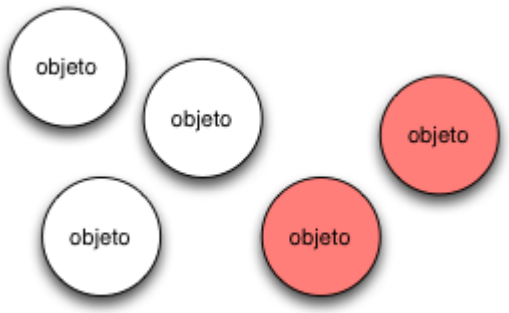
El bloque de Try with Resources en Java es una de los bloques de código más útiles que nos podemos encontrar cuando trabajamos con código que requiere la gestión de excepciones y el cierre de recursos. Vamos a ver un ejemplo de como funciona este bloque. Todos hemos trabajado con JDBC en algún momento. Puede ser que hoy en día estemos usando JPA u otra cosa pero a veces toca volver, todo depende de los diferentes proyectos en los que estemos trabajando. . "Java Try with Resources" fue una de las novedades que trajo Java 7 y que nos permite una mejor gestión del cierre de recursos a nivel de los objetos que construimos . Normalmente en cualquier programa Java tenemos una serie de objetos que se construyen.



Mientras que la mayoría de ellos se pueden simplemente destruir después de haberlos usado es decir cuando no hay referencias a ellos el recolector de basura parasá y se encargara de eliminarlos ,hay algunos otros que necesitan liberar una serie de recursos antes de ser destruidos .Ejemplo de este caso son Readers a nivel de Java IO o Connections , Statements a nivel de JDBC esto hace que el código sea a veces complicado de gestionar. El siguiente ejemplo muestra la casuística con JDBC.

```
Connection connection = null;
PreparedStatement pStatement = null;
ResultSet resultSet = null;

try {
    connection = getConnection();
    pStatement = connection.prepareStatement(sql);
    resultSet = pStatement.executeQuery();
```

```
    while (resultSet.next()) {
        // Gestionar ResultSet
    }
} catch (SQLException e) {
    throw new MiExcepcion("Ocurrió un error al ejecutar la consulta.",
e);
} finally {
    if (resultSet != null) {
        try {
            resultSet.close();
        } catch (SQLException e) {
            // Manejar el cierre del ResultSet
        }
    }

    if (pStatement != null) {
        try {
            pStatement.close();
        } catch (SQLException e) {
            // Manejar el cierre del PreparedStatement
        }
    }

    if (connection != null) {
        try {
            connection.close();
        } catch (SQLException e) {
            // Manejar el cierre de la conexión
        }
    }
}
```

Try with Resources al rescate

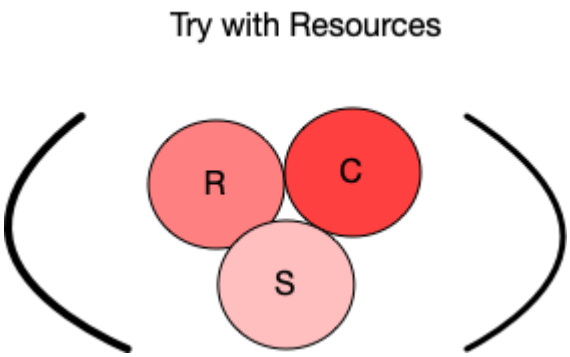
El código es realmente engorroso y duro de gestionar , nos vemos obligados a cerrar cada recurso de forma independiente y hacerlo con mucho cuidado . Gracias a las novedades que llegaron en Java 7 y a la estructura de Try with resources podemos simplificarlo de forma clara:

```
try (Connection connection = getConnection();
     PreparedStatement pStatement = connection.prepareStatement(sql);
     ResultSet resultSet = pStatement.executeQuery()) {

    while (resultSet.next()) {
        // Gestionar ResultSet
    }

} catch (SQLException e) {
    throw new MiExcepcion("Ocurrió un error al ejecutar la consulta.",
e);
}
```

El nuevo código es mucho mas compacto y mucho mas claro ya que permite gestionar todos los recursos de forma automática envolviendolos en un pareja de paréntesis .



Esta modificación se puede utilizar con cualquier clase que implemente el interface

AutoCloseable.

```
public interface AutoCloseable {
    void close();
}
```

Ha día de hoy en Java son muchos ya que el uso de Try with Resources ya que se ha ido standarizando.Las ventajas son evidentes y el código es mucho mas sencillo de gestionar.

Otros artículos relacionados:

- JDBC ResultSet Types y su funcionamiento
- Creando un Java 8 Custom Stream con JDBC ResultSets
- Spring MVC static resources y su configuración
- REST Resources Nombres vs Verbos