***JvmCache - JVM*内存缓存依赖库**

### 背景

##### 1.1 功能概述

JvmCache`是一个基于Caffeine实现的高性能的 JVM内存缓存工具，提供了一系列用于操作 JVM 缓存的方法，包括获取缓存键、获取缓存值、设置缓存、检查缓存是否存在、删除缓存等功能。

## 应用配置参数说明

JvmCache 提供了两个可配置的参数，用于调整缓存的性能和容量。这些参数可以通过环境变量或配置文件进行设置。

1. `initialCapacity`

描述：初始的缓存空间大小

单位：MB（兆字节）

默认值：

开发环境：10 MB

生产环境：10 MB

配置示例：`initialCapacity=20`

2. `maximumSize`

描述：缓存的最大条目数

单位：条目数量

默认值：

开发环境：10,000 条

生产环境：10,000 条

配置示例：`maximumSize=20000` 按照我的生成

## 方法说明

### **1. getAllKeys**

****功能描述****：获取所有缓存键。

****入参****：无。

****出参****：List<String>，所有缓存键的集合。

****日志记录****：

info级别：当缓存为空时，记录 “获取所有缓存键时，缓存为空”。

### **2. getAllCache**

* ****功能描述****：获取缓存中的所有键值对。
* ****入参****：无。
* ****出参****：Map<String, String>，包含所有键值对的 Map。
* ****日志记录****：
  + info级别：当缓存为空时，记录 “获取所有缓存时，缓存为空”。
  + error级别：当获取缓存过程中出现异常时，记录错误信息。

### **3. exist**

* ****功能描述****：检查键是否存在。
* ****入参****：String key，缓存键。
* ****出参****：Boolean，如果键存在返回 true，否则返回 false。
* ****日志记录****：
  + warn级别：当 key 为空或空白时，记录警告信息。

### **4. existHash**

* ****功能描述****：判断哈希键是否存在。
* ****入参****：
  + String key，主键。
  + String hashField，哈希字段。
* ****出参****：Boolean，如果存在返回 true，否则返回 false。
* ****日志记录****：
  + warn级别：当 key 或 hashField 为空或空白时，记录警告信息。

### **5. setCache**

* ****功能描述****：设置缓存。
* ****入参****：
  + String key，缓存键。
  + String value，缓存值。
* ****出参****：Boolean，存入成功返回 true，否则返回 false。
* ****日志记录****：
  + info级别：当缓存设置成功时，记录键和值。
  + error级别：当设置缓存失败时，记录错误信息。

### **6. setCacheWithExpire**

* ****功能描述****：设置带过期时间的缓存。
* ****入参****：
  + String key，缓存键。
  + String value，缓存值。
  + Long expire，过期时间（毫秒）。
* ****出参****：Boolean，设置成功返回 true，否则返回 false。
* ****日志记录****：
  + info级别：当缓存设置成功时，记录键、值和过期时间。
  + warn级别：当 key 为空或 expire 无效时，记录警告信息。
  + error级别：当设置缓存失败时，记录错误信息。

### **7. setHashCache**

* ****功能描述****：设置哈希缓存。
* ****入参****：
  + String key，缓存键。
  + String hashField，哈希字段。
  + String value，缓存值。
* ****出参****：Boolean，设置成功返回 true，否则返回 false。
* ****日志记录****：
  + info级别：当哈希缓存设置成功时，记录键、字段和值。
  + warn级别：当 key 或 hashField 为空或空白时，记录警告信息。
  + error级别：当设置哈希缓存失败时，记录错误信息。

### **8. setHashCacheWithExpire**

* ****功能描述****：设置带过期时间的哈希缓存。
* ****入参****：
  + String key，缓存键。
  + String hashField，哈希字段。
  + String value，缓存值。
  + Long expire，过期时间（毫秒）。
* ****出参****：Boolean，设置成功返回 true，否则返回 false。
* ****日志记录****：
  + info级别：当哈希缓存设置成功时，记录键、字段、值和过期时间。
  + warn级别：当参数无效时，记录警告信息。
  + error级别：当设置哈希缓存失败时，记录错误信息。

