
    xsi:schemaLocation="http://...">
    <persistence-unit name="book" transaction-type="JTA">
        <jta-data-source>jdbc/__default</jta-data-source>
        <class>com.widgets.Order</class>
        <class>com.widgets.Cust</class>
    </persistence-unit>
</persistence>
```

Clases entidad

- Son clases asociadas a tablas (una o varias) en una base de datos.
- Sus objetos se asocian a registros.
- Se declaran mediante la anotación @Entity.
- La clave primaria se indica mediante la anotación @Id (obligatorio).
- **@Entity**

  public class Person { @Id int id; … }

Clases entidad

- Si la tabla primaria tiene un nombre diferente de la clase, éste se especifica mediante la anotación @Table(name).
- NetBeans permite la opción de crear la tabla (si no existe) al compilar el código de la clase

Clases entidad

- Tipos de atributos:
  - Persistentes (ver próxima transparencia)
  - No persistentes (@Transient)
  - Embebidos (@Embedded), de otra entidad, incluidos en la tabla de la primera
  - Generados automáticamente (@GeneratedValue)
  - Relaciones (próxima transparencia)

Clases entidad: Atributos persistentes

- Son todos salvo los no persistentes
- Tienen que ser protegidos o privados
- Tipos: Primitivos, wrappers, String, temporales (Date, …), arrays de bytes y chars, enumeraciones, clases embedibles
- Colecciones: Collection, Set, List, Map

Definición de entidades en persistence.xml (alternativa)

```
<persistence>
   <persistence-unit name="…">
        <jta-data-source>…</jta-data-source>
        …
        <class>com.widgets.Order</class>
        <class>com.widgets.Cust</class>
   </persistence-unit>
</persistence>
```

JPA básico: Resumen

- Las clases de entidades representan registros de tablas. Se definen mediante la anotación @Entity.
- Los recursos JDBC y los pools de conexiones se utilizan por el servidor web para implementar la conexión a la base de datos. Se definen en el servidor.

JPA básico: Resumen, II

- Los recursos utilizados por una aplicación se especifican en el fichero persistence.xml.
- Los EntityManagers gestionan el acceso a las bases de datos. Se inyectan en un objeto gestionado por un contenedor o se crean a partir de una fábrica inyectada. Están asociados a un recurso JDBC.

JPA básico: Resumen, III

- Los TypedQuerys pueden ejecutar consultas JPA. Se crean a partir de ellas y en su caso devuelven listas de entidades.

Ejercicio

- [DBPERS0] Desarrollar una aplicación web que permita ver la lista de personas incluidas en una base de datos de personas sencilla, como la mencionada en los ejemplos anteriores.

JPA: Programación de consultas, II

- Se pueden definir TypedQuerys con tipo atómico (Integer, String, etc.) para consultas que seleccionan una sola columna:

```
TypedQuery<String> query =
        em.createQuery(
                "SELECT p.nombre
                        FROM Persona p",
        String.class);
```

JPA: Programación de consultas, III

- Las consultas de actualización o borrado se definen de forma similar:

```
UPDATE Persona p
    SET p.nacionalidad='española'

DELETE FROM Persona p
    WHERE p.estado='fallecido'
```

JPA: Programación de consultas, IV

- Para ejecutar una consulta de tipo UPDATE o DELETE se utiliza el método executeUpdate():

```
query.executeUpdate();
```

JPA: Programación de consultas, V

- Se pueden definir consultas parametrizadas utilizando patrones de parámetros:

```
TypedQuery<Persona> query=
        em.createQuery(
                "SELECT p FROM Persona p
                        WHERE p.edad=?1);
pers = query.setParameter(1, 3).getResultList();
```

Ejercicio

Desarrollar una aplicación web que permita gestionar a través de Internet una base de datos de personas sencilla, como la mencionada en los ejemplos anteriores, permitiendo dar de alta y de baja a personas y modificar sus datos.

Ejercicios voluntarios

Realizar un mantenimiento de Datos de la tabla productos de la base de datos Intranet.

JPA: Programación de consultas, VI

- Se pueden utilizar consultas SQL (consultas nativas) mediante el método createNativeQuery de la clase EntityManager.
- El objeto creado por EntityManager.createNativeQuery implementa la interfaz Query, superinterfaz de TypedQuery<?>.

JPA: Programación de consultas, VII

- Los registros obtenidos mediante el método Query.getResultList a partir de una consulta SQL son arrays de objetos.
- Ejemplo:

```
Object[] persona =
        (Object[]) em.createNativeQuery(
                "SELECT * FROM PERSONA”)
                        .getResultList().get(0);
out.println(persona[0] + " " + persona[1]);
```

JPA: Programación de consultas, VIII

- Se puede determinar el número del registro a partir del cual se extraen los resultados y el número máximo de resultados a extraer en una consulta:

```
em.createQuery(
    “SELECT p FROM Persona p”)
        .setFirstResult(5).setMaxResults(9);
```

Búsqueda de objetos individuales

- Se hace mediante el método find(Class<?>, Object) de la clase EntityManager.
- Utiliza su segundo argumento como clave primaria.
- Ejemplo:

  Person p = e.find(Person.class, 2130);

Entidades persistentes: Ciclo de vida

- Las entidades persistentes pueden estar desacopladas de la base de datos o acopladas a ella a través de un EntityManager. En este caso están en el estado Managed (gestionadas).
- Por ejemplo, una entidad persistente creada a través de una consulta a la base de datos normalmente está gestionada por el EntityManager que hace la consulta.