SpringBoot+Redis

前瞻：

Reids的Java客户端有

1.Jedis

2.Lettuce

3.Spring Data Redis

我们主要使用Spring Data Redis

1.操作步骤：

1.1 导入Spring Data Redis 的maven坐标

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

1.2 配置redis数据源

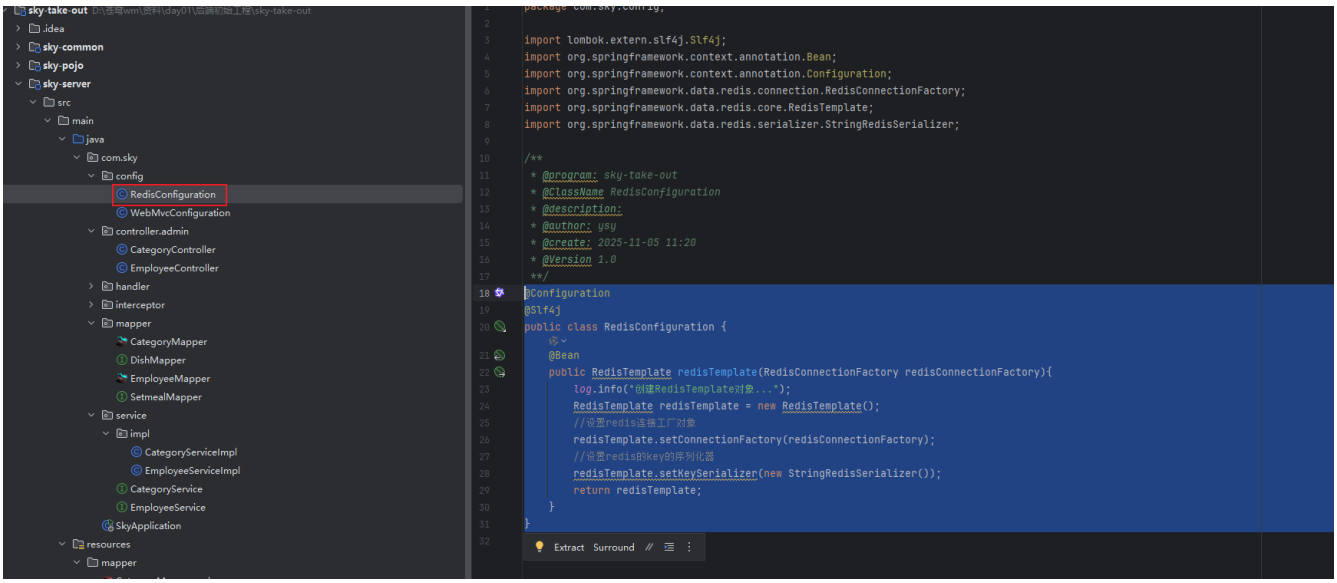
```
spring:
  redis:
    host: ${sky.redis.host}
    port: ${sky.redis.port}
    password: ${sky.redis.password}
    database: ${sky.redis.database}
```

```
sky:
  redis:
    host: localhost
    port: 6379
    password:
    database: 10
```

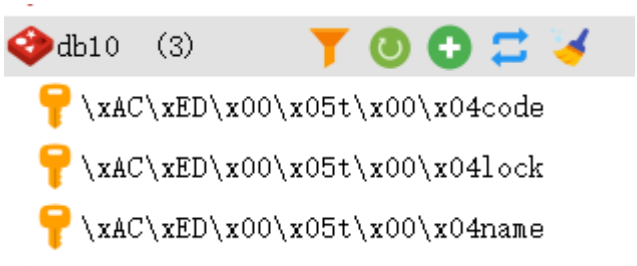
1.3 编写配置类，创建RedisTemplete对象

```
@Configuration
@Slf4j
public class RedisConfiguration {
    @Bean
    public RedisTemplate redisTemplate(RedisConnectionFactory redisConnectionFactory){
        log.info("创建RedisTemplate对象...");
```

```
        RedisTemplate redisTemplate = new RedisTemplate();
        //设置redis连接工厂对象
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        //设置redis的key的序列化器
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        return redisTemplate;
    }
}
```