### **9. getCache**

* ****功能描述****：获取缓存。
* ****入参****：String key，缓存键。
* ****出参****：String，缓存值，如果不存在或已过期返回 null。
* ****日志记录****：
  + info级别：当获取缓存成功时，记录键和值；当键不存在或已过期时，记录相应信息。
  + error级别：当获取缓存过程中出现异常时，记录错误信息。

### **10. getHashCache**

* ****功能描述****：获取哈希缓存。
* ****入参****：
  + String key，缓存键。
  + String hashField，哈希字段。
* ****出参****：String，缓存值，如果不存在或已过期返回 null。
* ****日志记录****：
  + info级别：当获取哈希缓存成功时，记录键、字段和值；当键或字段不存在或已过期时，记录相应信息。
  + error级别：当获取哈希缓存过程中出现异常时，记录错误信息。

### **11. getMultiCache**

* ****功能描述****：批量获取缓存值。
* ****入参****：List<String> keys，要获取的键的集合。
* ****出参****：Map<String, String>，包含存在的键值对的 Map。
* ****日志记录****：
  + warn级别：当键列表为空时，记录警告信息。
  + info级别：当批量获取缓存成功时，记录获取到的键值对数量。
  + error级别：当批量获取缓存过程中出现异常时，记录错误信息。

### **12. getOrComputeAndSet**

* ****功能描述****：获取缓存，如果缓存不存在则执行 function，并将查询的值放到缓存中。
* ****入参****：
  + String key，缓存键。
  + Long expireAfterWriteMillis，写入后过期时间（毫秒）。
  + Function<String, String> computeFunction，如果缓存不存在时，用于计算值的函数。
* ****出参****：String，缓存值或计算后的新值。
* ****日志记录****：
  + info级别：当从缓存中获取到值时，记录键和值；当计算并设置缓存成功时，记录键、值和过期时间。
  + warn级别：当计算值为 null，不设置缓存时，记录相应信息。
  + error级别：当操作过程中出现异常时，记录错误信息。

### **13. deleteCache**

* ****功能描述****：删除缓存。
* ****入参****：String key，缓存键。
* ****出参****：Boolean，删除成功返回 true，否则返回 false。
* ****日志记录****：
  + warn级别：当 key 为空或空白时，记录警告信息。
  + info级别：当删除缓存成功时，记录键。
  + error级别：当删除缓存过程中出现异常时，记录错误信息。

### **14. deleteHashCache**

* ****功能描述****：删除哈希缓存。
* ****入参****：
  + String key，缓存键。
  + String hashField，哈希字段。
* ****出参****：Boolean，删除成功返回 true，否则返回 false。
* ****日志记录****：
  + warn级别：当 key 或 hashField 为空或空白时，记录警告信息。
  + info级别：当删除哈希缓存成功时，记录键和字段。
  + error级别：当删除哈希缓存过程中出现异常时，记录错误信息。

