INTERACCIÓN CON REDIS MEDIANTE JAVA

Redis nos ofrece gran cantidad de clientes para interactuar con él mediante gran cantidad de lenguajes de programación (Clientes Redis (http://redis.io/clients)).

En nuestro ejemplo interactuaremos con Redis mediante el cliente Java denominado Jedis (https://github.com/xetorthio/jedis).

DEPENDENCIAS MAVEN

Comenzaremos creando un proyecto Maven incluyendo la depedencia siguiente en el `pom.xml` .

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.6.2</version>
    <type>jar</type>
    <scope>compile</scope>
</dependency>
```

No obstante, si aún no usas Maven y prefieres incluir el driver en el proyecto puedes descargar el driver .jar de Redis (http://mvnrepository.com/artifact/redis.clients/jedis).

CONEXIÓN A REDIS

La forma básica de obtener una conexión a Redis es instanciando directamente la clase `Jedis` . Indicaremos el `host` y usaremos el puerto por defecto

```
Jedis jedis = new Jedis("localhost");
```

Otra forma básica de conexión a Redis consiste en instanciar la clase `Jedis` especificando el `host` y el puerto de conexión

```
Jedis jedis = new Jedis("localhost", 6379);
```

Sin embargo, no se recomienda utilizar la misma instancia desde threads diferentes, ya que podemos obtener errores inesperados. Además, tampoco es adecuado la creación de múltiples instancias de Jedis ya que acarrea la creación de múltiples sockets y conexiones, lo que también puede desencadenar errores. Esto se debe a que una instancia de Jedis no es *threadsafe*.

Para evitar estos problemas se debe utilizar `JedisPool`, un pool de conexiones (http://es.wikipedia.org/wiki/Connection_pool) *threadsafe*. Utilizaremos el pool para crear de forma segura instancias de Jedis. De esta forma evitaremos errores y obtendremos un mejor rendimiento.

Iniciaremos un pool:

```
JedisPool pool = new JedisPool(new JedisPoolConfig(), "localhost");
```

`JedisPoolConfig` incluye una serie de valores predeterminados útiles. Por ejemplo, si usamos Jedis con `JedisPoolConfig` se liberará la conexión tras 300 segundos (5 minutos) de inactividad.

La conexión a Redis la obtendremos con el método `getResource()` de `JedisPoolConfig`

```
Jedis jedis = pool.getResource();
```

Cuando no ya no necesitemos una conexión la cerraremos con `close()` y quedará liberada.

```
jedis.close();
```

Cuando hayamos finalizado el trabajo destruiremos el pool con `destroy()`.

```
pool.destroy();
```

```
1   public class Redis {
2           // Threadsafe pool of network connections
3           JedisPool pool;
4
5           // Redis connection
6           Jedis jedis;
7
8           /**
9            * Returns a direct connection to a Redis database in localhost
10           * using the default port.
11           *
12           * @return      a connection to Redis
13           * @see         Image
14           */
15          public Jedis getDirectConnection() {
16                  jedis = new Jedis("localhost");
17
18                  return jedis;
19          }
20
21          /**
22           * Close a connection to a Redis database
23           *
24           */
25          public void closeDirectConnection() {
```

