redis
Clients

Java SpringBoot New
- Introduction
- Strings
- Lists
- Sets
- ZSets
- Hashes
- Streams
- Common Keys
- Pipelining
- Pub/Sub
- Master/Replica
- Sentinel
- Cluster
- ---
- Auto Config
- Manual Config
- Load Balancing (readFrom)
- RedisTemplate
- Connection Pool & Thread
- Async Spring & Lettuce
- DB select
- Spring Multi Data Source
- Lettuce Multi Data Source
- Spring Project Create
- Spring Project Eclipse
- Spring Project IntelliJ
- ---
- Spring Session Standalone
- Spring Session MasterRepli
- Spring Session Sentinel
- Spring Session Cluster

Java Lettuce(Spring)New

Java Lettuce(Plain)

Java Jedis

Java Redisson

C Hiredis

C# StackExchange

PHP PhpRedis

PHP Predis

Redis Admin & Monitoring Tool

# Spring Data Redis Introduction



## Spring Data Redis 개요

Spring Data Redis는 Java Spring Framework으로 Redis 서버에 접속해서 애플리케이션을 개발하는 방법을 제공합니다. Spring도 하위에서는 레터스(Lettuce) 라이브러리를 사용하지만, 레디스 서버에 접속 방법, 메서드 명 등 개발하는 방법이 레터스를 직접 사용했을 경우와는 다릅니다.

- 레터스를 사용해서 개발하는 방법은 여기를 보세요.
- Spring Data Redis 문서(v 3.1.3)는 여기에 있습니다.
- Spring Data Redis 버전별 문서 목록은 여기를 보세요.
- Spring Boot 2.0부터 Lettuce가 Redis 기본 클라이언트로 사용되고 있습니다.
- Spring Boot 1.x에서는 Jedis가 기본 클라이언트로 사용되었습니다.

### Spring Data Redis

- Spring은 **RedisConnectionFactory** 클래스로 레디스 서버에 연결합니다. 이 프로젝트에서는 자동연결 방법을 사용하므로 직접 RedisConnectionFactory 클래스 사용하지는 않습니다. Spring Boot 내부적으로 application.properties 파일을 읽어서 RedisConnectionFactory 클래스를 설정합니다.
- 레디스 명령을 사용하려면 RedisTemplate 또는 StringRedisTemplate를 사용합니다. RedisTemplate은 object를 인자로 받으므로 키와 값에 특수문자가 포함되어 레디스 서버에 저장됩니다. 따라서 redis-cli로 접속해서 조회했을 때 특수문자가 포함되어 조회되고, 키도 특수문자가 포함되어 있으므로 조회하기 매우 어렵습니다.

  **StringRedisTemplate**은 문자열만 저장됩니다. StringRedisTemplate를 사용하시기 바랍니다.
- Spring은 애플리케이션 시작 시 레디스 서버에 연결(접속)하지 않습니다. 실제 명령 실행 시 연결합니다. 이런 것을 일반적으로 지연 연결(Lazy Connection)이라고 합니다.

### Redis Data Type 별 인터페이스

레디스 명령을 사용하려면 각 데이터 타입별로 인터페이스를 사용해야 합니다.
각 Spring 메서드와 레디스 명령 연결 관계는 여기를 보세요.

- **Strings : ValueOperations**<String,String> valueOperations = stringRedisTemplate.opsForValue()
- **Lists : ListOperations**<String,String> listOperations = stringRedisTemplate.opsForList()
- **Sets : SetOperations**<String,String> setOperations stringRedisTemplate.opsForSet()
- **ZSets : ZSetOperations**<String,String> zsetOperations = stringRedisTemplate.opsForZSet()
- **Hashes : HashOperations**<String,String,String> hashOperations = stringRedisTemplate.opsForHash();
- **Streams : StreamOperations**<String,String,String> streamOperations = stringRedisTemplate.opsForStream()
- **Common Keys : stringRedisTemplate**를 직접 사용합니다.

