SpringData 第三章 Spring Data Redis

一、 Spring Data Redis 简介

<u>E. EKWERI</u>

Spring Data Redis



Spring Data Redis, part of the larger Spring Data family, provides easy configuration and access to Redis from Spring applications. It offers both low-level and high-level abstractions for interacting with the store, freeing the user from infrastructural concerns.

二、 Redis 安装

1 安装环境

Redis 版本: 3.0.0 环境: Linux

- 2 安装步骤
 - 2.1 安装 gcc 编译器

yum install gcc-c++

2.2解压安装包

tar -zxf redis-3.0.0.tar.gz

2.3进入解压目录进行编译

cd redis-3.0.0 make

2.4将 Redis 安装到指定目录

make PREFIX=/usr/local/redis install

尚学堂·**在於**程序员

2.5 启动 Redis

ENTER

2.5.1 前置启动

默认的是前置启动: ./redis-server

2.5.2 后置启动

先将 redis.conf 文件拷贝到 redis 的安装目录 cp redis.conf /usr/local/redis/bin 编辑 redis.conf 文件修改: daemonize yes 启动: ./redis-server redis.conf 查看 redis 进程: ps aux|grep redis 关闭后置启动的 Redis: ./redis-cli shutdown

三、 搭建整合环境

1 创建项目

- 13-spring-data-redis
 - ▲ 🕭 src
 - x applicationContext.xml
 - P redis.properties

 - A Referenced Libraries
 - aopalliance.jar
 - aspectirt.jar
 - aspectiweaver.jar
 - Dommons-logging-1.1.1.jar
 - D a commons-pool2-2.3.jar

 - ▶ 5 spring-aspects-4.2.0.RELEASE.jar
 - spring-beans-4.2.0.RELEASE.jar
 - spring-context-4.2.0.RELEASE.jar
 - spring-core-4.2.0.RELEASE.jar
 - spring-data-redis-1.6.0.RELEASE.jar
 - spring-expression-4.2.0.RELEASE.jar
 - ▶ 5 spring-test-4.2.0.RELEASE.jar
 - spring-tx-4.2.0.RELEASE.jar

lib

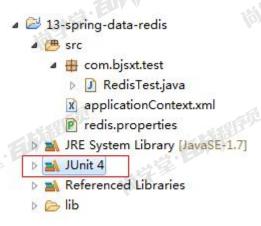
2 整合配置

配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
   <beans xmlns="http://www.springframework.org/schema/beans"</pre>
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx.xsd">
       <!-- 配置读取 properties 文件的工具类 -->
       <context:property-placeholder</pre>
location="classpath:redis.properties",
       <!-- Jedis 连接池 -->
       <bean id="poolConfig"</pre>
class="redis.clients.jedis.JedisPoolConfig">
           cproperty name="maxTotal" value="${redis.pool.maxTotal}"/>
           cproperty name="maxIdle" value="${redis.pool.maxIdle}"/>
           cproperty name="minIdle" value="${redis.pool.minIdle}"/>
       </bean>
       <!-- Jedis 的连接工厂 -->
       <bean id="jedisConnectionFactory"</pre>
class="org.springframework.data.redis.connection.jedis.JedisConnectionF
actory">
           cproperty name="hostName" value="${redis.conn.hostName}"/>
           cproperty name="port" value="${redis.conn.port}"/>
           cproperty name="poolConfig" ref="poolConfig"/>
       </bean>
       <!-- Redis 模板对象 -->
       <bean id="redisTemplate"</pre>
class="org.springframework.data.redis.core.RedisTemplate">
           property name="connectionFactory"
ref="jedisConnectionFactory"/>
           <!-- 序列化器: 能够把我们储存的 key 与 value 做序列化处理的对象
           <!-- 配置默认的序列化器 -->
           <!-- keySerializer、valueSerializer 配置 Redis 中的 String 类型 key
```

3 测试整合环境

3.1添加 junit 包



3.2关闭 linux 防火墙,或者在防火墙中开启 6379 端口

3.3 测试代码

```
/**

* <u>Redis</u>测试

* <u>@author</u> Administrator

*

*/
@RunWith(SpringJUnit4ClassRunner.class)
```

```
@ContextConfiguration("classpath:applicationContext.xml")
public class RedisTest {

    @Autowired
    private RedisTemplate<String, Object> redisTemplate;

    /**
        * 添加键值对
        */
        @Test
        public void test1(){
            this.redisTemplate.opsForValue().set("key", "test");
        }

        /**
        * 获取 redis 中的数据
        */
        @Test
        public void test2(){
            String str =
        (String)this.redisTemplate.opsForValue().get("key");
            System.out.println(str);
        }
    }
}
```

四、 Spring Data Redisd 存储实体对象

1 测试代码

```
/**
    * 添加 Users
    */
    @Test
public void test3(){
    Users users = new Users();
    users.setAge(30);
    users.setId(1);
    users.setName("张三");
    //更换序列化器
    this.redisTemplate.setValueSerializer(new

JdkSerializationRedisSerializer());
    this.redisTemplate.opsForValue().set("users", users);
}
```

```
/**
    * 获取 Users
    *

    */
    @Test
    public void test4(){
        //更换序列化器
        this.redisTemplate.setValueSerializer(new

JdkSerializationRedisSerializer());
        Users users =

(Users)this.redisTemplate.opsForValue().get("users");
        System.out.println(users);
}
```

五、 Spring Data Redis 以 JSON 的格式存储实体对象

1 导入 jackson 包

```
ijackson-annotations-2.8.0.jar
ijackson-core-2.8.10.jar
ijackson-databind-2.8.10.jar
```

2 编写测试代码

```
* 添加 Users JSON 格式

*/
@Test
public void test5(){
    Users users = new Users();
    users.setAge(23);
    users.setId(2);
    users.setName("李四");

    this.redisTemplate.setValueSerializer(new
Jackson2JsonRedisSerializer<>(Users.class));
    this.redisTemplate.opsForValue().set("usersjson", users);
}

/**

* 获取 Ueers JSON 格式
```

```
#/
@Test
public void test6(){
    this.redisTemplate.setValueSerializer(new
Jackson2JsonRedisSerializer<>(Users.class));
    Users users =
(Users)this.redisTemplate.opsForValue().get("usersjson");
    System.out.println(users);
}
```