DGTIC UNAM
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

REDEC UNAM

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
DIRECCIÓN GENERAL DE CÓMPUTO Y DE TECNOLOGÍAS
DE INFORMACIÓN Y COMUNICACIÓN

DIPLOMADO

Desarrollo de sistemas con tecnología Java

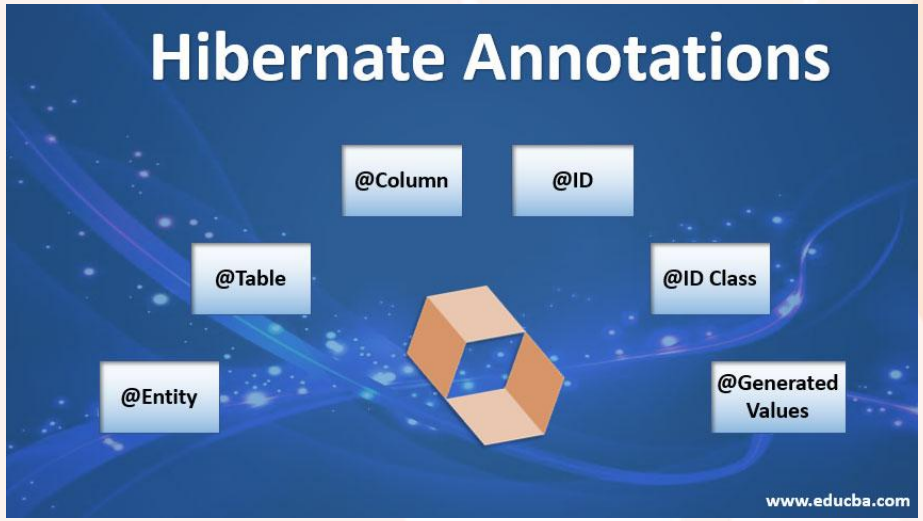
Módulo 04

Anotaciones Hibernate

Mtro. Alfonso Gregorio Rivero Duarte

Temario

1. **Anotaciones Hibernate**
   - Anotaciones Java
   - Anotaciones Hibernate



1. Anotaciones Hibernate

1.1 Anotaciones Java

Anotaciones Java

Una anotación en Java es aquella característica que nos permite incrustar información complementaria en un archivo fuente. Esta información no cambia las acciones de un programa, pero puede ser utilizada por varias herramientas, tanto durante el desarrollo como durante el despliegue o ejecución.

Anotaciones Java

Las anotaciones en Java pueden ser procesadas por un generador de código fuente, por el compilador o por una herramienta de despliegue.

En ocasiones, estas anotaciones también son llamadas metadata, pero el término anotación es el más descriptivo y más utilizado.

Anotaciones Java

¿Cuáles son sus características?

- Las anotaciones en Java comienzan con '@'.
- No cambian la actividad de un programa ordenado.
- Ayudan a relacionar metadatos (datos) con los componentes del programa, es decir, constructores, estrategias, clases, etc.
- No son comentarios sin alteraciones, ya que pueden cambiar la forma en que el compilador trata el programa.

Anotaciones Java

Ejemplos de anotaciones:

- Cuatro importados de `java.lang.annotation`:
  - `@Retention, @Documented, @Target, @Inherited`
- Cinco de `java.lang`:
  - `@Override, @Deprecated, @SafeVarargs, @FunctionalInterface, @SuppressWarnings`

Anotaciones Java

Principales Usos

- Información para el compilador:
  - El compilador puede utilizar las anotaciones para detectar errores o suprimir las advertencias.

- Procesamiento en tiempo de compilación y tiempo de implementación:
  - Las herramientas de software pueden procesar información de anotaciones para generar código, archivos XML, etc.