### ShareNativeConnection

LettuceConnectionFactory에는 shareNativeConnection을 설정할 수 있습니다. 기본값은 true입니다. 설정 방법은 lettuceConnectionFactory.setShareNativeConnection(true/false); 입니다.

**Redis Standalone 구성: RedisStandaloneConfiguration**

RedisStandaloneConfiguration으로 LettuceConnectionFactory을 사용해서 레디스 서버에 연결할 경우

**TRUE 설정일 때**

- 처음 레디스 명령(예 SET) 명령을 실행하면 연결하고 명령을 실행합니다. 연결을 닫지 않습니다.

  01:08:39.949 - Accepted 192.168.56.1:10883
  01:08:39.978 - 192.168.56.1:10883: HELLO 3 AUTH default redisgate SETNAME redisgate
  01:08:40.058 - 192.168.56.1:10883: SET key01 value-key01
  01:09:30.348 - 192.168.56.1:10883: SET key01 value-key01
  01:09:38.120 - 192.168.56.1:10883: GET key01

**FALSE 설정일 때**

- 명령을 실행할 때마다 레디스 서버에 연결하고 닫습니다.

  01:17:34.104 - Accepted 192.168.56.1:10962
  01:17:34.131 - 192.168.56.1:10962: HELLO 3 AUTH default redisgate SETNAME redisgate
  01:17:34.173 - 192.168.56.1:10962: SET key01 value-key01
  01:17:34.179 - **Client closed connection** addr=192.168.56.1:10962

**Master/Replica 구성: RedisStaticMasterReplicaConfiguration**

RedisStaticMasterReplicaConfiguration으로 LettuceConnectionFactory을 사용해서 레디스 서버에 연결할 경우

**TRUE 설정일 때**

- 처음 레디스 명령(예 SET) 명령을 실행하면 아래와 같이 총 3번 연결과 7개의 명령을 실행한다.
  세 번째 연결은 닫지 않는다.
  **마스터 서버**

  16:44:21.710 - Accepted 192.168.56.1:3487
  16:44:21.733 - 192.168.56.1:3487: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:44:21.778 - 192.168.56.1:3487: ROLE
  16:44:21.795 - **Client closed connection** addr=192.168.56.1:3487
  16:44:21.812 - Accepted 192.168.56.1:3490
  16:44:21.814 - 192.168.56.1:3490: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:44:21.816 - 192.168.56.1:3490: CLIENT SETNAME lettuce#MasterReplicaTopologyRefresh
  16:44:21.824 - 192.168.56.1:3490: PING NODES
  16:44:21.833 - **Client closed connection** addr=192.168.56.1:3490
  16:44:21.881 - Accepted 192.168.56.1:3492
  16:44:21.882 - 192.168.56.1:3492: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:44:21.885 - 192.168.56.1:3492: SET key01 value-key01

  **복제 서버**
  복제 서버에는 SET 명려을 실행하지 않지만, 2번 연결과 5개 명령을 실행한다. 복제 서버가 여러 대 있으면 동일하게 실행된다.

  16:44:21.710 - Accepted 192.168.56.1:3486
  16:44:21.733 - 192.168.56.1:3486: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:44:21.777 - 192.168.56.1:3486: ROLE
  16:44:21.795 - **Client closed connection** addr=192.168.56.1:3486
  16:44:21.815 - Accepted 192.168.56.1:3491
  16:44:21.815 - 192.168.56.1:3491: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:44:21.817 - 192.168.56.1:3491: CLIENT SETNAME lettuce#MasterReplicaTopologyRefresh
  16:44:21.825 - 192.168.56.1:3491: PING NODES
  16:44:21.832 - **Client closed connection** addr=192.168.56.1:3491

- 이후 SET, GET, PING 명령을 실행하면 세번 째 연결을 사용해서 해당 명령만 실행한다.