```
26                  if (jedis != null) {
27                          jedis.close();
28                  }
29          }
30
31          /**
32           * Returns a connection from a pool to a Redis database in
33           * localhost using the default port
34           *
35           * @return      a connection from a pool to Redis
36           */
37          public Jedis getConnection() {
38                  pool = new JedisPool(new JedisPoolConfig(), "localhost");
39
40                  jedis = pool.getResource();
41
42                  return jedis;
43
44          }
45
46          /**
47           * Destroys a pool of network connections and close the connection
48           *
49           */
50          public void destroyPool() {
51
52                  // Close the connection
53                  if (jedis != null) {
54                          jedis.close();
55                  }
56
57                  // Destroy the pool
58                  if (pool != null) {
59                          pool.destroy();
60                  }
61          }
62  }
```

view raw (https://gist.github.com/ualmtorres/021029e1cf1ca5a0c475/raw/c300204151d0699a923e3a5dd7406e7ae5686827/jedis-Connecting.java)
jedis-Connecting.java (https://gist.github.com/ualmtorres/021029e1cf1ca5a0c475#file-jedis-connecting-java) hosted with ♡ by GitHub (https://github.com)

SET, GET, MSET, MGET Y DEL

Con `set(key, value)` asignamos un valor a una clave y con `get(key)` recuperamos el valor asociado a una clave.

Con `mset(key1, value1 [, key n, value n])` asignamos una lista de pares clave-valor en una operación atómica. Con `mget(key1 [, key n])` obtenemos la lista de valores asociados a las claves pasadas como parámetro.

Con `del(key1 [, key n])` borramos una clave o lista de claves.

```
1   public void strings() {
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
5           // SET and GET
6           jedis.set("foo", "bar");
7           String value = jedis.get("foo");
8           System.out.println(value);
9
10          // MSET and MGET
11          jedis.mset("a", "10", "b", "20", "c", "30");
12          List<String> values= jedis.mget("a", "b", "c");
13          for (String v: values) {
14                  System.out.println(v);
15          }
16
17          // Delete the keys created
18          jedis.del("foo", "a", "b", "c");
19
20          // Close the connection
21          jedis.close();
22  }
```

view raw (https://gist.github.com/ualmtorres/d0006849279588e03be1/raw/5b94772a76834e6641746b5e4c0f5a325d07d646/jedis-set-get.java) jedis-set-get.java (https://gist.github.com/ualmtorres/d0006849279588e03be1#file-jedis-set-get-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
bar
10
20
30
```

INCR, INCRBY, DECR, DECRBY

`incr(key)` y `decr(key)` incrementan o disminuyen en 1 el valor de la clave especificada.

`incrby(key, quantity)` y `decrby(key, quantity)` incrementan o disminuyen el valor de la clave especificada en el argumento proporcionado.

```
1   public void numbers() {
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
5           // Setup the key
6           jedis.set("counter", "100");
7
8           // INCR, INCRBY, DECR and DECRBY
9           jedis.incr("counter");
10          jedis.incrBy("counter", 9);
11          jedis.decrBy("counter", 4);
12          jedis.decr("counter");
13
14          // Prints the value of the key
15          System.out.println("counter: " + jedis.get("counter"));
16
17          // Delete the key created
18          jedis.del("counter");
19
20          // Close the connection
21          jedis.close();
22  }
```

view raw (https://gist.github.com/ualmtorres/d17e0927af5184db6c8b/raw/888af5c6b6efa071e1a78a27049a94ca112e1704/jedis-incr-decr.java)
jedis-incr-decr.java (https://gist.github.com/ualmtorres/d17e0927af5184db6c8b#file-jedis-incr-decr-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
counter: 105
```

APPEND, SUBSTR Y STRLEN

`append(key, string)` añade la cadena propocionada al final del valor de la clave especificada.

`substr(key, left, right)` devuelve una subcadena de la clave especificada comprendida entre dos posiciones.

`strlen(key)` devuelve el número de caracteres del valor de la clave especificada.

```
1   public void moreStrings() {
```

```
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
5           // Setup the key
6           jedis.set("greeting", "Hello ");
7
8           // EXISTS and APPEND
9           if (jedis.exists("greeting")) {
10                  jedis.append("greeting", "World!");
11          }
12          System.out.println("Appended greeting: " + jedis.get("greeting"));
13
14          // SUBSTR and STRLEN
15          System.out.println("Substring: " + jedis.substr("greeting", 6, -1));
16          System.out.println("greeting: " + jedis.get("greeting"));
17          System.out.println("Length: " + jedis.strlen("greeting"));
18
19          // Delete the key created
20          jedis.del("greeting");
21
22          // Close the connection
23          jedis.close();
24  }
```

view raw (https://gist.github.com/ualmtorres/0ed9b91edb074bc3ded9/raw/8ff4f33cb40809f2cef7ce9073fe665051a61618/jedis-more-strings.java)
jedis-more-strings.java (https://gist.github.com/ualmtorres/0ed9b91edb074bc3ded9#file-jedis-more-strings-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
Appended greeting: Hello World!
Substring: World!
greeting: Hello World!
Length: 12
```

