UNIVERSIDAD DEL QUINDÍO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Información general	
FECHA :	2015-02-16
DURACIÓN ESTIMADA EN MINUTOS:	120
DOCENTE:	Christian Andrés Candela
GUÍA NO.	03
Nombre de la guía:	Entidades

Información de la Guía

Objetivos: Estudiar el uso de las entidad y su aplicación en el modelamiento de datos.

Conceptos Básicos: Manejo de Eclipse, Java, Bases de Datos, JDBC, XML, Glassfish.

Contextualización Teórica: Una entidad es un elemento java de persistencia, que permite modelar o representar un objeto o información del sistema objeto de análisis. Toda entidad está compuesta por uno o más atributos, cada uno de los cuales representa un atributo, propiedad o información del objeto que se está modelando. Normalmente, una entidad representa una tabla en una base de datos relacional, y cada instancia de la entidad corresponde a una fila de esa tabla.

Las entidades surgen inicialmente como un tipo de EJB especializado en la persistencia de la información. Sin embargo en la versión JEE 5 pasa de ser un tipo de EJB para convertirse en el corazón de la api JPA (Java Persistence Api).

Para que una clase sea considerada una entidad la misma debe cumplir los siguientes requisitos:

- La clase debe ser marcada como Entidad por medio de la anotación javax.persistence.Entity
- La clase debe tener un constructor público o privado sin argumentos. Se puede tener más de un constructor.
- Las entidades pueden extender tanto de otras entidades como de clases que no son entidades, y las clases que no son entidades pueden extender de las entidades.
- Los atributos persistentes deben ser declarados como privados, protegidos o privados de paquete, y sólo deben ser accedidos directamente por los métodos de la entidad.

Para construir una entidad se pueden usar dos enfoques. El primero de ellos encaminado a persistir los campos (variables de la entidad), en cuyo caso, se usarán anotaciones en las variables para modificar la forma en que estas son almacenadas. El segundo enfoque se basa en la persistencia de las propiedades, en este caso se debe hacer uso la convención de los componentes JavaBeans para los nombres de los métodos de la entidad, y es precisamente en estos métodos en los que se realizarán las anotaciones necesarias para especificar la forma en la que se debe persistir la entidad. Si en algún momento se desea que una de las variables de la clase no sea almacenada se debe usar la anotación @Transient o marcada como transitoria. De igual forma es importante tener en cuenta que los elementos bien sea estáticos, finales o transient no son persistidos.

Es importante tener en cuenta que los atributos de una entidad que vayan a ser persistido pueden ser uno de los siguientes:

- Otras entidades
- Clases marcadas como super clases (@MappedSuperClass)
- Clases embebibles (@Embeddable).
- Tipos de datos simples (serializables), los cuales comprenden los tipos de datos primitivos, sus wrappers, BigInteger, BigDecimal, String
- Tipos de datos que representan una fecha (java.util.Date, java.util.Calendar, java.sql.Date, java.sql.Time, java.sqk.Timestamp), este tipo de datos deben ser marcados como @Temporal y especificar si con fechas (@Temporal(TemporalType.DATE)), horas (@Temporal(TemporalType.TIME)) o fecha y hora (@Temporal(TemporalType.TIMESTAMP))
- Colecciones de datos del paquete java.util (package java.util: ArrayList, Vector, Stack, LinkedList, ArrayDeque, PriorityQueue, HashSet, LinkedHashSet, TreeSet)
- Maps del paquete de datos java.util (HashMap, Hashtable, WeakHashMap, IdentityHashMap, LinkedHashMap, TreeMap y Properties)
- Arrays
- Enumeraciones, se debe tener en cuenta que por defecto las enumeraciones son persistidas en su representación numérica, por lo que si una enumeración se modifica debe hacerce al final para no afectar los datos previamente almacenados. Si se desea almacenar la enumeración como cadena en lugar de su representación numérica se debe indicar con @Enumerated(EnumType.STRING) o @Enumerated(EnumType.ORDINAL)
- Objetos serializables.

