Paginación de consultas

Las interfaces Query y TypedQuery proporcionan una serie de métodos que permiten paginar los resultados de una consulta, a fin de poder tratarlos por bloques en lugar de todos a la vez.

Estos métodos son los siguientes:

- Query setFirstResult(int pos). Establece la posición del primer objeto a devolver del conjunto. La posición del primer del primer objeto del conjunto es 0.
- Query setMaxResult(int max). Establece el máximo número de resultados a devolver por la consulta.

Los métodos anteriores se deben establecer sobre el objeto Query o TypedQuery antes de llamar a *getResultList()*, pues la llamada a este método se verá condicionada por los valores establecidos en dichos métodos. De hecho, vemos que ambos métodos devuelven un nuevo objeto Query configurado para obtener los resultados según el valor establecido en la llamada al método.

Por ejemplo, si *tam* es una variable que contiene el tamaño de página y *q* es la variable que apunta al objeto Query, el siguiente método nos devolvería el conjunto de entidades Producto de la página indicada como parámetro:

```
public List<Producto> obtenerProductosPorPagina(int pag){

        //calculamos el total de páginas

        double totalEntidades=q.getResultList().size();

        int paginas=(int)Math.ceil(totalEntidades/tam);

        //si el número de página es superior a las existentes, devolvemos nulo

        if(pag>paginas){

                return null;

        }else{

                return q.setFirstResult((pag-1)*tam).setMaxResult(tam).getResultList();

        }

}
```

Vamos a comentar un poco las instrucciones anteriores. En primer lugar, tenemos la línea:

```
double totalEntidades=q.getResultList().size();
```

Lo que hacemos en esta instrucción es calcular el número total de entidades que devolvería la consulta asociada al objeto query que vamos a paginar. Lo almacenamos en una variable de

tipo *double* porque después vamos a realizar una división con ella, y nos interesa la parte decimal.

Seguidamente, tenemos la instrucción:

int paginas=(int)Math.ceil(totalEntidades/tam);

Lo que hacemos primeramente es calcular el número de páginas, para lo que dividimos el total de entidades por el tamaño de página. Si el resultado no es entero, por ejemplo, 3.2 o 3.7, el total de páginas será uno más que el valor entero de la división (en los ejemplos anteriores, sería 4), de ahí que apliquemos el método *ceil*, que nos da el entero superior más cercano a la división. Si el resultado de la división fuera 2.0, la llamada a *ceil* retornaría también 2.0, no habría que sumar uno al valor entero de la división porque directamente habría dos páginas.

Finalmente, comprobamos si la página solicitada es superior a las existentes, porque en ese caso no hay que devolver ningún resultado. Si la página solicitada está dentro de las existentes, tenemos la instrucción:

return q.setFirstResult((pag-1)*tam).setMaxResult(tam).getResultList();

La llamada a *setFirstResult* devuelve un objeto Query, de ahí que al resultado le apliquemos directamente *setMaxResult* que devuelve el objeto Query final configurado con la posición del primer resultado y el máximo de resultados a obtener. A este objeto Query final es al que le aplicamos el *getResultList().*