加redis的key的序列化器之前：



加redis的key的序列化器之后：

1.4 通过Redis Template对象操作Redis

```
@SpringBootTest
public class SpringDataRedisTest {
    @Autowired
    private RedisTemplate redisTemplate;
    @Test
```

```
    public void testRedisTemplate(){
        System.out.println(redisTemplate);
        //操作字符串类型
        ValueOperations valueOperations = redisTemplate.opsForValue();
        HashOperations hashOperations = redisTemplate.opsForHash();
        ListOperations listOperations = redisTemplate.opsForList();
        SetOperations setOperations = redisTemplate.opsForSet();
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();
    }
    //String类型
    @Test
    public void testString(){
        //set get setex setnx
        redisTemplate.opsForValue().set("name","小明");
        String  city = (String) redisTemplate.opsForValue().get("name");
        System.out.println(city);
        redisTemplate.opsForValue().set("code","1234",3, TimeUnit.MINUTES);
        //setIfAbsent就是setnx
        redisTemplate.opsForValue().setIfAbsent("lock","1");
        redisTemplate.opsForValue().setIfAbsent("lock","2");
    }
    //Hash类型
    @Test
    public void testHash(){
        //hset hget hdel hkeys hvals
        HashOperations hashOperations = redisTemplate.opsForHash();
        //hset
        hashOperations.put("100","name","yinshunyu");
        hashOperations.put("100","age","20");
        //get
        String name = (String) hashOperations.get("100", "name");
        System.out.println(name);
        //hkeys
        Set keys = hashOperations.keys("100");
        System.out.println(keys);
        //hvals
        List values = hashOperations.values("100");
        System.out.println(values);
        //hdel
        hashOperations.delete("100","age");
    }
    //List类型
    @Test
    public void testList(){
        //lpush  lrange rpop llen
        ListOperations listOperations = redisTemplate.opsForList();
        listOperations.leftPushAll("mylist","a","b","c");
        listOperations.leftPush("mylist","d");
        List mylist = listOperations.range("mylist", 0, -1);
        System.out.println(mylist);
        listOperations.rightPop("mylist");
        Long size = listOperations.size("mylist");
        System.out.println(size);
```

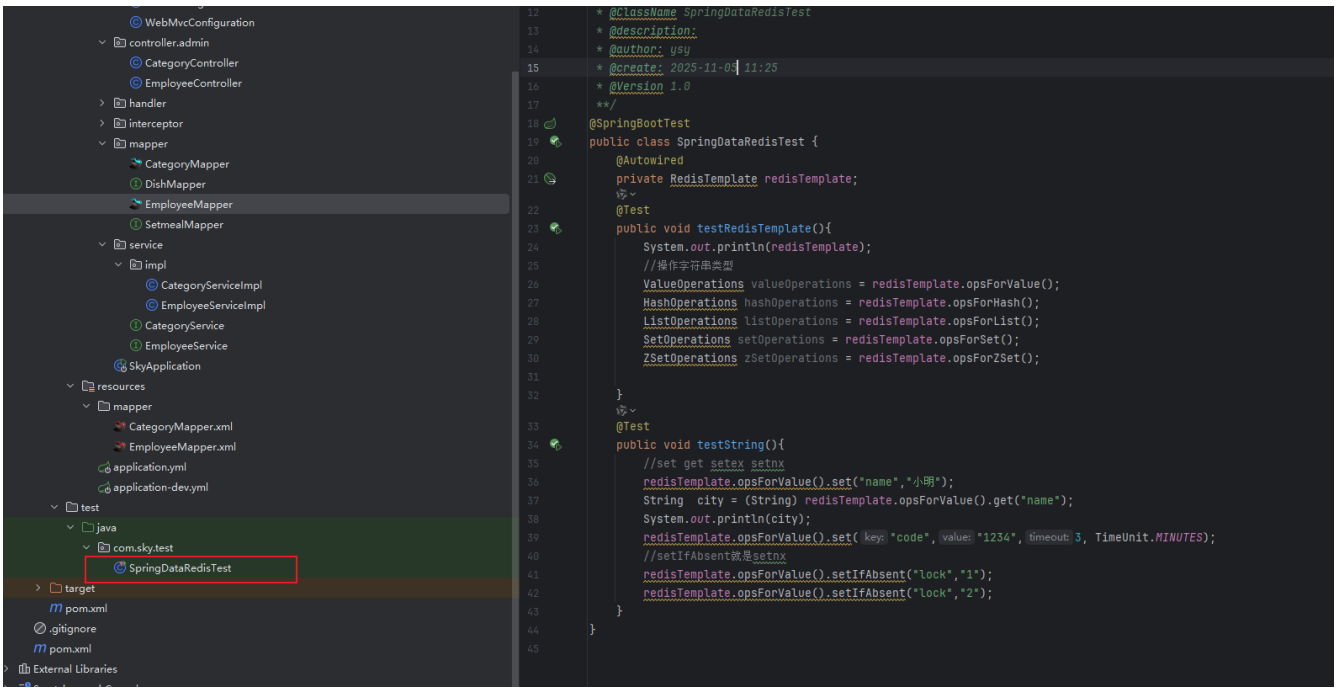
```
    }
    //Set类型
    @Test
    public void testSet(){
        //sadd smembers scard sinter sunion srem
        SetOperations setOperations = redisTemplate.opsForSet();
        setOperations.add("set1","a","b","c","d");
        setOperations.add("set2","a","b","x","y");
        //smembers 获取元素
        Set members = setOperations.members("set1");
        System.out.println(members);
        //scard 元素个数
        Long size = setOperations.size("set1");
        System.out.println(size);
        //sinter 交集
        Set intersect = setOperations.intersect("set1", "set2");
        System.out.println(intersect);
        //sunion 并集
        Set union = setOperations.union("set1", "set2");
        System.out.println( union);
        setOperations.remove("set1","a","b");
    }
    //ZSet类型
     @Test
    public void testZset(){
        //zadd zrange zincrby zrem
        ZSetOperations zSetOperations = redisTemplate.opsForZSet();

        zSetOperations.add("zset1","a",10);
        zSetOperations.add("zset1","b",12);
        zSetOperations.add("zset1","c",9);

        Set zset1 = zSetOperations.range("zset1", 0, -1);
        System.out.println(zset1);
        //zincrby 修改分数
        zSetOperations.incrementScore("zset1","c",10);

        zSetOperations.remove("zset1","a","b");
    }

}
```



