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

Información general	
ACTUALIZADO:	Carlos Andrés Flórez Villarraga
DURACIÓN ESTIMADA EN MINUTOS:	90
DOCENTE:	Christian Andrés Candela y Einer Zapata G
GUÍA NO.	05
Nombre de la guía:	Parametrización de atributos en una entidad

Información de la Guía

OBJETIVO

Aprender a parametrizar de forma adecuada los atributos de una entidad.

CONCEPTOS BÁSICOS

Manejo de Eclipse, Java, Bases de Datos, JDBC, XML, Glassfish.

CONTEXTUALIZACIÓN TEÓRICA

Se puede restringir, validar o limitar la información a ser almacenada en un determinado atributo de una entidad. Esto se logra mediante el uso de anotaciones que permiten indicar al proveedor JPA los elementos a tener en cuenta para un determinado atributo.

Una de las anotaciones básicas para la manipulación de un atributo es la anotación `@Column`. En primera instancia la anotación `@Column` nos permite especificar el nombre que se le dará al campo a nivel de base de datos. Ejemplo:

```
@Column(name="mi_campo")
private int miCampo;
```

En este caso, name nos permite indicar al proveedor JPA que el nombre con el cual será conocido el atributo `miCampo` a nivel de base de datos es `mi_campo`. De igual forma se puede especificar si el atributo podrá o no ser nulo, si debe o no ser único, si se puede o no insertar, actualizar, la longitud o tamaño del campo, la precisión (número de decimales) y la escala o número de dígitos. Todo esto a través del uso de atributos en la notación `@Column` así:

- `unique`: Indica si el campo es o no único.

  ```
  @Column(name="mi_campo", unique=true)
  ```

- `nullable`: Permite definir si el campo puede o no ser nulo. Es decir, si se debe llenar obligatoriamente o no.

  ```
  @Column(name="mi_campo",nullable=false)
  ```

- `length`: Permite definir la longitud máxima del campo. Es usado en los campos de tipo string.

  ```
  @Column(name="mi_campo", length=29)
  ```

- `precision`: Determina el número de dígitos de la parte entera que puede tener un campo de tipo BigDecimal.

  ```
  @Column(name="mi_campo", precision=8)
  ```

- `scale`: Determina el número de dígitos de la parte decimal que puede tener un campo de tipo BigDecimal.

  ```
  @Column(name="mi_campo", precision=8, scale=2)
  ```

- `insertable`: Determina si el campo podrá o no ser insertado.

  ```
  @Column(name="mi_campo", insertable=true)
  ```

- `updatable`: Indica si el campo puede o no ser actualizado.

  ```
  @Column(name="mi_campo", updatable=true)
  ```

- `columnDefinition`: Se usa para especificar opciones más complejas en los campos.

  ```
  @Column(name="timestamp", columnDefinition="TIMESTAMP DEFAULT CURRENT_TIMESTAMP")
  ```

Se debe tener en cuenta que elementos como la longitud, no debe ser usada en atributos con tipo de dato complejo. De igual forma no tendría sentido usar el atributo precisión en un campo de tipo `String`. El uso inadecuado de este tipo de elementos en un campo que no lo soporta puede causar errores en la aplicación.

De forma similar a la anotación `@Column`, se puede hacer uso de la anotación `@JoinColumn`, la misma es usada sobre atributos cuyo tipo es complejo (otra entidad o lista). A diferencia del `@Column`, el `@JoinColumn` solo permite definir la nulidad de un campo y si el mismo es o no único.

Además de esta anotación, también podemos hacer uso de otras que permiten de forma fácil establecer restricciones, sin embargo, estas otras restricciones trabajan a nivel lógico no físico como si lo hace la anotación `@Column`. Entre las más comunes tenemos:

- `@NotNull`: Indica si un campo puede o no ser nulo y el mensaje de error en caso de que el campo no cumpla con la restricción.
- `@Size.max`: Permite indicar el número máximo de caracteres de un `String`.
- `@Min`, `@Max`: Permite indicar el máximo o mínimo valor a ser asignado a un campo numérico.
- `@Digits`: Permite indicar el la precisión y la escala de un campo numérico.
- `@Lob`: Permite indicar que el motor de base de datos debe elegir el tipo de dato más grande disponible.

PRECAUCIONES Y RECOMENDACIONES

Al crear atributos que representan fechas recuerde usar la anotación `@Temporal` para especificar si el atributo representa una fecha, una hora o una fecha y una hora.

ARTEFACTOS

Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse para JEE en su última versión), 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 crear entidades estableciendo adecuadamente las restricciones y configuraciones a cada uno de los atributos de la entidad.

Procedimiento

1. Para el desarrollo de esta guía necesitará una base de datos en MySQL, un proyecto de tipo Maven con soporte para el uso de JPA, otro proyecto Maven para pruebas, y una conexión a dicha base de datos para ser usada en la generación de las tablas.

2. Si no lo ha hecho previamente, adicione a su proyecto de persistencia la siguiente dependencia.

```
<dependency>
	<groupId>javax</groupId>
	<artifactId>javaee-api</artifactId>
	<version>8.0</version>
	<scope>provided</scope>
</dependency>
```

3. Para cada uno de los atributos de las entidades creadas en la guía de Entidades aplique las validaciones y parametrizaciones que considere pertinentes mediante el uso de la anotación `@Column`. Asegúrese que al menos uno de los campos de alguna entidad sea de tipo `Date`. Haga que la llave primaria de alguna entidad sea autogenerada, añadiendo la anotación `@GeneratedValue`.

4. Genere las tablas y verifique los resultados en su base de datos.

5. Cree un método en la clase `ModeloTest` del proyecto de pruebas donde pueda verificar las restricciones y validaciones creadas en al menos una entidad. Ejemplo:

```
@Test
@Transactional(value=TransactionMode.COMMIT)
public void probarPersistencia(){

	Persona p = new Persona();
	p.setCedula("14785236");
	p.setNombre("Pepito");
	p.setApellido("Perez");
	p.setFechaNacimiento(new Date());
	p.setGenero(Genero.Masculino);

	entityManager.persist(p);
}
```

6. Ejecute la prueba unitaria. De click derecho al nombre del método y luego Run as - JUnit Test.

7. Cree un segundo conjunto de entidades equivalentes a las de la guía de entidades, pero en esta ocasión parametrícelo y valídelo sin hacer uso de la anotación `@Column`.

8. Genere las tablas y verifique los resultados en su base de datos.

9. Tome su documento de identidad y cree una entidad que represente todos y cada uno de los atributos que en él encuentre. Adicione las restricciones y parametrizaciones que considere necesarias.

10. Por último, cree un nuevo proyecto usando las entidades que ha diseñado para su proyecto final y adicione a cada uno de los elementos básicos las restricciones y parametrizaciones que considere necesarias.

Para la próxima clase

Leer sobre los tipos de relaciones en bases de datos.