Redis

基于内存进行存储，支持 key-value 的存储形式，底层是用 C 语言编写的。

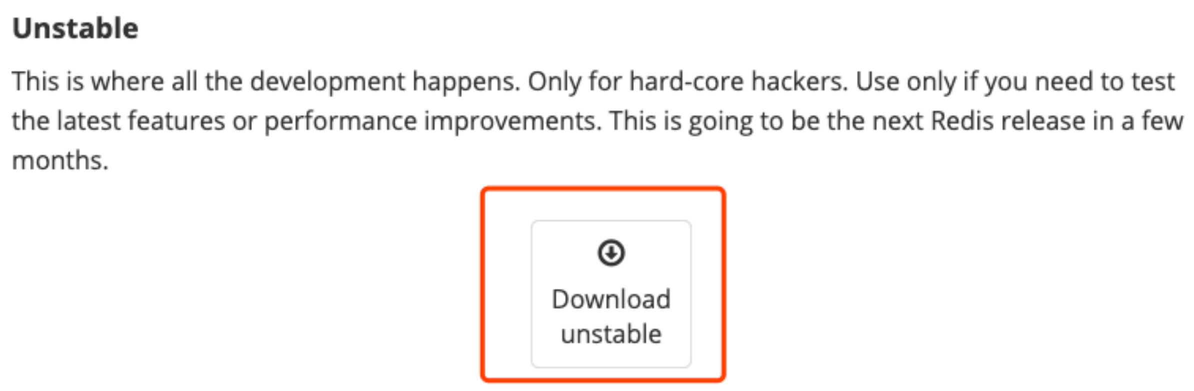
基于 key-value 形式的数据字典，结构非常简单，没有数据表的概念，直接用键值对的形式完成数据的管理，Redis 支持 5 种数据类型：

- 字符串
- 列表
- 集合
- 有序集合
- 哈希

安装 Redis

1、下载 Redis

https://redis.io/download



Unstable

This is where all the development happens. Only for hard-core hackers. Use only if you need to test the latest features or performance improvements. This is going to be the next Redis release in a few months.

2、解压，并在本地硬盘任意位置创建文件夹，在其中创建 3 个子文件夹

- bin：放置启动 Redis 的可执行文件
- db：放置数据文件
- etc：放置配置文件，设置 Redis 服务的端口、日志文件位置、数据文件位置...

启动 Redis 服务

1、进入 redis 目录，启动 redis-server。

```
sudo ./bin/redis-server ./etc/redis.conf
```

2、进入 redis 目录，启动 redis-cli，启动 Redis 的客户端管理窗口，在此窗口中即可操作 Redis 数据库。

```
./bin/redis-cli
```

3、对数据进行操作。

```
set key value
get key
```

4、关闭 Redis 服务。

```
shutdown
```

5、退出客户端，control+c。

Spring Boot 整合 Redis

Spring Data Redis 操作 Redis。

1、创建 Maven 工程。

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.5.RELEASE</version>
</parent>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
  </dependency>

  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
  </dependency>

  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
  </dependency>
</dependencies>
```

2、创建实体类，实现序列化接口，否则无法存入 Redis 数据库。

```
package com.southwind.entity;

import lombok.Data;

import java.io.Serializable;
import java.util.Date;

@Data
public class Student implements Serializable {
    private Integer id;
    private String name;
    private Double score;
    private Date birthday;
}
```

3、创建控制器。

```
package com.southwind.controller;

import com.southwind.entity.Student;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.web.bind.annotation.*;

@RestController
public class StudentHandler {

    @Autowired
    private RedisTemplate redisTemplate;

    @PostMapping("/set")
    public void set(@RequestBody Student student){
        redisTemplate.opsForValue().set("student",student);
    }

    @GetMapping("/get/{key}")
    public Student get(@PathVariable("key") String key){
        return (Student) redisTemplate.opsForValue().get(key);
    }

    @DeleteMapping("/delete/{key}")
    public boolean delete(@PathVariable("key") String key){
        redisTemplate.delete(key);
        return redisTemplate.hasKey(key);
    }
}
```

4、创建配置文件 application.yml

```
spring:
  redis:
    database: 0
    host: localhost
    port: 6379
```

5、创建启动类

```
package com.southwind;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class,args);
    }
}
```

Redis 5 种数据类型

字符串

```
@GetMapping("/string")
public String stringTest(){
    redisTemplate.opsForValue().set("str","Hello World");
    String str = (String) redisTemplate.opsForValue().get("str");
    return str;
}
```

列表

```
@GetMapping("/list")
public List<String> listTest(){
    ListOperations<String,String> listOperations = redisTemplate.opsForList();
    listOperations.leftPush("list","Hello");
    listOperations.leftPush("list","World");
    listOperations.leftPush("list","Java");
    List<String> list = listOperations.range("list",0,2);
    return list;
}
```

集合

```
@GetMapping("/set")
public Set<String> setTest(){
  SetOperations<String,String> setOperations = redisTemplate.opsForSet();
  setOperations.add("set","Hello");
  setOperations.add("set","Hello");
  setOperations.add("set","World");
  setOperations.add("set","World");
  setOperations.add("set","Java");
  setOperations.add("set","Java");
  Set<String> set = setOperations.members("set");
  return set;
}
```

有序集合

```
@GetMapping("/zset")
public Set<String> zsetTest(){
    ZSetOperations<String,String> zSetOperations = redisTemplate.opsForZSet();
    zSetOperations.add("zset","Hello",1);
    zSetOperations.add("zset","World",2);
    zSetOperations.add("zset","Java",3);
    Set<String> set = zSetOperations.range("zset",0,2);
    return set;
}
```

哈希

HashMap key value

HashOperations key hashkey value

key 是每一组数据的 ID，hashkey 和 value 是一组完整的 HashMap 数据，通过 key 来区分不同的 HashMap。

```
HashMap hashMap1 = new HashMap();
hashMap1.put(key1,value1);
HashMap hashMap2 = new HashMap();
hashMap2.put(key2,value2);
HashMap hashMap3 = new HashMap();
hashMap3.put(key3,value3);
HashOperations<String,String,String> hashOperations =
redisTemplate.opsForHash();
hashOperations.put(hashMap1,key1,value1);
hashOperations.put(hashMap2,key2,value2);
hashOperations.put(hashMap3,key3,value3);
```