2.缓存菜品模拟

修改用户端接口 DishController 的 list 方法，加入缓存处理逻辑：

```
@Autowired
private RedisTemplate redisTemplate;
/**
 * 根据分类id查询菜品
 *
 * @param categoryId
 * @return
 */
@GetMapping("/list")
@ApiOperation("根据分类id查询菜品")
public Result<List<DishVO>> list(Long categoryId) {

    //构造redis中的key，规则：dish_分类id
    String key = "dish_" + categoryId;

    //查询redis中是否存在菜品数据
    List<DishVO> list = (List<DishVO>) redisTemplate.opsForValue().get(key);
    if(list != null && list.size() > 0){
        //如果存在，直接返回，无须查询数据库
        return Result.success(list);
    }
    //////////////////////////////////////////////////////////
    Dish dish = new Dish();
    dish.setCategoryId(categoryId);
    dish.setStatus(StatusConstant.ENABLE);//查询起售中的菜品

    //如果不存在，查询数据库，将查询到的数据放入redis中
    list = dishService.listWithFlavor(dish);
    //////////////////////////////////////////////////////////
```

```
        redisTemplate.opsForValue().set(key, list);

        return Result.success(list);
    }
```

抽取清理缓存的方法：

在管理端DishController中添加

```
    @Autowired
    private RedisTemplate redisTemplate;
    /**
     * 清理缓存数据
     * @param pattern
     */
    private void cleanCache(String pattern){
        Set keys = redisTemplate.keys(pattern);
        redisTemplate.delete(keys);
    }
```

调用清理缓存的方法，保证数据一致性：

1). 新增菜品优化

```
    /**
     * 新增菜品
     *
     * @param dishDTO
     * @return
     */
    @PostMapping
    @ApiOperation("新增菜品")
    public Result save(@RequestBody DishDTO dishDTO) {
        log.info("新增菜品：{}", dishDTO);
        dishService.saveWithFlavor(dishDTO);

        //清理缓存数据
        String key = "dish_" + dishDTO.getCategoryId();
        cleanCache(key);
        return Result.success();
    }
```

EasyExcel初步理解

依赖导入

```
<!-- https://mvnrepository.com/artifact/com.alibaba/easyexcel -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>easyexcel</artifactId>
            <version>2.1.1</version>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.10</version>
```

创建与表格对应的实体类

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {

    @ExcelProperty(value = "学生id")
    private Integer id;
    @ExcelProperty(value = "学生姓名")
    private String name;

    @ExcelProperty(value = "学生年龄")
    private Integer age;
}
```

写操作

```
public class WriteExcel {
    public static void main(String[] args) {

        //准备文件路径
        String fileName="D:/destory/test/easyexcel.xls";
        //写出文件
        EasyExcel.write(fileName, Student.class).sheet("easyexcel")
                .doWrite(data());
    }


    private static List<Student> data(){
        ArrayList<Student> list = new ArrayList<>();
        for (int i = 0; i < 10; i++) {
            Student student = new Student(i, "董德" + 1, 22 + i);
            list.add(student);
        }
        return  list;
```

```
    }
}
```

读操作

改造实体类

```
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Student {

    @ExcelProperty(value = "学生id",index = 0)
    private Integer id;
    @ExcelProperty(value = "学生姓名",index = 1)
    private String name;

    @ExcelProperty(value = "学生年龄",index = 2)
    private Integer age;
}
```

创建监听器

```
public class EasyExcelLinster extends AnalysisEventListener<Student> {

  List<Student> list=  new ArrayList<Student>();
    //一行一行的去读取里面的数据
    @Override
    public void invoke(Student student, AnalysisContext analysisContext) {
        System.out.println(student);
        list.add(student);
    }

    @Override
    public void doAfterAllAnalysed(AnalysisContext analysisContext) {

    }


}
```

读取

```
public class ReadExcel {
    public static void main(String[] args) {
        //准备文件路径
        String fileName="D:/destory/test/easyexcel.xls";
        EasyExcel.read(fileName, Student.class, new ExcelListener()).sheet().doRead();
    }
}
```