# *Patrón Object Pool. Explicación del código.*

## 1.1. Piscina de Conexiones

En el mundo de la informática, una de las cosas más caras con las que nos encontramos es la conexión a base de datos. Muchas bases de datos y servidores de aplicaciones comparten conexiones por medio de las connection pools, o piscinas de conexiones. A continuación vamos a mostrar cómo implementar una piscina basada en el patrón Object Pool y desarrollaremos una piscina de conexiones JDBC como ejemplo.

Este patrón se basa en tres clases, una clase `Reutilizable`, una `Piscina-Reutilizable`, y un `Cliente`.

Los objetos de la clase `Reutilizable` son aquellos que queremos compartir.

La clase de `Piscina-Reutilizable` implementa la piscina y contiene, un número de objetos `Reutilizables`. Los objetos pueden ser adquiridos y devueltos por los objetos `Cliente` haciendo peticiones de la `Piscina-Reutilizable`.

Un aspecto importante sobre la `Piscina-Reutilizable` es que se implementa como un Singleton. La Piscina-Reutilizable siempre tiene cardinalidad 1 en cualquier asociación en la que participe. Además, el constructor es privado, y hay un método `getInstancia`(). Todas estas características, como ya se vio anteriormente en detalle, son propias de patrón Singleton.

Los objetos de la clase `Cliente` harán peticiones de la `piscina-Reutilizable` para conseguir acceso a los objetos `Reutilizables`. Puede haber cualquier número de `Clientes` y cada `Cliente` puede acceder a cualquier número de objetos `Reutilizables`, hasta alcanzar el tamaño máximo de la piscina.

### Funcionamiento de la Aplicación

Nuestra aplicación, o cualquier programa que requiera de conexiones a base de datos, puede hacer uso de un objeto `JDBCConexion` que encapsule la lógica de la base de datos. El string `nombreBD` (nombre de la base de datos), que es un identificador único para una base de datos en particular, es usado como un parámetro cuando se adquieren objetos Reutilizables `JDBCConexionImpl`. Cada objeto `JDBCConexion` puede acceder a un máximo de un objeto `JDBCConexionImpl`.

El procedimiento es el siguiente. Cuando se realiza una petición, el objeto `JDBCConexion` pide un objeto `JDBCConexionImpl` de la Piscina de Conexiones (`JDBCPool`). El objeto `JDBCConexion` la utiliza y la devuelve inmediatamente a la piscina. Si la piscina está vacía cuando la petición es realizada, se creará un nuevo objeto `JDBCConexionImpl`. Si se ha alcanzado el tamaño máximo de la piscina, la piscina no será capaz de crear un nuevo objeto `JDBCConexionImpl`, y pueden ocurrir varias cosas.

La primera opción es la de tener al cliente sentado y esperando hasta que un objeto `ConnectionImpl` sea devuelto a la piscina. Esto afectaría al rendimiento de la aplicación pero no requeriría ningún tipo de desarrollo extra. Por otra parte se podría hacer que el método `obtenerImpl`() generase una excepción. De este modo el cliente no perdería tiempo esperando, pero habría que introducir código para controlar y manejar la excepción.

## 1.2. JDBCConexion

Las implementaciones de las piscinas de conexiones, requieren que el cliente pida una conexión a base de datos, la use, y después la de de baja cuando ya no le sea necesaria. El cliente tiene que encargarse de introducir en su aplicación el código necesario para llevar a cabo estas acciones, de alta y baja. El problema aparece entonces si el cliente nunca da de baja una conexión, o alarga la baja por un excesivo período de tiempo. La clase `JDBCConexion` aparece entonces para encapsular todo este comportamiento, haciendo la codificación más sencilla, y más segura para la piscina.

`JDBCConexion` es la pasarela entre la aplicación y la base de datos. Cuando se crea una instancia de `JDBCConexion` por primera vez, ésta captura una referencia a una piscina de conexiones descrita por `nombreBD`, un parámetro que identifica únicamente la base de datos. Este parámetro corresponderá con aquellos que sean necesitados por el constructor de `JDBCConexionImpl`. Si se utilizasen múltiples bases de datos, sería necesario el nombre de la base (`nombreBD`), junto con su correspondiente sentencia en el constructor de `JDBCConexionImpl`.

Creando un objeto de tipo `JDBCConexion`, el cliente obtiene una conexión compartida a la base de datos. Debido a esta forma de instanciación (`Lazy Instantiation`), la verdadera conexión a base de datos no será creada hasta que vaya a ser usada.

