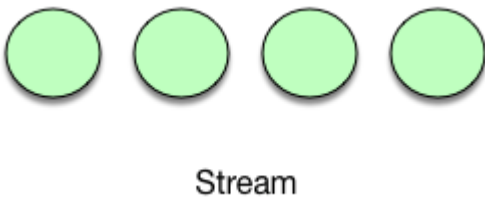
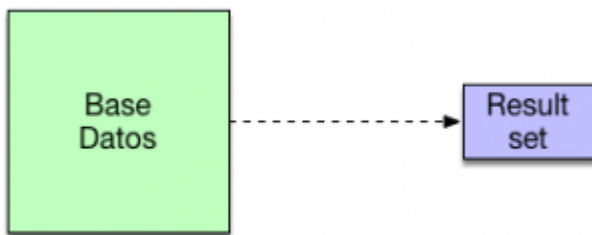


El concepto de Java 8 Custom Stream es un concepto que en más de una ocasión necesitaremos abordar. En la programación más clásica de Java 8 utilizamos el método `.stream()` y convertimos una lista en un Stream de objetos. Sin embargo hay situaciones que no son tan directas y en las cuales necesitaremos construir nuestro propio stream desde cero. ¿Cómo podemos construir un Java 8 Custom Stream?



Vamos a ver un ejemplo que se apoye en un ResultSet. El primer paso es crear un método que construya un ResultSet a partir de una conexión a base de datos



Veamos el código:

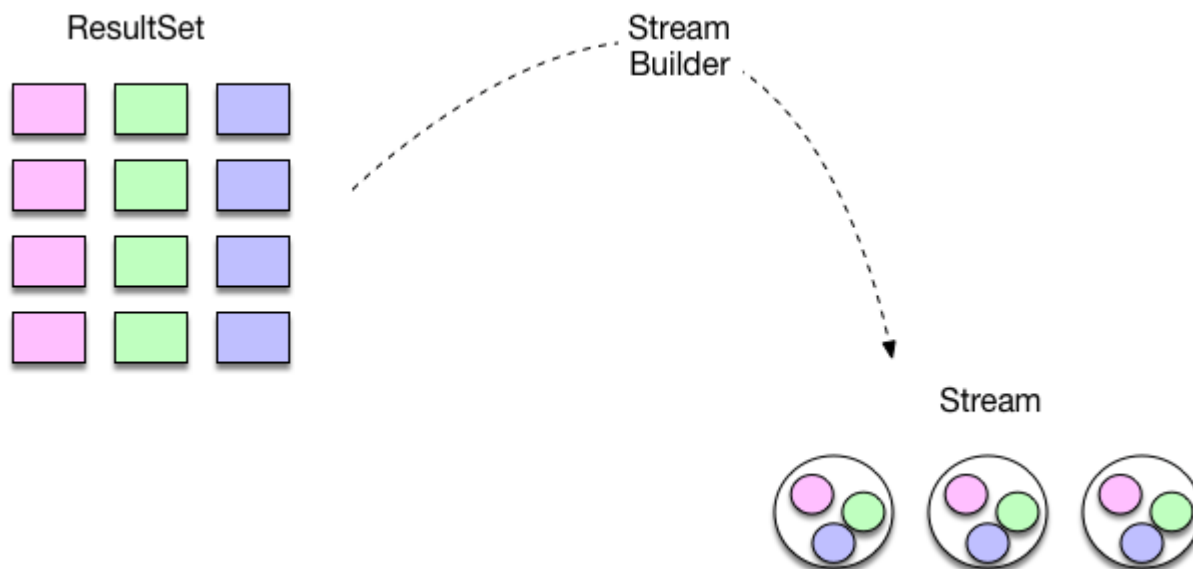
```
private static ResultSet crearResultSet() {  
    ResultSet rs = null;  
    String cadenaConexion =  
"jdbc:mysql://localhost:3306/curso?useUnicode=true&useJDBCComplian
```

```
tTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";  
        try {  
            Connection conexion =  
DriverManager.getConnection(cadenaConexion, "root", "root");  
            Statement sentencia =  
conexion.createStatement();  
            rs = sentencia.executeQuery("select * from  
persona");  
        } catch (SQLException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
  
        return rs;  
    }  
}
```

## Java 8 Custom Stream

Este método nos devuelve un ResultSet con el cual podemos trabajar de la forma clásica. Sin embargo vamos a utilizar otro método que nos permita convertir el ResultSet en un Stream.

## Creando un Java 8 Custom Stream con JDBC ResultSets



Veamos el código que usa un `Stream Builder` (design pattern)

```
public static Stream<Persona> crearStream(ResultSet rs) throws
SQLException

{
    Stream.Builder<Persona> builder = Stream.builder();
    while (rs.next()) {
        Persona persona = convertirFila(rs);
        builder.add(persona);
    }
    return builder.build();
}
```

En este método estamos convirtiendo un `ResultSet` a un `Stream` invocando el método `next()`

y creando un `Stream.Builder`. El `Stream.Builder` dispone de un método `add` que añade los objetos al futuro `Stream`. Por último invocamos al método `build()` y el nuevo `Stream` se genera. El `Stream Builder` se apoya en un sencillo método que convierte la fila del `ResultSet` en un objeto `Persona`.

```
public static Persona convertirFila(ResultSet rs) throws SQLException
{
    return new Persona(rs.getString("nombre"),
rs.getString("apellidos"), rs.getInt("edad")
    );
}
```

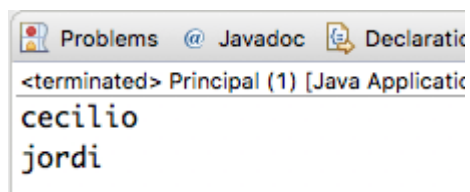
Una vez tenemos todos los métodos contruidos es momento de construir el método `main` que se encarga de usar el `Stream` e imprimirlo por la consola.

```
public static void main(String[] args) {
    try {
        Stream<Persona>
stream=crearStream(crearResultSet());

        stream
        .map(p->p.getNombre())
```

```
        .forEach(System.out::println);
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Acabamos de construir un Stream desde cero .En este caso para probar el Java 8 custom Stream nos estamos apoyando en el método map imprimiendo el nombre. El resultado se muestra en la consola.



Otros artículos relacionados:

1. [Java Stream Sum y Business Objects](#)
2. [Java Stream Context y simplificación de Streams](#)
3. [Java Stream Filter y Predicates](#)
4. [Programación Funcional, Java 8 Streams](#)
5. [Utilizando un JPA Stream con JPA 2.2](#)
6. [Java Stream map y estadísticas](#)

Externos

1. [Java Streams Oracle](#)