Anotaciones Java

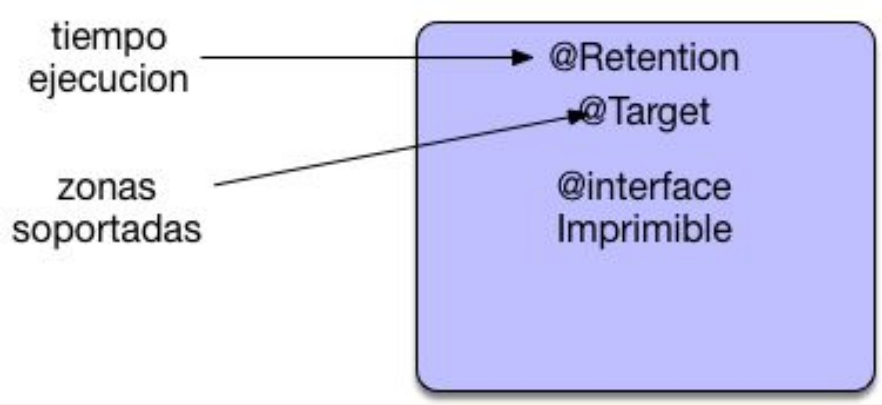
Principales Usos

- Anotación para el procesamiento de tiempo de ejecución:
  - Algunas anotaciones están disponibles para ser examinadas en tiempo de ejecución.

- Las anotaciones en Java se pueden aplicar a declaraciones:
  - Declaraciones de clases, campos, métodos y otros elementos del programa

Anotaciones Java

```java
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Book {
    public String title() default "Core Java";
}
```




```java
@Book("Spring in Action")
public class MyClass { ... }

@Book(title = "Spring in Action")
public class MyClass { ... }

@Book
public class MyClass { ... }
```

Anotaciones Java

Desde hace algunos años Java hacía un uso intensivo de XML, siendo el XML un formato de archivos demasiado largo de escribir, muy repetitivo, verboso, etc.

Esto ha llevado a crear una solución para evitar los archivos XML de persistencia en hibernate haciendo uso de Anotaciones Java en el propio código.

Estas anotaciones permiten especificar de una forma más compacta y sencilla la información de mapeo de las clases Java.

Anotaciones Java

Inicialmente Hibernate creó sus propias anotaciones en el paquete `org.hibernate.annotations` pero a partir de la versión 4 de Hibernate la mayoría de dichas anotaciones han sido `java.lang.Deprecated` y ya no deben usarse.

Las anotaciones que deben usarse actualmente son las del estándar de JPA que se encuentran en el paquete `javax.persistence`.

Las anotaciones están preconfiguradas con valores predeterminados sensibles, que reducen la cantidad de codificación requerida.

Anotaciones Java

- Configuración en **hibernate.cfg.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <!-- JDBC Database connection settings -->
    <property name="connection.driver_class">org.mariadb.jdbc.Driver</property>
    <property name="connection.url">jdbc:mariadb://localhost:3306/db</property>
    <property name="connection.username">root</property>
    <property name="connection.password">root</property>
    <!-- JDBCConnection pool settings ... using built-in test pool -->
    <property name="connection.pool_size">1</property>
    <!-- Select our SQL dialect -->
    <property name="dialect">org.hibernate.dialect.MariaDB103Dialect</property>
    <!-- Echo the SQL to stdout -->
    <property name="show_sql">true</property>
    <!-- Set the current session context -->
    <property name="current_session_context_class">thread</property>
    <!-- Drop and re-create the database schema on startup -->
    <property name="hbm2ddl.auto">create-drop</property>
    <!-- dbcp connection pool configuration -->
    <property name="hibernate.dbcp.initialSize">5</property>
    <property name="hibernate.dbcp.maxTotal">20</property>
    <property name="hibernate.dbcp.maxIdle">10</property>
    <property name="hibernate.dbcp.minIdle">5</property>
    <property name="hibernate.dbcp.maxWaitMillis">-1</property>
    <!-- entity mapping types -->
    <mapping resource="edu.unam.model.Entidad"/>
    <mapping class="Entidad"/>
  </session-factory>
</hibernate-configuration>
```

Anotaciones Java

Las entidades JPA son POJOs, por lo que en realidad, son entidades persistentes en Hibernate. Sus asignaciones se definen mediante anotaciones en lugar de archivos **hbm.xml**.

Las anotaciones se pueden dividir en dos categorías:

- Anotaciones de mapeo lógico:
  - describen el modelo de objetos, la asociación entre dos entidades.

- Anotaciones de mapeo físico:
  - describen el esquema físico, tablas, columnas, índices.

1. Anotaciones Hibernate

1.2 Anotaciones Hibernate

Anotaciones Hibernate

- `@Entity`
  - La anotación `@Entity` declara la clase como una entidad (es decir, una clase POJO persistente).

