En el articulo anterior de manejo de anotaciones java vimos como crear nuestras propias anotaciones .Sin embargo falto el código de como una vez construidas estas anotaciones podíamos usarlas en nuestro código .Vamos a cubrir esta parte en este articulo. Para ello lo primero que vamos a hacer es declarar una anotación denominada @Persistencia.

```
package es.curso;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(value={ElementType.FIELD} )
@Retention(RetentionPolicy.RUNTIME)
public @interface Persistencia {

String campo();
 String tipo();

}
```

La anotación en si queda marcada por otras dos anotaciones . Es algo que nos puede parecer extraño pero es habitual en el mundo de las anotaciones .Existen anotaciones que solo sirven para marcar a otras anotaciones .

- @Target : Esta anotación sirve para delimitar en que partes de nuestro código podemos

utilizar la anotación de @Persistencia .En este caso limita a unicamente los campos de la clase.

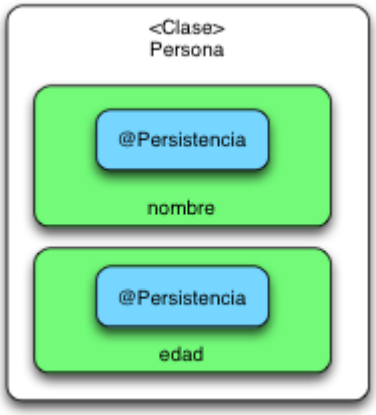
- @Retention :Esta anotación sirve para asegurarnos que la información que nos provee la anotacion @Persistencia esta disponible en tiempo de ejecución .Existen otras anotaciones que solo estan disponibles en tiempo de compilación por ejemplo.
  Una vez que hemos construido nuestra anotación podemos utilizar una clase para probarla.

```
package es.curso;
public class Persona {

@Persistencia(campo = "nombre", tipo = "varchar")
 private String nombre;

 @Persistencia(campo="edadPersona",tipo="decimal")
 private String edad;
 public String getNombre() {
 return nombre;
 }
 public void setNombre(String nombre) {
 this.nombre = nombre;
 }
 public String getEdad() {
 return edad;
 }
 public void setEdad(String edad) {
 this.edad = edad;
 }
}
```

Como podemos ver hemos utilizado la anotación a nivel de la propiedades “nombre” y “edad” ya que debido a la anotación @Target no podría ser utilizadas
en otro sitio.



Nos queda por ver como accedemos en tiempo de ejecución a la información que almacenan las anotaciones .Para ello utilizaremos el API de Reflection de Java

```
package es.curso;

import java.lang.reflect.Field;

public class PrincipalAnotaciones {

 public static void main(String[] args) {

Persona p = new Persona();
 Class clasePersona = p.getClass();
 Field[] campos= clasePersona.getDeclaredFields();
 for(Field campo: campos) {

 Persistencia persistencia=campo.getAnnotation(Persistencia.class);
```

```
 System.out.println("Nombre :" +persistencia.campo());
 System.out.println("Tipo :"+ persistencia.tipo());

 }

}

}
```

El código no imprimirá por pantalla la información almacenada en las anotaciones.

```
Nombre :nombre
Tipo :varchar
Nombre :edad
Tipo :decimal
```