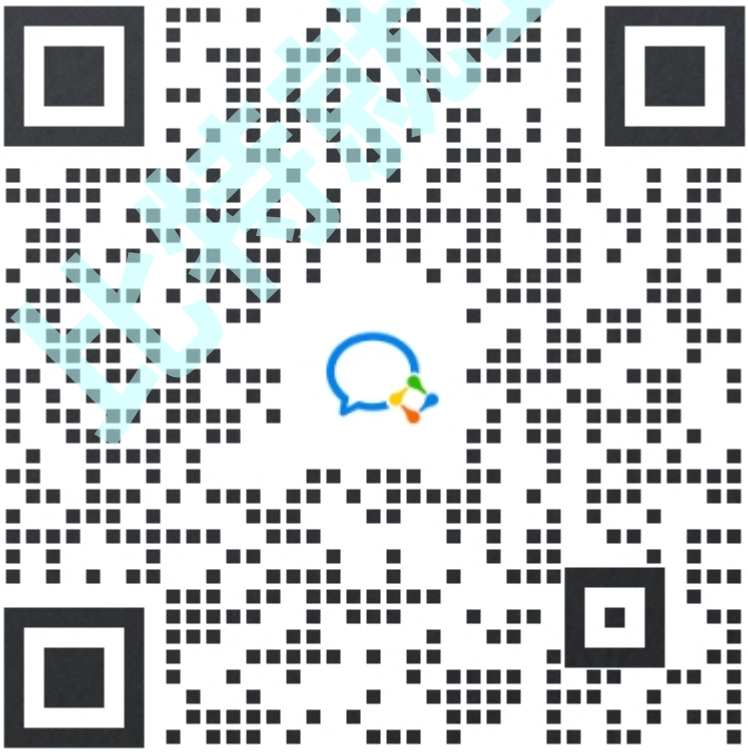
Redis Java 集成到 Spring Boot

版权说明

对比特课程感兴趣，可以联系这个微信。



代码 & 板书链接

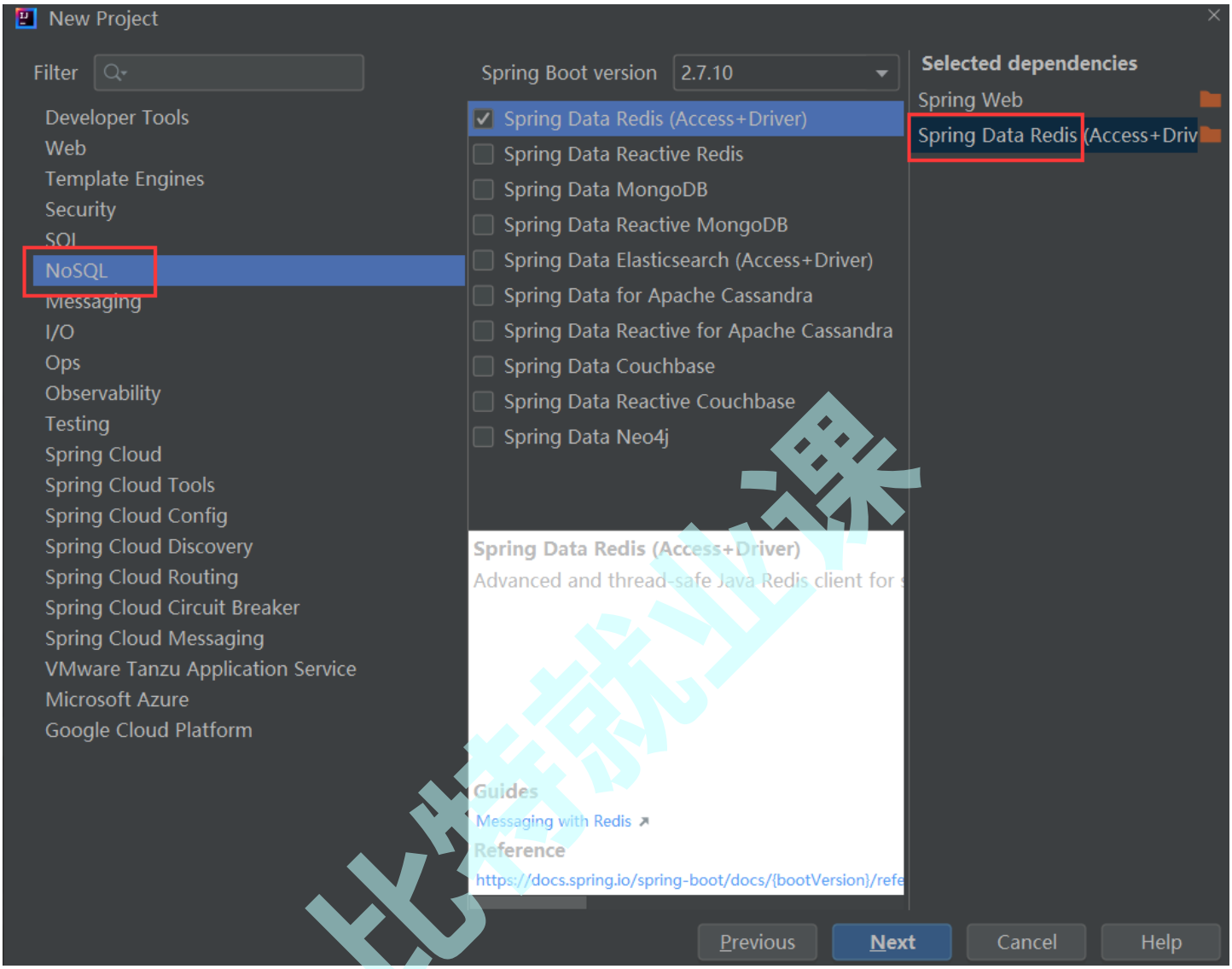
https://gitee.com/HGtz2222/bitproject/tree/master/redis