  **마스터 서버**

  16:45:27.103 - 192.168.56.1:3492: **SET** key01 value-key01
  16:46:12.501 - 192.168.56.1:3492: **GET** key01
  16:46:43.871 - 192.168.56.1:3492: **PING**

**FALSE 설정일 때**

- 처음 레디스 명령(예 SET) 명령을 실행하면 아래와 같이 총 3번 연결과 7개 명령을 실행한다.

  **세 번째 연결도 닫는다.**

  **마스터 서버**

  16:39:20.397 - Accepted 192.168.56.1:3406
  16:39:20.399 - 192.168.56.1:3406: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:39:20.401 - 192.168.56.1:3406: ROLE
  16:39:20.403 - **Client closed connection** addr=192.168.56.1:3406
  16:39:20.407 - Accepted 192.168.56.1:3411
  16:39:20.407 - 192.168.56.1:3411: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:39:20.409 - 192.168.56.1:3411: CLIENT SETNAME lettuce#MasterReplicaTopologyRefresh
  16:39:20.409 - 192.168.56.1:3411: PING NODES
  16:39:20.413 - **Client closed connection** addr=192.168.56.1:3411
  16:39:20.415 - Accepted 192.168.56.1:3412
  16:39:20.415 - 192.168.56.1:3412: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:39:20.417 - 192.168.56.1:3412: SET key01 value-key01
  16:39:20.418 - **Client closed connection** addr=192.168.56.1:3412

  **복제 서버**

  복제 서버에는 SET 명령을 실행하지 않지만, 2번 연결과 5개 명령을 실행한다. 복제 서버가 여러 대 있으면 동일하게 실행된다.

  16:39:20.398 - Accepted 192.168.56.1:3407
  16:39:20.398 - 192.168.56.1:3407: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:39:20.399 - 192.168.56.1:3407: ROLE
  16:39:20.400 - **Client closed connection** addr=192.168.56.1:3407
  16:39:20.405 - Accepted 192.168.56.1:3409
  16:39:20.406 - 192.168.56.1:3409: HELLO 3 AUTH default redisgate SETNAME redisgate
  16:39:20.408 - 192.168.56.1:3409: CLIENT SETNAME lettuce#MasterReplicaTopologyRefresh
  16:39:20.409 - 192.168.56.1:3409: PING NODES
  16:39:20.412 - **Client closed connection** addr=192.168.56.1:3409

- 이후 SET, GET, PING 명령을 실행하면 각각 위 똑같이 마스터에서 3번 연결과 7개 명령을 실행하고 복제 서버에도 2번 연결과 5개 명령이 실행된다.

**FALSE로 사용하는 경우**

Connection Pool을 사용하고 Thread 마다 각각 레디스 연결을 맺어서 명령을 실행하고 싶은 경우 사용합니다. 하지만 이 경우에도 성능상 이점은 없습니다.
자세한 내용은 여기를 보세요

Spring은 명령을 실행할 때 getSharedConnection() 메서드를 실행해서 연결을 얻습니다.
shareNativeConnection이 true인 경우 getConnection()으로 연결을 리턴하고 false이면 NULL을 리턴합니다.
아래는 단순화한 코드입니다.

```
getSharedConnection() {
   shareNativeConnection ? getConnection() : null;
}
```

**TRUE로 설정해도 별도 연결을 맺는 경우**

대기 명령(blocking) (예, BLPOP), 트랜적션 처리(MULTI/EXEC), 파이프라인(Pipeline) 등의 경우 새 연결을 맺어서 처리하고 종료(close)합니다.