Cada objeto almacena en una base de datos debe identificarse de forma única, en el caso de las entidades estas son identificadas por su tipo y por una llave. En muchos casos cuando se construye una entidad puede identificarse claramente el atributo que lo identifica de forma única, en otros casos puede ser que la entidad sea identificada de forma única por un conjunto de atributos, o simplemente puede no existir un atributo que la identifique de forma única, en cuyo caso se debe asignar un atributo que represente la llave y que sea asignado de forma automática.

Tipo de llave

Básico: Cuando un campo o atributo simple (tipo primitivo, wrapper, BigInteger, BigDecimal, String, Date, Time, Timestamp, enum, otra entidad) identifica de forma única la entidad, en este caso dicho campo es marcado como la llave de la entidad así:

```
@Id
private String isbn;
```

Compuestas: Las llaves compuesta son aquellas conformadas por más de un campo, para crear una llave compuesta se debe crear una clase que agrupe los campos que compondrán la llave e identificar dicha clase en la entidad de forma específica así:

```
public class Llave{
  private int campo1;
  private long campo2;
  …
}

@Entity
@IdClass(Llave.class)
public class Entidad{
  @Id
  private int atributo1;
  @Id
  private long atributo2;

 …
}
```

Embebibles: En el caso de las llaves embebibles, son llaves que son compuestas por más de un campo, pero que son declaradas dentro de la entidad como un único atributo representado a través de un tipo de dato que ha su vez a sido marcado con la anotación @Embeddable así:

```
@Embeddable
public class Llave{
  private int campo1;
  private long campo2;
  …
}

@Entity
public class Entidad{
  @EmbeddedId
  private Llave atributo;
 …
}
```

Generadas: Las llaves generadas como se ha dicho, son aquellas que se crean para poder identificar de forma única a una entidad que por si misma no tiene un campo que la identifique de forma única. Estas llaves son generadas de forma automática por el sistema.

```
@Entity
public class Entidad{
  @Id
  @GeneratedValue
  private Long id;
}
```

Para la generación de valores de forma automática para la llave puede realizarse mediante diferentes estrategias (Auto, Identity, Sequence, Table).

- Auto (@GeneratedValue(strategy=GenerationType.AUTO)): En este caso se tiene un generador de números para cada base de datos, los cuales son usados para crear valores a ser asignados como llaves primarias. Esta es la estrategia usada por defecto.
- Identity (@GeneratedValue(strategy=GenerationType.IDENTITY)): Similar al auto, sin mebargo no existe un generador por base de datos, en su lugar existe un generador de números para cada tipo.
- Sequence (@GeneratedValue(strategy=GenerationType.SEQUENCE, generator=**"seq"**)): Se usa cuando el motor de base de datos soporta el uso de secuencias para la generación de números. En el caso de la secuencia se debe usar una anotación de clase que declare la secuencia y sus valores iniciales @SequenceGenerator(name="**seq**", initialValue=1, allocationSize=100)
- Table (@GeneratedValue(strategy=GenerationType.TABLE,generator="**tab**")): Similar a la secuencia, pero en este caso es el proveedor de JPA el encargado de crear una tabla para simular una secuencia y así porder asignar los números a ser usados como llaves primarias. La tabla tambien debe declararse previamente con una anotación de clase @TableGenerator(name="**tab**", initialValue=0, allocationSize=50)

Precauciones y Recomendaciones: Recuerde verificar que el servidor de aplicaciones soporte el motor de base de datos que usará, de igual forma debe verificar que eclipse este haciendo uso del JDK y no del JRE y recuerde adicionar al workspace el servidor de aplicaciones Glassfish antes de crear cualquier proyecto. También puede ser importante verificar que los puestos usados por Glassfish no estén ocupados (Para ello puede hacer uso del comando ***netstat -npl*** o ***netstat -a***).

Artefactos: Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse Luna), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish y el motor de base de datos Mysql.

Evaluación o Resultado: Se espera que el alumno logre modelar entidades lógicas de una aplicación y por medio de ellas se cree tablas en la base de datos que las representen de forma adecuada.

