

SpringDataRedis

创建项目

The image shows two screenshots of the IntelliJ IDEA 'New Project' wizard. The top screenshot shows the 'Spring Initializr' option selected in the left-hand menu. The 'Project SDK' is set to '1.8 (java version "1.8.0_231")'. The 'Choose Initializr Service URL' section has 'Default: https://start.spring.io' selected. The bottom screenshot shows the 'Project Metadata' configuration screen. The 'Group' is 'com.xxxx', 'Artifact' is 'springdataredis-demo', 'Type' is 'Maven Project (Generate a Maven based project archive.)', 'Language' is 'Java', 'Packaging' is 'Jar', and 'Java Version' is '8'. The 'Version' is '0.0.1-SNAPSHOT', 'Name' is 'springdataredis-demo', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.xxxx.springdataredisdemo'.

New Project

Project SDK: 1.8 (java version "1.8.0_231")

Choose Initializr Service URL.

☒ Default: <https://start.spring.io>

☐ Custom:

Make sure your network connection is active before continuing.

Previous Next Cancel Help

New Project

Project Metadata

Group: com.xxxx

Artifact: springdataredis-demo

Type: Maven Project (Generate a Maven based project archive.)

Language: Java

Packaging: Jar

Java Version: 8

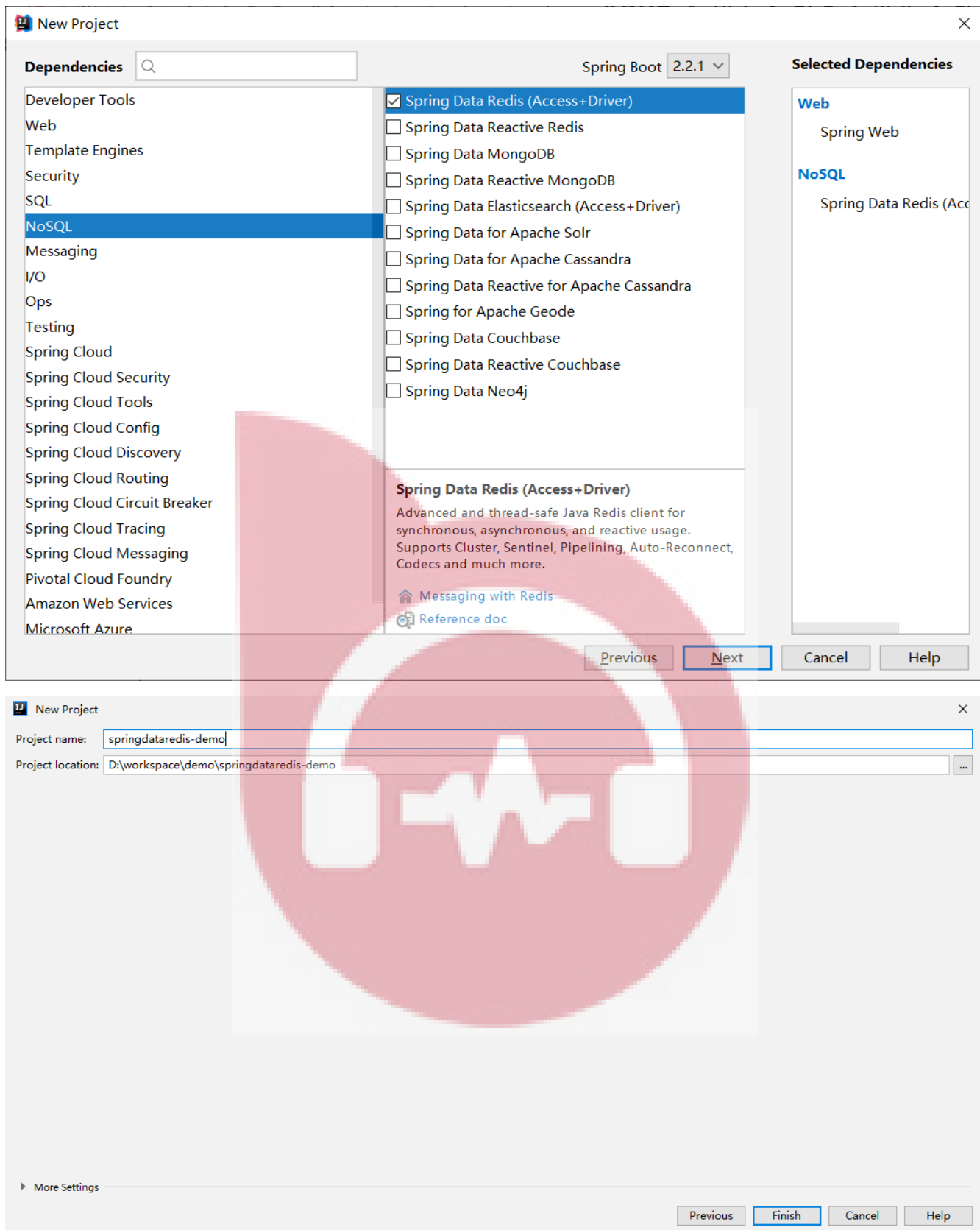
Version: 0.0.1-SNAPSHOT

Name: springdataredis-demo

Description: Demo project for Spring Boot

Package: com.xxxx.springdataredisdemo

Previous Next Cancel Help



添加依赖

```
<dependencies>
  <!-- spring data redis 组件 -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
```



```
<artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
<!-- commons-pool2 对象池依赖 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-pool2</artifactId>