使用 Spring Boot 连接 Redis 单机

创建项目

勾选 NoSQL 中的 `Spring Data Redis`

> 当然, 把 Web 中的 Spring Web 也勾选一下. 方便写接口进行后续测试.



配置 redis 服务地址

在 application.yml 中配置.

```yaml
spring:
  redis:
    host: 127.0.0.1
    port: 8888
```

创建 Controller

创建 `MyController` 类

> 由于当前只是写简单的测试代码, 我们就不进行分层了. 就只创建个简单的 Controller 即可.

```
@RestController
public class MyController {
    @Autowired
    private StringRedisTemplate redisTemplate;

}
```

此处需要引入 `StringRedisTemplate` 实例.

使用 String

```
@GetMapping("/testString")
@ResponseBody
public String testString() {
    redisTemplate.opsForValue().set("key", "value");
    String value = redisTemplate.opsForValue().get("key");
    System.out.println(value);

    redisTemplate.delete("key");
    return "OK"
}
```

使用 List

```
@GetMapping("/testList")
@ResponseBody
public String testList() {
    redisTemplate.opsForList().leftPush("key", "a");
    redisTemplate.opsForList().leftPushAll("key", "b", "c", "d");
    List<String> values = redisTemplate.opsForList().range("key", 1, 2);
    System.out.println(values);

    redisTemplate.delete("key");
    return "OK";
}
```

使用 Hash

```
@GetMapping("/testHashmap")
@ResponseBody
public String testHashmap() {
    redisTemplate.opsForHash().put("key", "name", "zhangsan");
    String value = (String) redisTemplate.opsForHash().get("key", "name");
    System.out.println(value);
    redisTemplate.opsForHash().delete("key", "name");
    boolean ok = redisTemplate.opsForHash().hasKey("key", "name");
    System.out.println(ok);

    redisTemplate.delete("key");
    return "OK";
}
```

使用 Set

```
@GetMapping("/testSet")
@ResponseBody
public String testSet() {
    redisTemplate.opsForSet().add("key", "aaa", "bbb", "ccc");
    boolean ok = redisTemplate.opsForSet().isMember("key", "aaa");
    System.out.println(ok);
    redisTemplate.opsForSet().remove("key", "aaa");
    long n = redisTemplate.opsForSet().size("key");
    System.out.println(n);

    redisTemplate.delete("key");
    return "OK";
}
```

使用 ZSet

```java
@GetMapping("/testZSet")
@ResponseBody
public String testZSet() {
    redisTemplate.opsForZSet().add("key", "吕布", 100);
    redisTemplate.opsForZSet().add("key", "赵云", 98);
    redisTemplate.opsForZSet().add("key", "典韦", 95);

    Set<String> values = redisTemplate.opsForZSet().range("key", 0, 2);
    System.out.println(values);

    long n = redisTemplate.opsForZSet().count("key", 95, 100);
    System.out.println(n);

    redisTemplate.delete("key");
    return "OK";
}
```

执行结果

运行程序, 构造上述请求, 即可在服务器的控制台中看到执行结果.

使用 Spring Boot 连接 Redis 集群

配置 redis 集群地址

```yaml
spring:
  redis:
    cluster:
      nodes:
        - 172.30.0.101:6379
        - 172.30.0.102:6379
        - 172.30.0.103:6379
        - 172.30.0.104:6379
        - 172.30.0.105:6379
        - 172.30.0.106:6379
        - 172.30.0.107:6379
        - 172.30.0.108:6379
        - 172.30.0.109:6379
    lettuce:
      cluster:
        refresh:
```

```
17            adaptive: true
18            period: 2000
```

🐵 下方的 lettuce 系列配置, 目的是为了自动刷新集群的拓扑结构. 当集群中有节点宕机/加入新节点之后, 我们的代码能够自动感知到集群的变化.

改完配置之后, 其他代码无需做出任何调整, 直接就可以正常运行了.

⚽ 注意!

由于上述 ip 都是 docker 容器的 ip, 在 windows 主机上不能直接访问.

因此需要把程序打成 jar 包, 部署到 linux 上, 再通过 `java -jar [jar包名]` 的方式执行.

小结

Spring Boot 2 系列内置的 Redis 是 Lettuce, 和 Jedis 的使用还是存在一定的差异.

对于 Jedis 来说, 各个方法和 Redis 的命令基本是一致的.

而集成到 Spring Boot 之后, 接口上和原始 Redis 命令存在部分差别, 但是使用起来也并不困难, 只要大家熟悉 Redis 的基本操作, 还是很容易可以通过方法名字理解用法的.