### **15. clearAllCache**

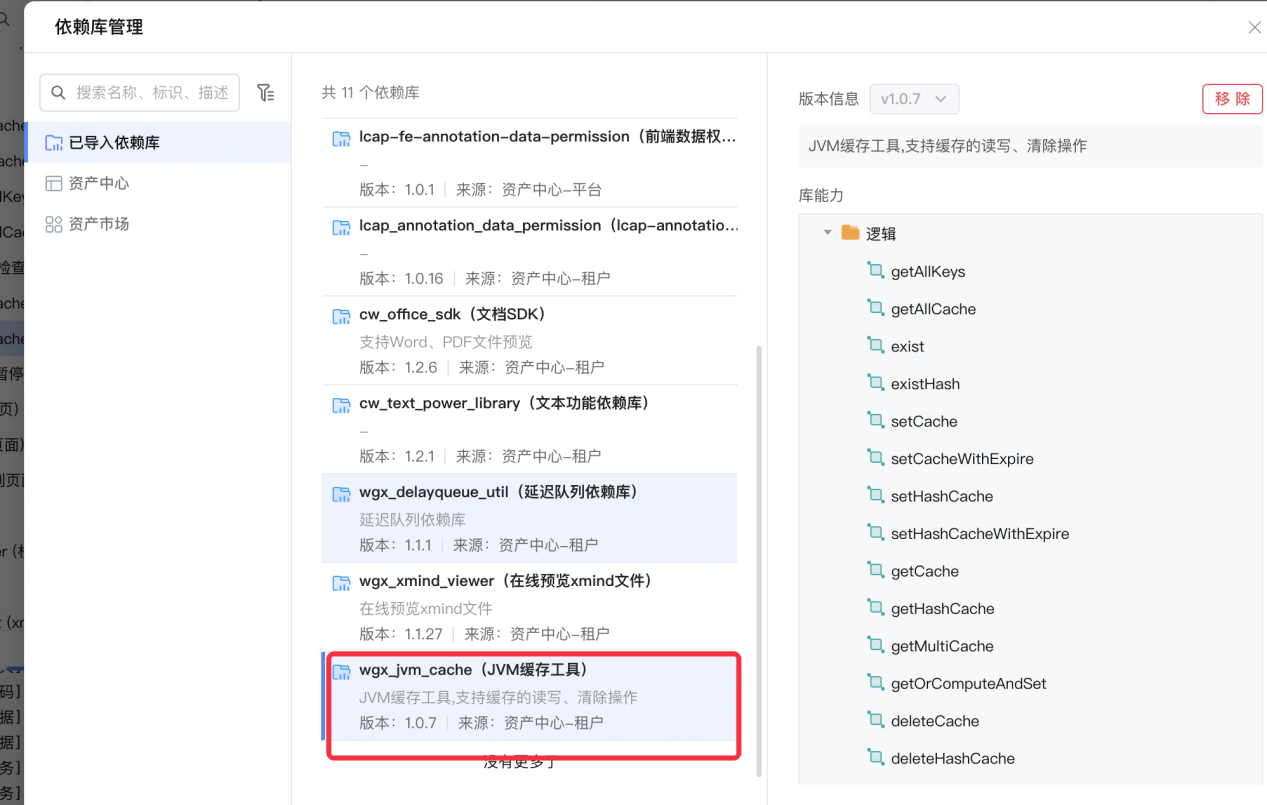
* ****功能描述****：清空所有缓存。
* ****入参****：无。
* ****出参****：Boolean，清空成功返回 true，否则返回 false。
* ****日志记录****：
  + info级别：当清空所有缓存成功时，记录信息。
  + error级别：当清空缓存过程中出现异常时，记录错误信息。

### **16. getEstimatedSize**

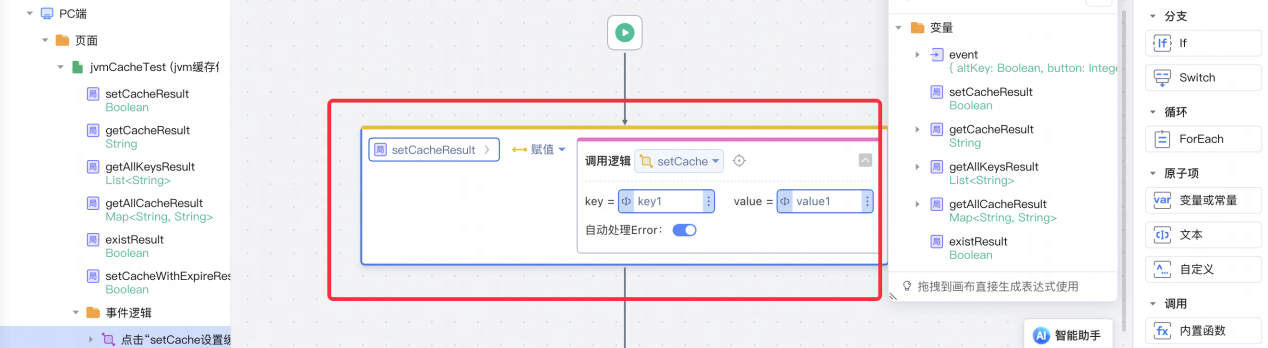
* ****功能描述****：获取缓存的估计大小。
* ****入参****：无。
* ****出参****：Long，缓存中的估计条目数。
* ****日志记录****：
  + info级别：当获取缓存估计大小时，记录当前缓存估计大小。

### 使用案例

3.1引入依赖库



3.2调用方法传入参数



3.4IDE页面内调用显示（测试）



3.5查看结果