Procedimiento

1. Para empezar será necesario crear una base de datos en mysql. Si ya ha creado una obvie este paso.

2. Adicione al workspace el servidor de aplicaciones GlassFish.

3. En eclipse cree un Proyecto de tipo Maven Proyect con el arquetipo proyecto-archetype.

4. Configure la conexión a su base de datos. Si ya posee una puede omitir este paso.

5. Cambie la perspectiva de trabajo a la de JPA.

6. Abra el archivo persistence.xml. Para ello acceda a la sección Source Code de su proyecto, en ella al src/java/resources y al Metainf. De clic derecho sobre el archivo y seleccione abrir con Persistence Editor.

7. Para lograr la generación automática de las tablas seleccione la pestaña Schema Generation. Cambie la propiedad DDL Generation Type a Drop and Create Tables.

8. Cree una entidad lógica de una aplicación. Para este caso usaremos una persona con los atributos cédula, nombre y apellido. Para ello, de clic derecho sobre el proyecto y en el menú new seleccione nueva entidad. Deberá proporcionar el nombre del paquete en el que se almacenará la entidad así como el nombre de la clase a ser usada como entidad. En la siguiente pantalla adicione los atributos de la entidad, seleccione la cédula como llave. En la parte inferior podrá determinar la forma en la que desee se estructure la entidad (Fields o Properties).

9. Al manejar entidades es importante poder compararlas y decir cuando una entidad es igual a otra, por lo que es aconsejable aunque no estrictamente necesario sobreescribir el método equals de las entidades creadas. Para hacer esto, teniendo el archivo de su entidad abierta y el nombre de la clase seleccionado acceda al menú source, opción generate hashcode and equals. En la ventana que aparece verá todos los atributos de su entidad seleccionados, se sugiere dejar seleccionado solo el atributo correspondiente a la llave primaria de la entidad.

10. Es posible hacer que el servidor de aplicaciones genere las tablas de la base de datos de forma automática cuando el proyecto sea ejecutado, para ello corra el proyecto en el servidor y verifique si se creo o no la tabla en la base de datos.

11. Cree una enumeración que le permita representar el genero de una persona (masculino o femenino).

12. Adicione a su entidad un atributo que represente el genero haciendo uso de la enumeración creada previamente.

13. Ejecute nuevamente la aplicación y observe los resultados.

14. Cree una segunda entidad que represente un programa, con atributos nombre, descripción y versión. Tenga en cuenta que al no tener un atributo que lo pueda identificar de forma única deberá adicionar un atributo que lo identifique y sea generado de forma automática por el sistema.

15. Corra nuevamente la aplicación y observe los resultados.

16. Cree una tercera entidad que represente un punto en el planeta, el cual tiene como atributos la longitud, la latitud y un nombre. Tenga en cuenta que para esta entidad deberá usar una llave compuesta.

17. Corra nuevamente la aplicación y observe los resultados.

18. Ejercicio: Basado en el siguiente texto, elabore un diagrama de clases que represente el siguiente problema.

Una empresa de transporte intermunicipal le ha solicitado que diseñe e implemente una aplicación web que le permita a sus clientes comprar fácilmente los tiquetes. Para ello la empresa debe poder programar con anticipación los buses, su cupo, el trayecto (ciudades que por las que pasa) y el viaje(un bus que cubre una ruta en una fecha y hora determinada). Solo se deben vender tiquetes con un mínimo de 1 día de anticipación, si falta menos de un día, ya no se realizará venta de tiquetes en línea.

Para la empresa es vital tener buena imagen con sus clientes, por lo que es muy importante que la aplicación evite sobre vender los cupos de un bus. El cliente debe tener la posibilidad de imprimir su tiquete en cualquier momento.

Cliente:
- Debe poder registrarse en la aplicación
- Comprar tiquetes
- Imprimir tiquetes
- Buscar cupos para un destino específico en una fecha determinada.
- Recuperar contraseña en caso de haberla olvidado
- Modificar datos de contacto

Administrador:
- Gestionar datos básicos(buses, usuarios, entre otros)
- Gestionar viajes (Un bus que cubre un trayecto en una determinada fecha)
- Recuperar contraseña en caso de haberla olvidado
- Modificar datos de contacto

La gestión incluye búsqueda, actualización, creación y borrado de información.