Redis in Java (Using Jedis)

Redis is an open-source, in-memory data store that is particularly useful for applications that require high-speed data retrieval and low-latency access. In Java programs, you can interact with Redis using the **Jedis** library, a popular Java client for Redis.

- To include Jedis as a dependency in your application, edit the dependency file, as follows.
  - If you use **Maven**:

    ```
    <dependency>
      <groupId>redis.clients</groupId>
      <artifactId>jedis</artifactId>
      <version>6.0.0</version>
    </dependency>
    ```

  - If you use **Gradle**:

    ```
    dependencies {
          implementation 'redis.clients:jedis:6.0.0'
          //….
    }
    ```

- Connecting to Redis Server from Java:
  - Here's how you can create a connection:
  - Code:

```java
import redis.clients.jedis.Jedis;

public class RedisConnection {
   public static void main(String[] args) {
      //Connecting to Redis server on localhost
      Jedis jedis = new Jedis("localhost", 6379);
      System.out.println("Connection to server
successfully");
      //check whether server is running or not
      System.out.println("Server is running: " +
jedis.ping());
   }
}
```

  - Output:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Connection to server successfully
Server is running: PONG

Process finished with exit code 0
```

💡 Redis Caching Example (Java Class)

- This example demonstrates:
  - Checking Redis cache
  - Falling back to a simulated DB
  - Storing structured JSON-like values
  - Applying TTL (expiration)
  - Auto key expiration
- 🔧 Java Code:

```
import redis.clients.jedis.Jedis;
import com.google.gson.Gson;

import java.util.HashMap;
import java.util.Map;

public class RedisCacheDemo {

    private static final String REDIS_HOST = "localhost";
    private static final int REDIS_PORT = 6379;
    private static final String USER_KEY = "user:101";
    private static final int TTL_SECONDS = 30;

    public static void main(String[] args) throws
InterruptedException {
        Jedis jedis = new Jedis(REDIS_HOST, REDIS_PORT);
        Gson gson = new Gson();

        // Step 1: Try to get value from Redis
        String value = jedis.get(USER_KEY);

        if (value == null) {
            System.out.println("🔍 Cache MISS —
simulating DB call...");

            // Simulated DB call
            String dbValue = getFromDatabase();

            // Convert to JSON
            Map<String, Object> jsonObject = new
```

```
HashMap<>();
            jsonObject.put("name", dbValue);
            jsonObject.put("role", "user");

            String json = gson.toJson(jsonObject);

            // Store in Redis with TTL
            jedis.set(USER_KEY, json);
            jedis.expire(USER_KEY, TTL_SECONDS);

            System.out.println("☑ Stored in Redis as
JSON: " + json);
        } else {
            System.out.println("☑ Cache HIT — value from
Redis: " + value);
        }

        // Step 2: GET again — should return cached JSON
        System.out.println("\n🔁 Step 2: GET from
cache...");
        String cached = jedis.get(USER_KEY);
        System.out.println("Value in Redis: " + cached);

        // Step 3: Update value (simulate PUT)
        System.out.println("\n🛠 Step 3: Update user
data...");
        Map<String, Object> updatedJson = new
HashMap<>();
        updatedJson.put("name", "Alice");
        updatedJson.put("role", "admin");
        updatedJson.put("active", true);

        String updated = gson.toJson(updatedJson);
        jedis.set(USER_KEY, updated);
        jedis.expire(USER_KEY, TTL_SECONDS);

        System.out.println("☑ Updated user JSON: " +
updated);

        // Step 4: Fetch updated value
        System.out.println("\n📥 Step 4: GET after
update...");
        String afterUpdate = jedis.get(USER_KEY);
        System.out.println("Current value in Redis: " +
afterUpdate);

        // Step 5: Wait for TTL to expire
        System.out.println("\n⏳ Step 5: Sleeping " +
TTL_SECONDS + " seconds...");
```

```
        Thread.sleep(TTL_SECONDS * 1000);

        // Step 6: Check if key is expired
        System.out.println("\n🔍 Step 6: Final GET after
expiry...");
        String afterExpiry = jedis.get(USER_KEY);
        if (afterExpiry == null) {
            System.out.println("☑ Key has expired and is
removed from Redis.");
        } else {
            System.out.println("❌ Key still exists: " +
afterExpiry);
        }

        jedis.close();
    }

    private static String getFromDatabase() {
        // Simulated DB call
        return "Alice_from_DB";
    }
}
```

- Output:

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
🔍 Cache MISS – simulating DB call...
✅ Stored in Redis as JSON: {"role":"user","name":"Alice_from_DB"}

🔁 Step 2: GET from cache...
Value in Redis: {"role":"user","name":"Alice_from_DB"}

🛠️ Step 3: Update user data...
✅ Updated user JSON: {"role":"admin","name":"Alice","active":true}

📦 Step 4: GET after update...
Current value in Redis: {"role":"admin","name":"Alice","active":true}

⏳ Step 5: Sleeping 30 seconds...

🔍 Step 6: Final GET after expiry...
✅ Key has expired and is removed from Redis.

Process finished with exit code 0
```