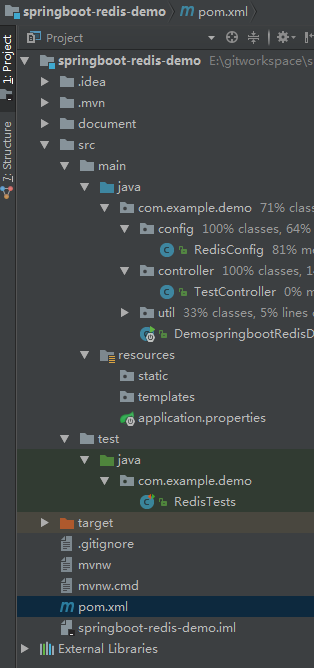
# springboot-redis-demo说明

## 项目结构：



## Pom.xml引入的jar包

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.example</groupId>  
 <artifactId>demo</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>demospringboot-redis-demo</name>  
 <description>Demo project for Spring Boot</description>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>1.5.10.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-redis</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
  
 <dependency>  
 <groupId>com.fasterxml.jackson.datatype</groupId>  
 <artifactId>jackson-datatype-jsr310</artifactId>  
 </dependency>  
  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
  
</project>

## 配置文件application.properties

server.context-path=/test1  
server.port=8080  
  
server.tomcat.uri-encoding=UTF-8  
spring.http.encoding.charset=UTF-8  
spring.http.encoding.enabled=true  
spring.http.encoding.force=true  
spring.messages.encoding=UTF-8  
  
# =============================================================  
spring.application.name=springboot-mybatis  
spring.application.index=0  
  
spring.devtools.restart.exclude=static/\*\*  
spring.devtools.restart.additional-exclude=static/\*\*,public/\*\*  
  
spring.session.store-type=none  
  
spring.profiles.active=dev  
  
  
# ============================================ 日志 ============  
logging.pattern.console=%d{yyyy/MM/dd-HH:mm:ss} [%thread] %-5level %logger- %msg%n   
logging.pattern.file=%d{yyyy/MM/dd-HH:mm} [%thread] %-5level %logger- %msg%n   
logging.level.com.example.demo.dao=DEBUG  
logging.file=E:/testspace/log.log  
  
# ==============================================================  
  
# REDIS (RedisProperties)  
# Redis数据库索引（默认为0）  
spring.redis.database=0   
# Redis服务器地址  
spring.redis.host=127.0.0.1  
# Redis服务器连接端口  
spring.redis.port=6379   
# Redis服务器连接密码（默认为空）  
spring.redis.password=  
# 连接池最大连接数（使用负值表示没有限制）  
spring.redis.pool.max-active=8   
# 连接池最大阻塞等待时间（使用负值表示没有限制）  
spring.redis.pool.max-wait=-1   
# 连接池中的最大空闲连接  
spring.redis.pool.max-idle=8   
# 连接池中的最小空闲连接  
spring.redis.pool.min-idle=0   
# 连接超时时间（毫秒）  
spring.redis.timeout=0   
#spring.redis.sentinel.master=mymaster  
#spring.redis.sentinel.nodes=10.100.11.248:26379,10.100.11.249:26379,10.100.11.250:26379

## Config：

package com.example.demo.config;  
  
import com.fasterxml.jackson.annotation.JsonAutoDetect;  
import com.fasterxml.jackson.annotation.PropertyAccessor;  
import com.fasterxml.jackson.databind.ObjectMapper;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.cache.CacheManager;  
import org.springframework.cache.annotation.CachingConfigurerSupport;  
import org.springframework.cache.annotation.EnableCaching;  
import org.springframework.cache.interceptor.KeyGenerator;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.redis.cache.RedisCacheManager;  
import org.springframework.data.redis.connection.RedisConnectionFactory;  
import org.springframework.data.redis.core.\*;  
import org.springframework.data.redis.serializer.Jackson2JsonRedisSerializer;  
import org.springframework.data.redis.serializer.JdkSerializationRedisSerializer;  
import org.springframework.data.redis.serializer.RedisSerializer;  
import org.springframework.data.redis.serializer.StringRedisSerializer;  
  
import java.lang.reflect.Method;  
  