LISTAS

Las listas son colecciones de valores que admiten repetidos.

`lpush(key, value)` inserta al principio (izquierda) de la clave especificada el valor proporcionado.
`rpush(key, value)` inserta al final (derecha).

`rpop(key)` extrae un valor del final (derecha) de la clave especificada.

`linsert(key, beforeOrAfter, pivot, value)` inserta en la clave especificada el valor proporcionado. El valor se inserta antes `BinaryClient.LIST_POSITION.BEFORE` o después `BinaryClient.LIST_POSITION.AFTER` del valor especificado (*pivot*).

`lset(key, value, position)` establece en la clave especificada el valor proporcionado en la posición especificada.

`lrange(key, left, right)` devuelve los elementos de la lista de la clave especificada entre que estén entre las posiciones indicadas (-1 representa el final).

```
1  public void lists() {
2          // Create a connection
3          Jedis jedis = new Jedis("localhost");
4
5          // Delete the key to avoid unexpected results
6          jedis.del("sessions:ggvd");
7
8          // LPUSH, RPUSH, RPOP, LINSERT and LSET
9          jedis.lpush("sessions:ggvd", "10/3");
10         jedis.rpush("sessions:ggvd", "24/3");
11         jedis.rpush("sessions:ggvd", "25/3");
12         jedis.rpop("sessions:ggvd");
13         jedis.linsert("sessions:ggvd", BinaryClient.LIST_POSITION.BEFORE, "24/3", "17/3");
14         jedis.rpush("sessions:ggvd", "31/3");
15         jedis.lset("sessions:ggvd", -1, "7/4");
16
17         // Obtain all the values of the lists
18         List<String> values = jedis.lrange("sessions:ggvd", 0, -1);
19
20         // Print the list
21         for (String v: values) {
22                 System.out.println(v);
23         }
24
25         // Delete the key created
26         jedis.del("sessions:ggvd");
27
28         // Close the connection
29         jedis.close();
30 }
```

view raw (https://gist.github.com/ualmtorres/1c421ffd5a42227fc734/raw/714aed4252d1be4457f0abffd57aa463d378a0cf/jedis-lists.java)
jedis-lists.java (https://gist.github.com/ualmtorres/1c421ffd5a42227fc734#file-jedis-lists-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
10/3
17/3
24/3
7/4
```

CONJUNTOS

Los conjuntos son colecciones de valores que no admiten repetidos.

`sadd(key, value1 [,value n])` añade a la clave especificada los elementos proporcionados.

`srem(key, value1 [, value n])` elimina de la clave especificada los elementos proporcionados.

`smembers(key)` devuelve todos los miembros del conjunto asociado a la clave especificada.

`scard(key)` devuelve el número de elementos del conjunto de la clave especificada.

```
1   public void sets() {
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
5           // Delete the key to avoid unexpected results
6           jedis.del("students:ggvd");
7
8           // SADD, SREM, SCARD and SMEMBERS
9           jedis.sadd("students:ggvd", "student1", "student2", "student3");
10          jedis.srem("students:ggvd", "student3");
11          System.out.println(jedis.scard("students:ggvd") + " elements");
12          Set<String> students = jedis.smembers("students:ggvd");
13
14          // Print the list
15          for (String student: students) {
16                  System.out.println(student);
17          }
18
19                  // Delete the key created
20                  jedis.del("students:ggvd");
21
22                  // Close the connection
23                  jedis.close();
24  }
```

view raw (https://gist.github.com/ualmtorres/0127ee3ceb0564c54085/raw/d7b6009bcd694f49b0528fcc7998a4c3bdbc273f/jedis-set.java)
jedis-set.java (https://gist.github.com/ualmtorres/0127ee3ceb0564c54085#file-jedis-set-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
2 elements
student2
student1
```

