# Redis序列化方式

设置缓存redis和@Cacheable中的value和hash的value使用json序列化方式
源码:

```
/**
   * 注入RedisTemplate
   */
  @Primary
  @Bean
  public RedisTemplate<String, Object> redisTemplate(RedisConnectionFactory
      // 指定序列化方式
      RedisTemplate<String, Object> redisTemplate = new RedisTemplate<>();
      redisTemplate.setKeySerializer(RedisSerializer.string());
      redisTemplate.setHashKeySerializer(RedisSerializer.string());
      redisTemplate.setValueSerializer(RedisSerializer.json());
      redisTemplate.setHashValueSerializer(RedisSerializer.json());
      redisTemplate.setConnectionFactory(factory);
      return redisTemplate;
  }
  /**
   * 配置使用@Cachable时在redis中序列化的方式
   */
    @Bean
  public RedisCacheConfiguration redisCacheConfiguration() {
      RedisSerializationContext.SerializationPair<Object> pair = RedisSeria
      return RedisCacheConfiguration.defaultCacheConfig().serializeValuesWi
  }
```

# Spring cahce生成key的格式

设置@Cachable在redis生成的key的规则为 “应用名:类名.方法(入参列表)”
源码:

```java
/**
 * 配置使用@Cachable在redis生成的key的规则
 * <p>
 * 格式：应用名:类名.方法(入参列表)
 * 示例：order:OrderService.getOrderById(1111)
 */
@Bean
@Override
public KeyGenerator keyGenerator() {
    return (target, method, params) -> {
        String paramStr = Stream.of(params).map(Object::toString).collect(Col
        return appName + ":" + target.getClass().getSimpleName() + "." + meth
    };
}
```

# 使用@Cacheable时Redis过期时间配置

spring原生的@Cacheable在配合redis使用时并不支持自定义不同的过期时间，框架对@Cacheable进行了扩展，增加了@CacheExpire注解，支持自定义redis过期时间
使用方式：

```java
@CacheExpire(2000)
@Cacheable("myCache")
public String getCache(String arg) {
  ...
}
```

## @CacheExpire

支持两个属性，分别为失效时间（秒）和随机增加的秒数，防止缓存穿透

```java
/**
 * 失效时间（秒）
 */
int value();

/**
 * 上下浮动范围（秒）
 * 根据浮动范围生成随机秒数加在失效时间后，防止缓存穿透
 */
float floatRange() default 0.0f;
```