```
1 @Entity
2 public class Flight implements Serializable
```

Anotaciones Hibernate

- `@Table`
  - La anotación `@Table` impone el nivel de clase. Permite definir una tabla, el catálogo y el esquema de nombres para el mapeo de la entidad. Si no existe una anotación de este tipo se utilizan los valores por defecto: el nombre de la clase de la entidad.

```java
@Entity
@Table(name="tbl_sky")
public class Sky implements Serializable {
    ...
}
```

```java
@Table(name="tbl_sky",
    uniqueConstraints = {@UniqueConstraint(columnNames={"month", "day"})}
)
```

Anotaciones Hibernate

- `@Id`
  - La anotación `@Id` declara un propiedad como la clave principal. El valor del atributo se puede crear solo, pero Hibernate recomienda generarlo a través de Hibernate.
- `@GeneratedValue`
  - La anotación `@GeneratedValue` puede definir la estrategia de generación para el identificador

- `@Auto`
- `@Table`
- `@Identity`
- `@Sequence`

```
1 @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
2 public Long getId() { ... }
```

```
1 @GeneratedValue(generator="sequenciaDePrueba")
2 @SequenceGenerator(name="sequenciaDePrueba", sequenceName="DB_SEQUENCIA", allocationSize=1)
```

Anotaciones Hibernate

- `@Basic`
  - La anotación `@Basic` permite declarar la estrategia de búsqueda de una propiedad. Declara la propiedad persistente

```
1 @Basic
2 int getLength() { ... } // persistent property
```

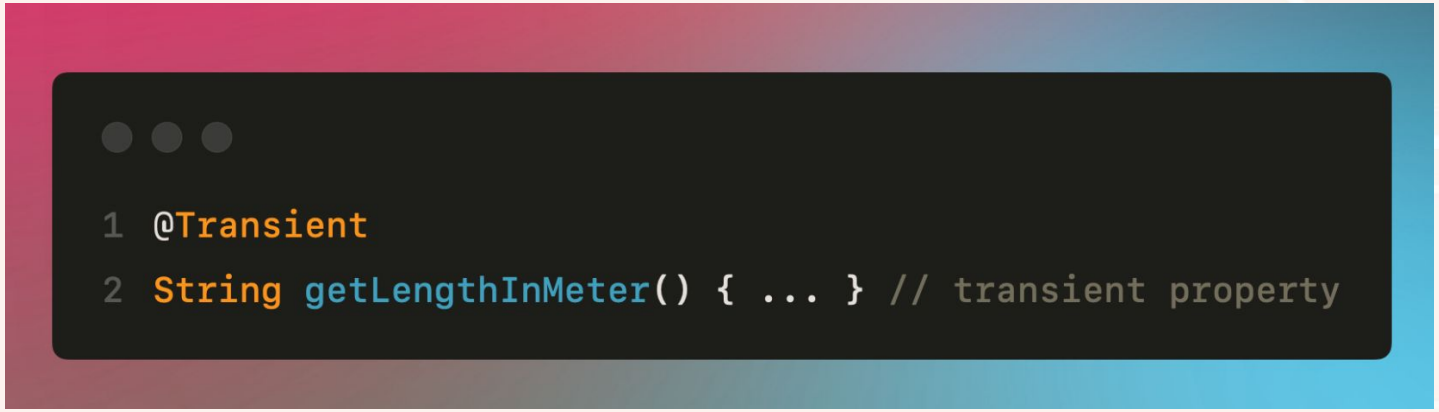
Anotaciones Hibernate

- `@Column`
  - La anotación `@Column` permite indicar las características físicas de una columna de la base de datos