OPERACIONES DE CONJUNTOS

`sunion(set1, set2)`, `sinter(set1, set2)` y `sdiff(set1, set2)` obtienen, respectivamente, la unión, intersección y diferencia de conjuntos.

```
1   public void setOperations() {
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
5           // Delete the key to avoid unexpected results
6           jedis.del("students:bd");
7           jedis.del("students:ggvd");
8
9           // Setup the sets
10          jedis.sadd("students:ggvd", "student1", "student2", "student3");
11          jedis.sadd("students:bd", "student3", "student4", "student5");
12
13          // SUNION, SINTER and SDIFF
14          Set<String> totalStudents = jedis.sunion("students:bd", "students:ggvd");
15          Set<String> commonStudents = jedis.sinter("students:bd", "students:ggvd");
16          Set<String> studentsOnlyInGGVD = jedis.sdiff("students:ggvd", "students:bd");
17
18          // Print the union
19          System.out.println("*** Total students:");
20          for (String s: totalStudents) {
21                  System.out.println(s);
22          }
23
24          // Print the intersection
25          System.out.println("*** Common students:");
26          for (String s: commonStudents) {
27                  System.out.println(s);
28          }
29          // Print the difference
30          System.out.println("*** Students only in GGVD:");
31          for (String s: studentsOnlyInGGVD) {
32                  System.out.println(s);
33          }
```

```
34
35          // Delete the keys created
36          jedis.del("students:bd");
37          jedis.del("students:ggvd");
38
39          // Close the connection
40          jedis.close();
41  }
```

view raw (https://gist.github.com/ualmtorres/c3a445d4ced28f6aa29d/raw/aec48cf8fc0d7f8a34e5bc03b97505a39cecf796/jedis-set-operations.java)
jedis-set-operations.java (https://gist.github.com/ualmtorres/c3a445d4ced28f6aa29d#file-jedis-set-operations-java) hosted with ♡ by GitHub (https://github.com)

La ejecución de este método imprime

```
*** Total students:
student2
student1
student5
student4
student3
*** Common students:
student3
*** Students only in GGVD:
student2
student1
```

CONJUNTOS ORDENADOS

Los conjuntos ordenados son conjuntos cuyos elementos están acompañados de una puntuación que permite establecer un orden en el conjunto.

`zadd(key, score, value)` añade a la clave especificada la puntuación y el elemento proporcionado.

`zincrby(key, increment, value)` añade el incremento propocionado a la puntuación del elemento y clave especificados.

`zcount(key, lower, higher)` devuelve el número de elementos del conjunto que tienen su puntuación entre los límites propocionados

`zRangeByScoreWithScores(key, lower, higher)` devuelve los elementos y puntuación del conjunto y clave especificados cuyas puntuaciones están en el rango proporcionado.

```
1   public void sortedSets() {
2           // Create a connection
3           Jedis jedis = new Jedis("localhost");
4
```

```
        // Delete the key to avoid unexpected results
        jedis.del("scores:ggvd");

        // ZADD
        jedis.zadd("scores:ggvd", 9, "student1");
        jedis.zadd("scores:ggvd", 3, "student2");
        jedis.zadd("scores:ggvd", 8, "student3");

        // ZINCRBY
        jedis.zincrby("scores:ggvd", 1, "student2");

        // ZCOUNT
        long numberOfPassStudents = jedis.zcount("scores:ggvd", 5, 10);

        // ZRANGEBYSCOREWITHSCORES
        Set<Tuple> passedStudents = jedis.zrangeByScoreWithScores("scores:ggvd", 5, 10);

        // Print the results
        System.out.println("*** Number of passed students: " + numberOfPassStudents);

        for (Tuple s: passedStudents) {
                System.out.println(s.getElement() + " " + s.getScore());
        }

        // Delete the keys created
        jedis.del("scores:ggvd");

        // Close the connection
        jedis.close();
}
```