## Spring Project

### Spring Project 생성

이제 Java Spring Project를 생성해서 애플리케이션을 작성/실행해 봅니다.
Spring initializr를 사용해서 Java Spring Project를 생성하는 URL: **start.spring.io**


- Project: **Gradle-Groovy**
- Language: **Java**
- Spring Boot: **3.1.8**
- Project Metadata
  - Group: **com.redisgate**
  - Artifact: **RedisSpring** -> Project 명칭, 이 이름으로 압축파일이 생긴다.
  - Name: **Main** -> Main Class Name
  - Description: **Redis Spring Application** -> 설명
  - Package name: **com.redisgate.redis** -> Package Name
  - Packaging: **Jar**
  - Java: **17** -> Java 버전 선택
- Dependencies에서 **[ADD ... CTRL + B]** 버튼을 클릭해서 아래 3개를 추가합니다.
  - **Spring Web**
  - **Lombok**
  - **Spring Data Redis(Access+Driver)**

- 마지막으로 **[GENERATE CTRL + ⏎ ]** 버튼 클릭해서 압축 파일(zip)을 다운받는다.
- 스프링부트3.x 는 자바17 이상, 스프링부트 2.x는 자바11을 사용합니다.

### 개발 툴(IDE) 설정, 사용법

- **Spring Project** 생성, IntelliJ(인텔리J), Eclipse(이클립스) Java 버전 설정 방법은 여기를 보세요.
- **Eclipse** Spring Project Import(Open) 방법, Lombok 에러 해결 방법은 여기를 보세요.
- **IntelliJ** Spring Project Open 방법, IntelliJ Lombok 인식 오류 해결 방법은 여기를 보세요.

**이 프로젝트에 사용된 Lettuce, Spring, Redis 버전**

- Lettuce-6.2.7
- Spring-Boot: 3.1.8, Spring: 6.0.16
- Redis-7.2.3

### Class/File 구성

- Main Class: 메인 클래스
- application.properties: 레디스 서버에 연결 값들
- Redis_String class: 레디스 스트링 명령 구현 클래스
- Redis_List class: 레디스 리스트 명령 구현 클래스
- Redis_Set class: 레디스 셋 명령 구현 클래스
- Redis_ZSet class: 레디스 정렬셋 명령 구현 클래스
- Redis_Hash class: 레디스 해시 명령 구현 클래스
- Redis_Stream class: 레디스 스트림 명령 구현 클래스
- Redis_Common class: 레디스 키 공통 명령 구현 클래스
- Redis_Pipeline class: 레디스 Pipeline 설명과 구현 클래스
- Redis_PubSub class: 레디스 Pub/Sub 구현 클래스

### Main class

```
package com.redisgate.redis;

import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@Slf4j
public class MainApplication {
	public static void main(String[] args) {
		SpringApplication.run(MainApplication.class, args);
		log.info("----- Redis Standalone Spring Application 시작 -----");
	}
}
```

### application.properties

레디스 서버에 연결은 아래 형식에 맞추어 입력하면 자동으로 연결된다. 그러므로 레디스 서버에 연결하기 위한 별도의 클래스가 필요하지 않다.
제일 간단히 연결하기 위한 항목은 host와 port이다. password, database 등 나머지 항목은 선택이므로 필요에 따라 입력하면 된다.
레디스 서버의 ip와 port가 localhost(127.0.0.1)과 6379일 경우 host와 port도 입력하지 않아도 연결된다.
host와 port가 제공되지 않았을 경우 Spring(Lettuce)는 기본적으로 127.0.0.1과 6379로 레디스 서버에 연결

한다.

```
# Redis Spring Properties
#spring.data.redis.host: 127.0.0.1
#spring.data.redis.port: 6379
spring.data.redis.host: 192.168.56.102
spring.data.redis.port: 6000
spring.data.redis.password: redisgate
spring.data.redis.database: 0
spring.data.redis.timeout: 5s
spring.data.redis.connectTimeout: 5s
spring.data.redis.clientName: redisgate
#server.port: 9090
#logging.level.root: debug
```

Lettuce Introduction Strings >>

redisgate@gmail.com 02.503.2235
서울시 강남구 강남대로 342 역삼빌딩 5층 (역삼동) 우 06242