# 乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取

乐字节晓啡

乐字节官方交流群

# 商城项目中集成Redis实现缓存

## shop-manager引入依赖

pom.xml

```
<!-- spring data redis 依赖 -->
<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>
```

application.yml

```
# Redis配置
  redis:
    timeout: 10000ms                        # 连接超时时间
    host: 192.168.10.100                   # Redis服务器地址
    port: 6379                              # Redis服务器端口
    database: 0                             # 选择哪个库，默认0库
    lettuce:
      pool:
        max-active: 1024                    # 最大连接数，默认 8
        max-wait: 10000ms                   # 最大连接阻塞等待时间，单位毫秒，默认 -1
        max-idle: 200                       # 最大空闲连接，默认 8
        min-idle: 5                          # 最小空闲连接，默认 0
# Redis Key
# 商品分类列表 Key
goods.category.list.key: goods:category:list:goodsCategoryList
```

RedisConfig.java

```
package com.xxxx.manager.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;

/**
 * Redis配置类
 *
 * @author zhoubin
 * @since 1.0.0
 */
@Configuration
public class RedisConfig {
    @Bean
    public RedisTemplate<String,Object> redisTemplate(LettuceConnectionFactory 
redisConnectionFactory){
        RedisTemplate<String,Object> redisTemplate = new RedisTemplate<>();
        //为string类型key设置序列器
        redisTemplate.setKeySerializer(new StringRedisSerializer());
        //为string类型value设置序列器
        redisTemplate.setValueSerializer(new GenericJackson2JsonRedisSerializer());
        //为hash类型key设置序列器
        redisTemplate.setHashKeySerializer(new StringRedisSerializer());
        //为hash类型value设置序列器
        redisTemplate.setHashValueSerializer(new GenericJackson2JsonRedisSerializer());
        redisTemplate.setConnectionFactory(redisConnectionFactory);
        return redisTemplate;
    }
}
```

shop-common添加Json工具类

JsonUtil.java

```
package com.xxxx.common.util;

import com.fasterxml.jackson.core.JsonParseException;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JavaType;
import com.fasterxml.jackson.databind.JsonMappingException;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.IOException;
import java.util.List;

/**
 * @see(功能介绍) : Json转换工具类
 * @version(版本号) : 1.0
 * @author(创建人) : zhoubin
 */
public class JsonUtil {
    private static ObjectMapper objectMapper = new ObjectMapper();

    /**
     * 将对象转换成json字符串
     *
     * @param obj
     * @return
     */
    public static String object2JsonStr(Object obj) {
```

```java
        try {
            return objectMapper.writeValueAsString(obj);
        } catch (JsonProcessingException e) {
            //打印异常信息
            e.printStackTrace();
        }
        return null;
    }

    /**
     * 将字符串转换为对象
     *
     * @param <T> 泛型
     */
    public static <T> T jsonStr2Object(String jsonStr, Class<T> clazz) {
        try {
            return objectMapper.readValue(jsonStr.getBytes("UTF-8"), clazz);
        } catch (JsonParseException e) {
            e.printStackTrace();
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }

    /**
     * 将json数据转换成pojo对象list
     * <p>Title: jsonToList</p>
     * <p>Description: </p>
     *
     * @param jsonStr
     * @param beanType
     * @return
     */
    public static <T> List<T> jsonToList(String jsonStr, Class<T> beanType) {
        JavaType javaType =
objectMapper.getTypeFactory().constructParametricType(List.class, beanType);
        try {
            List<T> list = objectMapper.readValue(jsonStr, javaType);
            return list;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return null;
    }
}
```