*/\*\*  
 \* Created by liulanhua on 2018/2/6.  
 \*/*@Configuration  
@EnableCaching  
public class RedisConfig extends CachingConfigurerSupport {  
  
 @Bean(name="redisTemplate")  
 public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory factory) {  
  
 RedisTemplate<String, Object> template = new RedisTemplate<>();  
  
 RedisSerializer<String> redisSerializer = new StringRedisSerializer();  
  
 Jackson2JsonRedisSerializer jackson2JsonRedisSerializer = new Jackson2JsonRedisSerializer(Object.class);  
 ObjectMapper om = new ObjectMapper();  
 om.setVisibility(PropertyAccessor.*ALL*, JsonAutoDetect.Visibility.*ANY*);  
 om.enableDefaultTyping(ObjectMapper.DefaultTyping.*NON\_FINAL*);  
 jackson2JsonRedisSerializer.setObjectMapper(om);  
  
 template.setConnectionFactory(factory);  
 //key序列化方式  
 template.setKeySerializer(redisSerializer);  
 //value序列化  
 template.setValueSerializer(jackson2JsonRedisSerializer);  
 //value hashmap序列化  
 template.setHashValueSerializer(jackson2JsonRedisSerializer);  
  
 return template;  
 }  
  
 */\*\*  
 \* 设置数据存入 redis 的序列化方式  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@param*** *redisConnectionFactory  
 \*/* private void initDomainRedisTemplate(RedisTemplate<String, Object> redisTemplate, RedisConnectionFactory redisConnectionFactory) {  
 redisTemplate.setKeySerializer(new StringRedisSerializer());  
 redisTemplate.setHashKeySerializer(new StringRedisSerializer());  
 redisTemplate.setHashValueSerializer(new JdkSerializationRedisSerializer());  
 redisTemplate.setValueSerializer(new JdkSerializationRedisSerializer());  
 redisTemplate.setConnectionFactory(redisConnectionFactory);  
 }  
  
 */\*\*  
 \* 实例化 HashOperations 对象,可以使用 Hash 类型操作  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@return*** *\*/* @Bean  
 public HashOperations<String, String, Object> hashOperations(RedisTemplate<String, Object> redisTemplate) {  
 return redisTemplate.opsForHash();  
 }  
  
 */\*\*  
 \* 实例化 ValueOperations 对象,可以使用 String 操作  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@return*** *\*/* @Bean  
 public ValueOperations<String, Object> valueOperations(RedisTemplate<String, Object> redisTemplate) {  
 return redisTemplate.opsForValue();  
 }  
  
 */\*\*  
 \* 实例化 ListOperations 对象,可以使用 List 操作  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@return*** *\*/* @Bean  
 public ListOperations<String, Object> listOperations(RedisTemplate<String, Object> redisTemplate) {  
 return redisTemplate.opsForList();  
 }  
  
 */\*\*  
 \* 实例化 SetOperations 对象,可以使用 Set 操作  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@return*** *\*/* @Bean  
 public SetOperations<String, Object> setOperations(RedisTemplate<String, Object> redisTemplate) {  
 return redisTemplate.opsForSet();  
 }  
  
 */\*\*  
 \* 实例化 ZSetOperations 对象,可以使用 ZSet 操作  
 \*  
 \** ***@param*** *redisTemplate  
 \** ***@return*** *\*/* @Bean  
 public ZSetOperations<String, Object> zSetOperations(RedisTemplate<String, Object> redisTemplate) {  
 return redisTemplate.opsForZSet();  
 }  
  
 @Bean  
 @Override  
 public KeyGenerator keyGenerator() {  
 return new KeyGenerator() {  
 @Override  
 public Object generate(Object target, Method method, Object... params) {  
 StringBuilder sb = new StringBuilder();  
 sb.append(target.getClass().getName());  
 sb.append(method.getName());  
 for (Object obj : params) {  
 sb.append(obj.toString());  
 }  
 return sb.toString();  
 }  
 };  
 }  
  