</dependency>
<!-- web 组件 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<!-- test 组件 -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

添加application.yml配置文件

```
spring:
  redis:
    # Redis服务器地址
    host: 192.168.10.100
    # Redis服务器端口
    port: 6379
    # Redis服务器端口
    password: root
    # Redis服务器端口
    database: 0
    # 连接超时时间
    timeout: 10000ms
  lettuce:
    pool:
      # 最大连接数，默认8
      max-active: 1024
      # 最大连接阻塞等待时间，单位毫秒，默认-1ms
      max-wait: 10000ms
      # 最大空闲连接，默认8
      max-idle: 200
      # 最小空闲连接，默认0
      min-idle: 5
```

Lettuce和Jedis的区别

Jedis 是一个优秀的基于 Java 语言的 Redis 客户端，但是，其不足也很明显：Jedis 在实现上是直接连接 Redis-Server，在多个线程间共享一个 Jedis 实例时是线程不安全的，如果想要在多线程场景下使用 Jedis，需要使用连接池，每个线程都使用自己的 Jedis 实例，当连接数量增多时，会消耗较多的物理资源。

Lettuce 则完全克服了其线程不安全的缺点：Lettuce 是基于 Netty 的连接（StatefulRedisConnection），



Lettuce 是一个可伸缩的线程安全的 Redis 客户端，支持同步、异步和响应式模式。多个线程可以共享一个连接实例，而不必担心多线程并发问题。它基于优秀 Netty NIO 框架构建，支持 Redis 的高级功能，如 Sentinel，集群，流水线，自动重新连接和 Redis 数据模型。

测试环境测试环境是否搭建成功

```
@RunWith(SpringRunner.class)
@SpringBootTest(classes = SpringDataRedisApplication.class)
public class SpringDataRedisApplicationTests {

    @Autowired
    private RedisTemplate redisTemplate;
    @Autowired
    private StringRedisTemplate stringRedisTemplate;

    @Test
    public void initconn() {
        ValueOperations<String, String> ops = stringRedisTemplate.opsForValue();
        ops.set("username", "lisi");
        ValueOperations<String, String> value = redisTemplate.opsForValue();
        value.set("name", "wangwu");
        System.out.println(ops.get("name"));
    }
}
```