La aplicación envía una petición, en la forma de una cadena SQL, a la base de datos por medio del método `enviarPetición().` El método `enviarPetición(),` primero hace una comprobación par ver si la conexión todavía está en uso. Si lo está, la cierra.

A continuación, se obtiene de la piscina la verdadera conexión a la base de datos, mediante el método `obtenerImpl(),` se ejecuta la sentencia SQL, y el resultSet se pone a disposición de la aplicación que efectúa la llamada.

La aplicación `Cliente`, puede obtener una referencia al resultSet por medio del método `getRs(),` y hacer con él todo el procesamiento que requiera.

Como el resultSet no se copia en ningún momento para pasárselo al cliente, la conexión debe mantenerse abierta hasta que el cliente haya terminado con ella. Cuando el procesamiento realizado con el resultSet, se haya completado, se realiza una llamada al método `cerrarPeticion(),` que se encargará de liberar los recursos de la base de datos y de devolver la conexión a la piscina.

El flag `enUso` es utilizado para determinar si la conexión está en uso o no. Si el cliente hace otra llamada a `enviarPeticion()` sin haber llamado primero a `cerrarPeticion(),` el método `cerrarPeticion()` será llamado automáticamente, liberando los recursos de las peticiones anteriores antes de realizar una nueva.

## 1.3. JDBCConexionImpl

`JDBCConexionImpl` es la clase en la que se crean los objetos de tipo Connection reusables.

El constructor de esta clase es privado de forma que nadie fuera de la clase pueda crear una conexión por sí mismo. Como resultado, la clase `JDBCPool`, encargada de la piscina de objetos, debe ser un miembro de `JDBCConexionImpl` de forma que pueda tener acceso a su constructor.

El objetivo de esta clase es la de crear una conexión a una base de datos concreta Access y proporcionar los métodos para acceder a ella.

## 1.4. JDBCPool

Es la clase encargada de crear la piscina de conexiones. Es una clase estática que se encuentra dentro de `JDBCConexionImpl` con el objetivo de tener acceso a su constructor.

La clase `JDBCPool` se implementa haciendo uso del patrón Singleton. Como característica propia de este patrón tiene un constructor privado vacío para que su uso esté controlado. Además tiene una variable de clase de tipo `JDBCPool`.

`tablaPiscinas` es una tabla `Hash` que se va a encargar de guardar un diccionario de nombres de BBDD con su correspondiente vector de conexiones.

`maxTamPiscina` y `contObjPiscina` son el número de objetos máximo que permite la piscina, y el contador de objetos actuales de la piscina, respectivamente.

El método `getInstancia(),` propio del Singleton, se va a encargar de devolver una referencia a la clase. Si es la primera llamada que recibe el método, y el objeto de tipo `JDBCPool` no ha sido creado, se crea mediante la sentencia `new`. En caso contrario, se devuelve una referencia al mismo.

Cuando se realiza una petición para conseguir una conexión a base de datos por medio del método `obtenerImpl(),` el objeto `JDBCPool` buscará el nombre de la base de datos (nombreBD) que se le ha proporcionado en un diccionario privado para obtener el correspondiente vector que contiene los objetos `JDBCConexionImpl` disponibles.

La primera vez que esta petición se realice, el vector estará vacío. Como no existirá ningún objeto de conexión, se creará un nuevo objeto `JDBCConexionImpl` y se devolverá. Si el vector no está vacío, se sacará un objeto `JDBCConexionImpl` del mismo, y se devolverá al objeto llamante `JDBCConexion`.

Cuando `JDBCConexion` ha terminado con el objeto `JDBCConexionImpl`, éste es devuelto a la piscina por medio del método `devolverImpl().` El objeto `JDBCConexionImpl` recuerda el `nombreBD` de la base de datos a la que pertenece. Este nombre de la base (`nombreBD`) se usa para devolver el objeto al vector adecuado que contiene los objetos de conexión disponibles.

Si éste es el primer objeto devuelto, el vector será null. Entonces se creará un nuevo vector que contendrá el objeto devuelto, y será añadido al diccionario. Si el vector ya tiene objetos disponibles dentro de él, existe, el objeto devuelto, simplemente se añade, y queda listo para ser reutilizado.

### 1.4.1 ErrorPiscina

Decíamos que cuando un cliente pide una conexión, haciendo una llamada al método `obtenerImpl(),` si hay una conexión disponible, ésta le es devuelta. Si no hay conexiones disponibles, es necesario crear una nueva conexión.

Si el máximo de conexiones no se ha alcanzado, se crea una nueva. Si por el contrario, se ha alcanzado el número máximo de conexiones, se lanza una excepción de tipo `ErrorPiscina` y se le devuelve al cliente.