```java
@Entity
public class Flight implements Serializable {
...
@Column(updatable = false, name = "flight_name", nullable = false, length = 50)
public String getName() { ... }
```

Anotaciones Hibernate

- `@Transient`
  - La anotación `@Transient` permite definir propiedades que no se hacen persistentes.



Anotaciones Hibernate

- Tipo de Dato Temporal
  - Los tipos temporales son el conjunto de tipos basados en tiempo que pueden usarse en el mapeo entidad-relación.
  - `java.sql`
    - `java.sql.Date`
    - `java.sql.Time`
    - `java.sql.Timestamp`
  - `java.util`
    - `java.util.Date`
    - `java.util.Calendar.`

Anotaciones Hibernate

- Tipo de Dato Temporal
  - Los tipos `java.util` necesitan metadatos adicionales, para indicar qué tipo JDBC `java.sql` hay que usar. Esto se consigue anotándolos con la anotación `@Temporal` y especificando el valor del tipo JDBC utilizando el valor correspondiente del tipo enumerado TemporalType.
    - DATE, TIME y TIMESTAMP representan los tipos java.sql

```
1 @Temporal(TemporalType.TIME)
2 java.util.Date getDepartureTime() { ... } // persistent property
```

Anotaciones Hibernate

- Anotaciones de mapeo lógico
  - `@OneToOne`
  - `@OneToMany`
  - `@ManyToOne`
  - `@ManyToMany`

Anotaciones Hibernate

- `@OneToOne`
  - La anotación `@OneToOne` indica la relación uno a uno de las 2 tablas.

- `@PrimaryKeyJoinColumn`
  - La anotación `@PrimaryKeyJoinColumn` indica la relación entre las dos tablas se realiza mediante la clave primaria.

```
1 @OneToOne(cascade = CascadeType.ALL)
2 @PrimaryKeyJoinColumn
3 private Direccion direccion;
```

Anotaciones Hibernate

- `@OneToMany`
  - La anotación `@OneToMany` indica que la propiedad es un conjunto de datos asociados de la clase entidad que la contiene.

```
1 @OneToMany(mappedBy = "profesor", cascade = CascadeType.ALL)
2 private Set<CorreoElectronico> correosElectronicos;
```

  - **mappedBy:** Este atributo contendrá el nombre de la propiedad Java de la clase hija que enlaza con la clase padre.
  - **cascade:** Este atributo tiene el mismo significado que el del archivo de mapeo de Hibernate **hbm.xml**

Anotaciones Hibernate

- `@ManyToOne`
  - La anotación `@ManyToOne` permite indicar que la relación es de muchos a uno, esto para definir que la relación es de lado B hacia lado A.

```
1 @ManyToOne
2 @JoinColumn(name = "IdProfesor")
3 private Profesor profesor;
```

- `@JoinColumn`
  - Con la anotación `@JoinColumn` indicaremos que en la tabla hija contiene la clave ajena a la tabla padre.
    - **name:** nombre de la columna que se encuentra en la clase hija que enlaza con la clase padre

Anotaciones Hibernate

- `@ManyToMany`
  - La anotación `@ManyToMany` indica que la propiedad contiene una lista de objetos que participan en una relación muchos a muchos.

```java
@ManyToMany(cascade = {CascadeType.ALL})
@JoinTable(name="ProfesorModulo", joinColumns={@JoinColumn(name="idProfesor")}, inverseJoinColumns={@JoinColumn(name="idModulo")})
private Set<Modulo> modulos = new HashSet();
```

Anotaciones Hibernate

- `@JoinTable`
  - Esta anotación contiene la información sobre la tabla que realiza la relación muchos a muchos
  - **name:** Nombre de la tabla intermedia
  - **joinColumns:** contiene cada una de las columnas de la tabla intermedia
  - **inverseJoinColumns:** Contiene cada una de las columnas que forman la clave primaria de la clase clase con la que tenemos la relación.

Anotaciones Hibernate

- `@ManyToMany`

```
1 @ManyToMany(cascade = {CascadeType.ALL}, mappedBy="modulos")
2 private Set<Profesor> profesores = new HashSet();
```

  - **cascade:** Este atributo tiene el mismo significado que el del fichero de mapeo de Hibernate.
  - **mappedBy:** Contiene el nombre de la propiedad Java de la otra clase desde la cual se relaciona con ésta.

Contacto

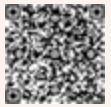
Mtro. Alfonso Gregorio Rivero Duarte

Senior Data Manager - CBRE

devil861109@gmail.com

Tels: (+52) 55 289970 69

Redes sociales:



https://www.linkedin.com/in/alfonso-gregorio-rivero-duarte-139a9225/