 @SuppressWarnings("rawtypes")  
 @Bean  
 public CacheManager cacheManager(RedisTemplate redisTemplate) {  
 RedisCacheManager rcm = new RedisCacheManager(redisTemplate);  
 //设置缓存过期时间  
 //rcm.setDefaultExpiration(60);//秒  
 return rcm;  
 }  
}

## 启动入口

package com.example.demo;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class DemospringbootRedisDemoApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(DemospringbootRedisDemoApplication.class, args);  
 }  
}

## 其他

Util

RedisClient

package com.example.demo.util;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.dao.DataAccessException;  
import org.springframework.data.redis.connection.RedisConnection;  
import org.springframework.data.redis.core.HashOperations;  
import org.springframework.data.redis.core.RedisCallback;  
import org.springframework.data.redis.core.RedisTemplate;  
import org.springframework.data.redis.core.ValueOperations;  
import org.springframework.stereotype.Component;  
  
import java.util.Optional;  
  
*/\*\*  
 \* Created by zhoufurong on 2017/11/1.  
 \*/  
  
/\*\*  
 \* Redis客户端  
 \*/*@Component  
public class RedisClient {  
  
 @Autowired  
 private RedisTemplate redisTemplate;  
  
 public boolean isExists(String key){  
  
 return (boolean) redisTemplate.execute(new RedisCallback<Boolean>() {  
 @Override  
 public Boolean doInRedis(RedisConnection redisConnection) throws DataAccessException {  
 return redisConnection.exists(key.getBytes());  
 }  
 });  
  
 }  
  
 public void delete(String key){  
 redisTemplate.delete(key);  
 }  
  
 public void set(String key, String value) {  
 ValueOperations valueOperations = redisTemplate.opsForValue();  
 valueOperations.set(key, value);  
 }  
 public void set(String key, final String value,final long expireTime) {  
 redisTemplate.execute((RedisCallback) redisConnection -> {  
 redisConnection.set(key.getBytes(),value.getBytes());  
 return null;  
 });  
  
 }  
  
  
 public Optional<String> get( String key) {  
 String result = (String) redisTemplate.execute((RedisCallback) redisConnection -> {  
 String result1 = null;  
 try {  
 result1 = new String(redisConnection.get(key.getBytes()));  
 }catch (Exception e){  
 return null;  
 }  
 return result1;  
  
 });  
 return result == null ? Optional.*empty*() : Optional.*of*(result.toString());  
 }  
  
 public void hashSet(String key, String field, String value) {  
 HashOperations hashOperations = redisTemplate.opsForHash();  
 hashOperations.put(key, field, value);  
  
 }  
  
  
 public Optional<String> hashGet(String key, String field) {  
 HashOperations hashOperations = redisTemplate.opsForHash();  
 Object result = hashOperations.get(key, field);  
 return result == null ? Optional.*empty*() : Optional.*of*(result.toString());  
 }  
  
  
  
  
}

JsonUtils

package com.example.demo.util;  
  
import com.fasterxml.jackson.annotation.JsonInclude;  
import com.fasterxml.jackson.core.JsonFactory;  
import com.fasterxml.jackson.core.JsonGenerator;  
import com.fasterxml.jackson.core.JsonParser;  
import com.fasterxml.jackson.core.JsonProcessingException;  
import com.fasterxml.jackson.core.type.TypeReference;  
import com.fasterxml.jackson.databind.\*;  
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.io.IOException;  
import java.io.StringWriter;  
import java.util.List;  
import java.util.Map;  
  