view raw (https://gist.github.com/ualmtorres/0a823dab8516ef8dc2c0/raw/fc0428e6e15712325b09fb05b31d043f8a2f5aa5/jedis-sorted-sets.java)
jedis-sorted-sets.java (https://gist.github.com/ualmtorres/0a823dab8516ef8dc2c0#file-jedis-sorted-sets-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
*** Number of passed students: 2
student3 8.0
student1 9.0
```

HASHES

Los hashes son listas de campo-valor asociados a una clave.

`hset(key, field, value)` asigna a la clave especificada el campo y valor propocionados.

`hget(key, field)` obtiene el valor asociado a la clave y campo especificados.

`hkeys(key)` obtiene un conjunto con la lista de campos de un clave.

```
public void hashes() {
        // Create a connection
        Jedis jedis = new Jedis("localhost");

        // Delete the key to avoid unexpected results
        jedis.del("user:mtorres");

        // HSET
        jedis.hset("user:mtorres", "email", "mtorres@ual.es");
        jedis.hset("user:mtorres", "name", "Manuel");
        jedis.hset("user:mtorres", "surname", "Torres Gil");
        jedis.hset("user:mtorres", "twitter", "@ualmtorres");

        // HKEYS
        Set<String> keys = jedis.hkeys("user:mtorres");

        // Print the results
        for (String c: keys) {
                System.out.println(c + ": " + jedis.hget("user:mtorres", c));
        }

        // Delete the key created
        jedis.del("user:mtorres");

        // Close the connection
        jedis.close();
}
```

view raw (https://gist.github.com/ualmtorres/210be8cb6ec002bb01c1/raw/13a4b37c10f68cd2ad46aae0d467c280cc2ffb2e/jedis-hashes.java)
jedis-hashes.java (https://gist.github.com/ualmtorres/210be8cb6ec002bb01c1#file-jedis-hashes-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
twitter: @ualmtorres
surname: Torres Gil
email: mtorres@ual.es
name: Manuel
```

TRANSACCIONES

Las transacciones se inician con `multi()` .

Para finalizar la transacción usaremos:

- `exec()` : ejecuta las instrucciones de una transacción.
- `discard()` : cancela las instrucciones de una transacción.

```
1  public void transactions() {
2          Jedis jedis = new Jedis("localhost");
3
4          // Transaction commiting results
5          Transaction t = jedis.multi();
6          t.set("a", "1");
7          t.set("b", "2");
8          t.exec();
9
10         // Transaction discarding results
11         t = jedis.multi();
12         t.set("a", "3");
13         t.set("b", "4");
14         t.discard();
15
16         System.out.println("*** Keys after discarding: ");
17         System.out.println(jedis.get("a"));
18         System.out.println(jedis.get("b"));
19
20         // Delete the keys created
21         jedis.del("a", "b");
22
23         // Close the connection
24         jedis.close();
25 }
```

view raw (https://gist.github.com/ualmtorres/f247272dbe2a68d2a283/raw/747ccd607556bc8e11d9cc72a2bc2070c5deb1df/jedis-transactions.java)
jedis-transactions.java (https://gist.github.com/ualmtorres/f247272dbe2a68d2a283#file-jedis-transactions-java) hosted with ♡ by **GitHub (https://github.com)**

La ejecución de este método imprime

```
*** Keys after discarding:
1
2
```

Puedes descargar el código de este tutorial (https://github.com/ualmtorres/RedisJava) que incluye una batería de juegos de prueba del GitHub.