*/\*\*  
 \* Created by liulanhua on 2018/3/19.  
 \*/*public class JsonUtils {  
  
 private static final Logger *logger* = LoggerFactory.*getLogger*(JsonUtils.class);  
 public static final ObjectMapper *OBJECT\_MAPPER* = *createObjectMapper*();  
  
 public JsonUtils() {  
 }  
  
 private static ObjectMapper createObjectMapper() {  
 ObjectMapper objectMapper = new ObjectMapper();  
 objectMapper.configure(JsonParser.Feature.*ALLOW\_UNQUOTED\_FIELD\_NAMES*, true);  
 objectMapper.configure(DeserializationFeature.*ACCEPT\_EMPTY\_STRING\_AS\_NULL\_OBJECT*, true);  
 objectMapper.configure(DeserializationFeature.*FAIL\_ON\_UNKNOWN\_PROPERTIES*, false);  
 objectMapper.configure(DeserializationFeature.*USE\_BIG\_DECIMAL\_FOR\_FLOATS*, true);  
 objectMapper.setSerializationInclusion(JsonInclude.Include.*NON\_NULL*);  
 objectMapper.configure(SerializationFeature.*WRITE\_DATES\_AS\_TIMESTAMPS*, false);  
 objectMapper.registerModule(new JavaTimeModule());  
 return objectMapper;  
 }  
  
 public static String object2Json(Object o) {  
 StringWriter sw = new StringWriter();  
 JsonGenerator gen = null;  
  
 try {  
 gen = (new JsonFactory()).createGenerator(sw);  
 *OBJECT\_MAPPER*.writeValue(gen, o);  
 } catch (IOException var11) {  
 throw new RuntimeException("不能序列化对象为Json", var11);  
 } finally {  
 if(null != gen) {  
 try {  
 gen.close();  
 } catch (IOException var10) {  
 throw new RuntimeException("不能序列化对象为Json", var10);  
 }  
 }  
  
 }  
  
 return sw.toString();  
 }  
  
 public static Map<String, Object> object2Map(Object o) {  
 return (Map)*OBJECT\_MAPPER*.convertValue(o, Map.class);  
 }  
  
 public static <T> T json2Object(String json, Class<T> clazz) {  
 try {  
 return *OBJECT\_MAPPER*.readValue(json, clazz);  
 } catch (IOException var3) {  
 throw new RuntimeException("将 Json 转换为对象时异常,数据是:" + json, var3);  
 }  
 }  
  
 public static <T> List<T> json2List(String json, Class<T> clazz) throws IOException {  
 JavaType type = *OBJECT\_MAPPER*.getTypeFactory().constructCollectionType(List.class, clazz);  
 List<T> list = (List)*OBJECT\_MAPPER*.readValue(json, type);  
 return list;  
 }  
  
 public static <T> T[] json2Array(String json, Class<T[]> clazz) throws IOException {  
 return *OBJECT\_MAPPER*.readValue(json, clazz);  
 }  
  
 public static <T> T node2Object(JsonNode jsonNode, Class<T> clazz) {  
 try {  
 T t = *OBJECT\_MAPPER*.treeToValue(jsonNode, clazz);  
 return t;  
 } catch (JsonProcessingException var3) {  
 throw new RuntimeException("将 Json 转换为对象时异常,数据是:" + jsonNode.toString(), var3);  
 }  
 }  
  
 public static JsonNode object2Node(Object o) {  
 try {  
 return (JsonNode)(o == null?*OBJECT\_MAPPER*.createObjectNode():(JsonNode)*OBJECT\_MAPPER*.convertValue(o, JsonNode.class));  
 } catch (Exception var2) {  
 throw new RuntimeException("不能序列化对象为Json", var2);  
 }  
 }  
  
 public static <T> T json2GenericObject(String json, TypeReference<T> tr) {  
 if(json != null && !"".equals(json)) {  
 try {  
 return *OBJECT\_MAPPER*.readValue(json, tr);  
 } catch (Exception var3) {  
 throw new RuntimeException("将 Json 转换为对象时异常,数据是:" + json, var3);  
 }  
 } else {  
 throw new RuntimeException("将 Json 转换为对象时异常,数据是:" + json);  
 }  
 }  
}

## 单元测试结果展示

